**République Algérienne Démocratique et Populaire**

**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**

**Université A. Mira de Béjaia**


**Faculté des Sciences Exactes**

**Département de Recherche Opérationnelle**

# *Mémoire de Master*

***Option : Modélisation Mathématique et Evaluation de Performances des Réseaux***

*Thème*

# Méthodes d'optimisation dans les réseaux de transport et applications

***Présenté par :***

DAHOU Mohamed Lamine & CHETBANI Aissa

***D*evant le jury composé de :**

| | |
|---|---|
| Présidente : | Mr TAOUINET Smail |
| Rapporteur : | Dr KABYL Kamal |
| Examinatrice : | Dr ZIANE Yasmina |
| Examinateur : | Mr BOUAROURI Wahib |

***Promotion 2017-2018***


***Louange A Dieu, le miséricordieux, sans Lui rien de tout cela n'aurait pu être.***

**N***ous tenons tout d'abord à remercier* $D^r$ ***Kabyl.K****, pour l'honneur qu'il nous a fait en acceptant de nous encadrer. Ces conseils précieux ont permis une bonne orientation dans la réalisation de ce modeste travail.*

**N***ous tenons également à remercier les membres du jury* $D^r$ ***Ziane.Y*** *et* $M^r$ ***Taouinat-S*** *et* $M^r$ ***Bouarouri-W*** *pour l'honneur qu'il nous ont fait en acceptant de juger ce travail, et d'avoir consacrer leurs temps pour sa lecture.*

**N***ous tenons à exprimer notre profonde gratitude à l'ensemble du corps enseignant qui a contribué à notre formation.*

**E***nfin nous tenons à rendre hommage à toutes nos famille et nos amis pour le soutien qu'ils nous ont apportés durant toutes ces années d'études.*


*A cœur veillant, rien d'impossible.*

*A conscience tranquille, tout est accessible.*

*Quand il y a la soif d'apprendre.*

*Tout vient à point à qui sait attendre.*

*Les études sont avant tout notre unique et seul atout.*

*Souhaitant que le fruit de nos efforts fournis jour et nuit*

*Nous mènera vers le bonheur fleuri.*

*Je dédie ce modeste travail:*

*A celle qui m'a donné la vie, le symbole de tendresse, qui s'est sacrifiée pour mon bonheur. et ma réussite, à ma mère.*

*A mon père, école de mon enfance, qui a été mon ombre durant toutes les années des études, Que dieu les gardes et les protégé.*

*A ma très chére sœur .*

*A mes très chers frères .*

*A mes très chéres cousines .*

*A toute ma famille.*

*A mon ami proches Youcef Latreche .*

*A tous mes amis avec lesquels j'ai partagé mes moments de joie et de bonheur.*

***Chetbani Aissa***


*A cœur veillant, rien d'impossible.*
*A conscience tranquille, tout est accessible.*
*Quand il y a la soif d'apprendre.*
*Tout vient à point à qui sait attendre.*
*Les études sont avant tout notre unique et seul atout.*
*Souhaitant que le fruit de nos efforts fournis jour et nuit*
*Nous mènera vers le bonheur fleuri.*

*Je dédie ce modeste travail à :*

*Mes très chère parents en gratitude spécialement (mon père et ma mère).*

*Mon très cher frère.*

*Mes très chéres soeurs.*

*A mes amis et , collègues et mes cammarades .*

*A toutes la promotion de la* $\Re - \emptyset$ *.*

***Dahou Mohamed Lamine***

# TABLE DES MATIÈRES

# TABLE DES FIGURES

# INTRODUCTION

**La Recherche Opérationnelle (RO)** est la discipline des mathématiques appliquées qui traite l'optimalité des ressources dans l'industrie. Depuis une dizaine d'années, le champ d'application de la RO s'est élargi à des domaines comme l'économie, la finance, le marketing et la planification d'entreprise. Plus récemment, la RO a été utilisée pour la gestion des systèmes de santé et d'éducation, pour la résolution de problèmes environnementaux .[9]

La Recherche Opérationnelle est née pendant la Seconde Guerre mondiale des efforts conjugués d'éminents mathématiciens (dont von Neumann, Dantzig, Blackett) à qui il avait été demandé de fournir des techniques d'optimisations des ressources militaires. Le premier succès de cette approche a été obtenue en 1940 par le Prix Nobel de physicien Patrick Blackett qui à résolu un problème d'implantation optimale de radars de surveillance. Le qualificatif " opérationnelle " vient du fait que les premières applications de cette discipline avait trait aux opérations militaires. La dénomination est restée par la suite, même si le domaine militaire n'est plus le principal champ d'application de cette discipline, le mot " opérationnelle " prenant alors plutôt le sens d' "effectif ". Ce sont donc ces mathématiciens qui ont créé une nouvelle méthodologie caractérisée par les mots-clés Modélisation et Optimisation.

Si l'on cherche à trouver des précurseurs à la Recherche Opérationnelle, on peut penser à Alcuin ou à Euler qui se sont intéressés à des problèmes du type RO, bien qu'aucune application n'ait motivé leur travail.

Le premier problème de Recherche Opérationnelle à visée pratique a été étudié par Monge en 1781 sous le nom du problème des déblais et remblais. Considérons $n$ tas de sable, devant servir à combler $m$ trous. Notons $a_i$ la masse du ième tas de sable et $b_j$ la masse de sable nécessaire pour combler le $j^{eme}$ trou. Quel plan de transport minimise la distance totale parcourue par le sable ?

L'importance de l'**optimisation** est la nécessité d'un outil simple pour modéliser des problèmes de décision, qu'ils soient économiques, militaires ou autres,

ont fait de la programmation linéaire un des champs de recherche les plus actifs au milieu du siècle dernier. Les premiers travaux (1947) sont ceux de George B. Dantzig et ses associés du département des forces de l'air des Etats Unis D'Amérique notamment pour résoudre certains problèmes tels que l'implantation optimale de radars de surveillance ou la gestion optimale des convois d'approvisionnement.

**La programmation linéaire** consiste généralement à allouer des ressources limitées, de la meilleure façon possible, afin de maximiser un profit ou de minimiser un coût. Ces décisions sont en général le résultat d'un problème mathématique.

Dans le milieu de l'industrie manufacturière, la recherche opérationnelle permet un meilleur ordonnancement des plans de productions, de mieux disposer les machines dans l'atelier, de diminuer de façon significative le gaspillage des matières premières, d'optimiser la découpe, de mieux gérer la consommation énergétique ou bien encore d'optimiser la livraison des produits.

De nombreuses entreprises dont les clients sont géographiquement dispersés répartissent leur production entre plusieurs établissements dont chacun se caractérise par une localisation et une fonction de coût particulière. Aussi pour assurer l'approvisionnement de leurs clients au moindre coût doivent-elles considérer à la fois le niveau de production de chaque établissement. Elles ont donc à résoudre un problème de transport.[2]

Ce manuscrit est organisé en cinq chapitres précédés par une introduction générale. Dans le premier chapitre nous définissons quelques éléments fondamentaux de la théorie des graphes et de la programmation linéaire. Le deuxième chapitre est consacré à la détermination et la définition du problème de transport classique, affectation, et le problème du flot maximum a coût minimum ainsi que les définitions principales de ce travail. Dans le troisième chapitre nous présentons les algorithmes d'optimisation qui vont être utilisés pour la résolution de quelques problèmes et le quatrième chapitre consacré pour la résolution de problème d'affectation et du flot maximum à coût minimum .

Le cinquième chapitre est consacré à la résolution de certains problèmes d'optimisation dans les réseaux en utilisant le langage C , Les résultats obtenus et leurs interprétations sont relatés à la fin de ce chapitre, nous terminons le manuscrite par une conclusion générale et quelques perspectives.

CHAPITRE 1

# QUELQUES NOTIONS ET NOTATIONS DE BASE POUR L'OPTIMISATION

Un bon dessin vaut mieux qu'un long discours. Le langage des graphes essaie de mettre en pratique cette idée. Bien souvent, en effet, c'est un réflexe naturel qui pousse à abstraire une situation donnée en traçant, sur une feuille de papier, des points pouvant représenter des individus, des localités, des corps chimiques,... La relation entre eux par des lignes ou des flèches symbolise une certaine relation. L'objectif de ce chapitre est de rappeler quelques notions de base de la théorie des graphes, et des points généraux de la programmation linéaire, qui seront utilisés par la suite dans le traitement du problème de transport, d'affectation , et de flot maximum à coût minimum .

## 1.1 Concepts fondamentaux

### 1.1.1 Qu'est ce qu'un graphe ?

Un graphe $G$ est un objet mathématique composé d'un ensemble **V** non vide et fini de points $\{ v_1, v_2, ....v_n\}$ appelés **sommets**(noeuds),et par un ensemble des couples de sommets noté **E** (qui peut être vide) de segments (flèches) $\{e_1, e_2, ..., e_m\}$ appelés **arêtes**(arcs), en reliant chaque paires de sommets $v_1$ et $v_2$ par une arête (arc) $e$ s'écrit comme étant $e = \{v_1, v_2\}$ $(v_1v_2)$, les points $v_1$ et $v_2$ appelés *les extrémités* de $e$.[7]

On appelle **ordre** d'un graphe $G$ le nombre de ses sommets, noté par

$$|V(G)|$$

Soit $x, y \in V$, on dit que $x$ est adjacent à $y$ si $x$ et $y$ sont reliés par un arête(arc)

Si $e = \{x, y\}$ est un arête, et si $x = y$ alors $e$ définit une boucle. en d'autre terme une boucle est une arête reliant un sommet à lui même.[7]
Le graphe $H = (V, E')$ est un graphe partiel de $G$, si $E' \subseteq E$, Autrement dit on obtient $H$ en enlevant une ou plusieurs arêtes au graphe G.[1]
Un sous graphe engendré par $A \subseteq V$ est un graphe dont les sommets sont ceux de $A$ et les arêtes sont celles ayant les deux extrémités dans $A$. on le note $G_A$.[8]

La figure ci-dessous présente un graphe à 5 sommets et 5 arcs

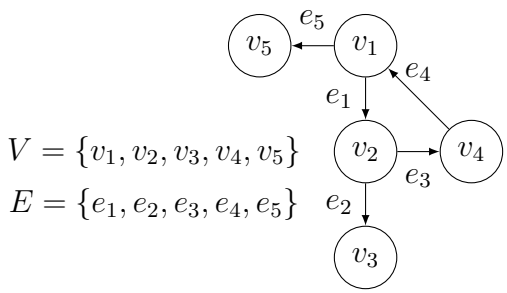



FIGURE 1.1 – Graphe à 5 sommets et 5 arcs

Considérons le graphe correspondant à la Figure 1.1

- De $v_1$ on peut atteindre $v_2$ par $e_1$. Donc, $v_1$ forment l'ensemble des prédécesseurs de $v_2$, qu'on note $\Gamma^-(v_2)$
- De $v_2$ on peut atteindre $v_4$ et $v_3$ par $e_3$ et $e_2$. Donc, $v_4$ et $v_3$ forment l'ensemble des successeurs de $v_2$, qu'on note $\Gamma^+(v_2)$
- L'ensemble des voisins du sommet $v_2$ noté, $N(v_2)$ est égale à la réunion de l'ensemble de ses prédécesseurs et de ses successeurs.

### 1.1.2 Quelques Graphes particuliers

On distingue plusieurs types de graphes :

• **Graphe simple :** Un graphe $G$ est dit simple si tous ses sommets sont sans boucles et entre chaque paire de sommets, il y a au plus une arête.

• **Graphe complet :** Un graphe $G = (V, E)$ est dit complet si tous les sommets sont deux à deux adjacents, le graphe complet à $n$ sommet est noté $K_n$.[8]

• **Graphe d'écart :** Lorsque l'on travaille sur les flots, il est souvent intéressant d'utiliser un graphe spécial, dérivé du graphe initial, nommé graphe d'écart et noté $G(v) = (V, E(v))$. soit $(i, j)$ un arc de $G$ de capacité minimale et maximum respective $l_{ij}$ et $u_{ij}$ et portant le flot $x_{ij}$. Il est alors possible soit :

- d'ajouter encore jusqu'à $u_{ij} - x_{ij}$ unités de flot depuis $i$ vers $j$ .

– de retirer jusqu'à $x_{ij} - l_{ij}$ unités de flot depuis $i$ vers $j$, ce qui peut être vu comme l'ajout d'autant d'unités de flot depuis $i$ vers $j$ .

Le graphe d'écart $G(x)$ va donc proposer deux arcs mettant en avant ces deux possibilités :

$$(i,j) \textit{ de capacité résiduelle } r_{ij} = u_{ij} - x_{ij} \textit{ arc direct de } (i,j)$$

$$(j,i) \textit{ de capacité résiduelle } r_{ji} = x_{ij} \textit{ arc opposé de } (i,j)$$

En outre, seuls les arcs de capacité résiduelle non nulle sont présents dans le graphe d'écart. Notez le cas intéressant où $l_{ij} = u_{ij}$ . Nécessairement, tout flot compatible est tel que $x_{ij} = l_{ij} = u_{ij}$ . Alors, le graphe d'écart ne contient ni l'arc $(i,j)$ , ni l'arc $(j,i)$ car ils ont tous deux une capacité résiduelle nulle. [6]

• **Graphe biparti :** Un graphe $G$ esr dit **biparti** si l'ensemble de ses sommets peut-être divisé en deux sous-ensembles $V_1$ et $V_2$ telle que deux sommets de même sous-ensemble ne soient jamais adjacents, et on note parfois $G = (V_1, V_2, E)$.[8]

La figure 1.2 présente un graphe biparti

$$V_1 = \{1,2,3\}$$

$$V_2 = \{4,5\}$$

$$E = \{\{1,4\};\{1,5\}\{2,4\};\{2,5\};\{3,4\};\{3,5\}\}$$

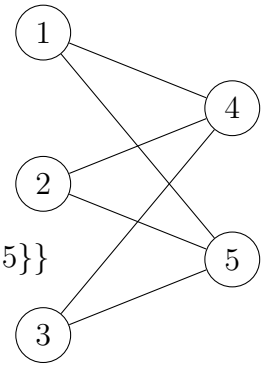



FIGURE 1.2 – Graphe biparti

### 1.1.3 Flux et Flots

#### Flux

Un flux est la quantité $\phi_{ij}$ transportée sur chaque arc $(i,j)$.

#### Flot

Un flot $\varphi$ est déterminé par la donnée du flux pour tout arc du réseau de transport. La valeur d'un flot $V(\varphi)$ est par définition, la somme des flux partant de la source $x_1$ ($V(\varphi)$ est aussi égale à la somme des flux des arcs arrivant sur le puits $x_p$)

La loi de Kirchoff (loi de conservation aux noeuds) est dite le flot entrant égale au flot sortant.

$$\sum_{\varphi \in w^+(i)} V(\varphi) = \sum_{\varphi \in w^-(i)} V(\varphi) \qquad \forall i \in w \backslash \{s,t\}$$

Où le cocycle
$w^+(i)$ est l'ensemble des arcs sortant en $i$.
$w^-(i)$ est l'ensemble des arcs entrant en $i$.

### 1.1.4 Couplage

Étant donné un graphe non orienté, un couplage, ou " matching " en anglais, est un sous-ensemble d'arêtes disjoint deux à deux.

**Le couplage maximum** est le couplage couvrant le plus grand nombre de sommets possibles, en laissant donc le moins de sommets non saturés.[18]

## 1.2 Chaine, Chemin, Cycle et Circuit

### 1.2.1 Chaîne

Une **chaîne** dans un graphe est une suite $\{v_0, v_1, ...., v_k\}$ tel que $v_i$,
$\forall i \in \{1, 2, ..., k\}$, est reliés par une arête de $v_{i-1}$ et une arête à $v_{i+1}$.
• La **longueur** d'une chaîne correspond au nombre d'arêtes parcourues.

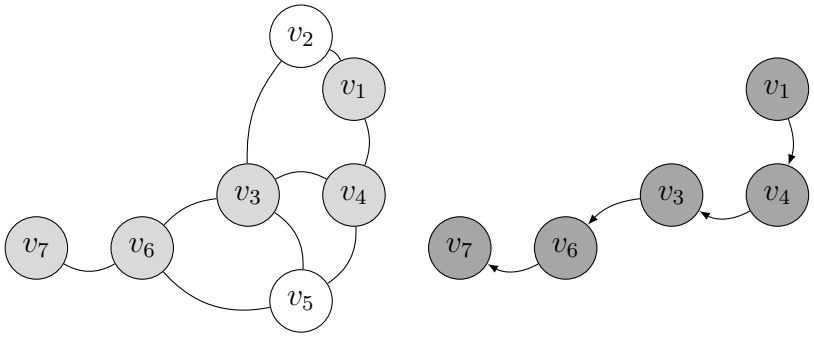
### 1.2.2 Chaîne améliorante

$\mu$ est une chaîne améliorante (ou augmentante ) de $s$ à $t$ pour un flot $\varphi$ admissible donné si :
– $\varphi_{ij} < c_{ij}$ pour tout arc $(i, j)$ de $\mu$ dans le bon sens (de $s$ vers $t$).
– $\varphi_{ij} > 0$ pour tout arc $(i, j)$ de $\mu$ dans le mauvais sens (de $s$ vers $t$).[5]

### 1.2.3 Chemin

Un **chemin** dans un graphe orienté est un suite de sommets $\{v_0, v_1, ...., v_k\}$ tel que $\forall i \in \{1, 2, ..., k\}$, $\exists$ un arcs de $v_{i-1}$ vers $v_i$ et autre arc de $v_i$ vers $v_{i+1}$
• Un chemin p est **simple** si chaque arc du chemin est empruntée une seule fois .

Le graphe de la figure 1.3($b$) présente un chemin de longueur 4 dans le graphe de la figure 1.3($a$)




**(a)** **(b)**

FIGURE 1.3 – Chemin

### 1.2.4 Cycle

Un **cycle** est une chaîne simple finissant à son point de départ.*i.e* on ne rencontre pas deux fois le même sommet, sauf celui choisi comme sommet de départ et d'arrivée.

### 1.2.5 Circuit

Dans un graphe orienté un **circuit** est un chemin tel que le sommet de départ est le même que celui d'arrive, en d'autre terme c'est un chemin qui se ferme sur lui-même.

### 1.2.6 Connexité

Un graphe connexe est un graphe tel que pour toute paire de sommets x et y, il existe une chaîne reliant x et y, un graphe non connexe est décomposé en plusieurs composantes connexes.

### 1.2.7 Arbre

Un **arbre** est un graphe simple connexe acyclique (sans cycle). également un arbre comporte exactement $|V| - 1$ arêtes.

### 1.2.8 Composantes connexes

On appelle **composantes connexes** un ensemble de sommets, qui on deux à deux la relation de connexité, de plus tout sommet en dehors de la composante n'a pas de relation de connexité avec les sommets de cette composante.

• Pour la recherche des composants connexes d'un graphe nous utiliserons l'algorithme de marquage suivant .

**Algorithme de marquage**

L'algorithme de marquage simple est un algorithme qui permet de déterminer les composantes connexe d'un graphe.[5][1]

**Principe :** Soit $G = (V, E)$ un graphe orienté(diagraphe) . L'idée de cet algorithme est la suivante : pour un sommet quelconque $v \in G$, il s'agit de trouver toutes les chaînes reliant ce sommet aux autres sommets du graphe ; ainsi les sommets reliés au sommet $v$ par une chaîne forment la composante connexe qui contient le sommet $v$.

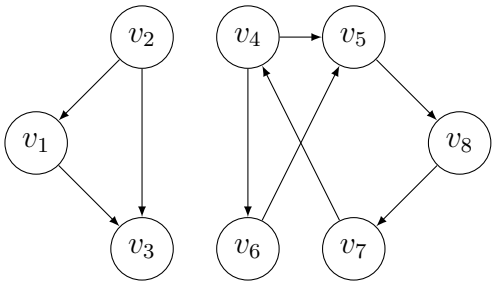
**Enoncé :** $*$Données : un graphe $G = (V, E)$

1. Initialisation $k \leftarrow 0$, $A \leftarrow V$.
2. (a) Choisir un sommet $v_i$ de A et le marquer avec un signe $(+)$, puis marquer tous ses voisins avec le même signe. continuer cette procédure jusqu'à ce qu'on ne puisse plus marquer de sommets.
   (b) Poser $k = k + 1$ et $C_k$ l'ensemble des sommets marqués.
   (c) Retirer de $A$ les sommets de $C_k$ et poser $A = A - C_k$.
   (d) Tester si $A = \emptyset$
   • Si oui terminer ; aller à (3).
   • Si non aller à (2) ;
3. Le nombre de composantes connexes de $(G)$ est $k$.
   Chaque ensemble $C_i$, $i = 1, .., k$ correspond aux sommets d'une composante connexe de $(G)$.

∗**Résultat**: le nombre $k$ de composantes connexes de $G$ ainsi que la liste $\{C_1, C_2, ...C_k\}$ de ses composantes.

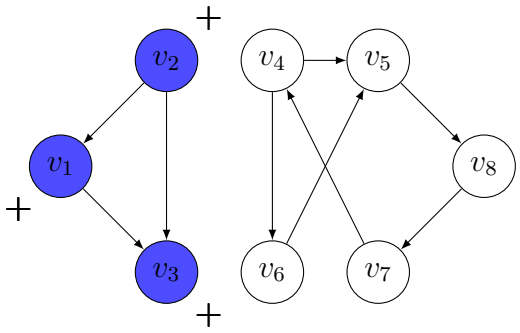
Le graphe de la figure suivant n'est pas connexe car il n'existe pas de chaîne reliant les sommets $v_1$ et $v_6$.

Soit le graphe $G = (V, E)$ suivant :



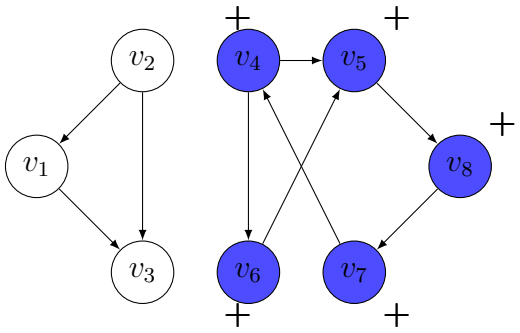

- On peut le vérifie par l'algorithme de marquage précédent pour déterminer les composantes connexes :

• Initialisation :
$k = 0$.
$A = V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$.
• **Itération 1 :**
on choisir dans $A$ le sommet $v_1$, et on le marque d'un signe (+), on marque ensuite ses voisins $v_2$ et $v_3$.




Soit $C_1 = \{v_1, v_2, v_3\}$ l'ensemble des sommets marqués.
On retire de $A$ les sommets $C_1$, on obtient :$A = \{v_4, v_5, v_6, v_7\} \neq \emptyset$.

• **Itération 2 :**
on choisir dans $A$ le sommet $v_4$, et on le marque d'un signe (+), on marque ensuite ses voisins $v_5$,$v_6$ et $v_7$.




Soit $C_2 = \{v_4, v_5, v_6, v_7, v_8\}$ l'ensemble des sommets marqués.
On retirons de $A$ les sommets de $C_2$, on obtient : $A = \emptyset$ **terminer.**

## 1.3 Représentations matricielle des graphes

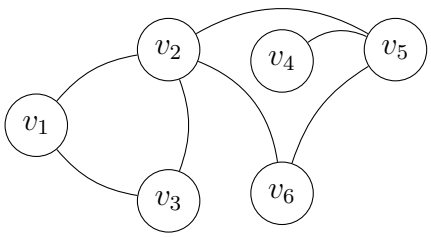
### 1.3.1 Matrice d'adjacence

Soit le graphe $G = (V, E)$ d'ordre $n$. On suppose que les sommets de $V$ sont numérotés de 1 à $n$. La représentation par matrice d'adjacence de $G$ consiste en une matrice booléenne $M$ de taille $n * n$ telle que :

$$M_{ij} = \begin{cases} 1 & \text{si } (ij) \in E. \\ 0 & \text{sinon.} \end{cases}$$

• Dans le cas d'un graphe $G$ non orienté, la matrice d'adjacence $M$ est symétrique .[11]

La matrice d'adjacence associée au graphe de la figure ci-dessous est donnée par :




$$M = \begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{matrix} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix}$$

FIGURE 1.4 – Graphe à 6 sommets et 7 arcs et la matrice adjacence associée

### 1.3.2 Matrice d'incidence (sommet arêtes)

Une **matrice d'incidence** est une représentation matricielle d'un graphe montrant la relation d'incidence entre arêtes et sommets.[11]

• Soit $G$ un graphe orienté sans boucle $G=(V,E)$ comportant $n$ sommets $\{ v_1, v_2, ....v_n\}$ ,et $m$ arêtes $\{ e_1, e_2, ....e_m\}$.On appelle matrice d'incidence $M=(m_{ij})$ de dimension $n * m$ telle que :

$$M_{ij} = \begin{cases} 1 & \text{si } v_i \text{ est l'extrémité initiale de } e_j. \\ -1 & \text{si } v_i \text{ est l'extrémité términale de } e_j. \\ 0 & \text{si } v_i \text{ n'est pas une extrémité de } e_j. \end{cases}$$

• La matrice d'incidence d'un graphe $G$ non orienté sans boucle définie par :

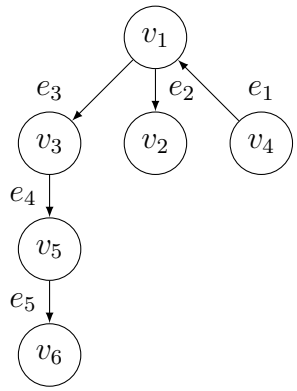
$$M_{ij} = \begin{cases} 1 & \text{si } v_i \text{ est une extrémité de } e_j, \\ 0 & \text{sinon} \end{cases}$$

• La matrice d'incidence d'un graphe $G$ non orienté avec boucle définie par :

$$M_{ij} = \begin{cases} 1 & \text{si } v_i \text{ est l'extrémité de } e_j. \\ 2 & \text{si } v_i \text{ admet un boucle.} \\ 0 & \text{si } v_i \text{ n'est pas une extrémité de } e_j. \end{cases}$$

La matrice d'incidence associée au graphe de la figure 1.5 est donnée par :

soit le graphe suivant :




$$M = \begin{array}{c} \begin{matrix} e_1 & e_2 & e_3 & e_4 & e_5 \end{matrix} \\ \begin{pmatrix} -1 & 1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & -1 \end{pmatrix} \end{array} \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix}$$

FIGURE 1.5 – Graphe à 6 sommets et 5 arêtes sans boucle et la matrice d'incidence associée

## 1.4 Programmation linéaire

La programmation linéaire est un outil très puissant de la recherche opérationnelle. C'est un outil générique qui peut résoudre un grand nombre de problèmes. En effet, une fois un problème modélise sous la forme d'équations linéaires, des méthodes assurent la résolution du problème de manière exacte. On distingue dans la programmation linéaire, la programmation linéaire en nombres réels, pour laquelle les variables des équations sont dans $\mathbb{R}^+$ et la programmation en nombres entiers, pour laquelle les variables sont dans $\mathbb{N}$ . La résolution d'un problème avec des variables entières est nettement plus compliquée qu'un problème en nombres réels.

Une des méthodes les plus connues pour résoudre des programmes linéaires en nombre réels est la méthode du Simplex. En théorie, elle a une complexité non polynomiale et est donc supposée peu efficace. Cependant, en pratique, il s'avère au contraire qu'il s'agit d'une bonne méthode.

Un programme linéaire est la maximisation où la minimisation d'une fonction linéaire sous contraintes linéaires.

• La transposée $A^T$ (aussi notée $A'$) d'une matrice $A$ s'obtient par symétrie axiale par rapport à la diagonale principale de la matrice. La transposée de la transposée $(A^T)^T$ est la matrice $A$ d'origine. La transposée d'un vecteur colonne est un vecteur ligne.[4]

### 1.4.1 Forme générale d'un programme linéaire

$$\begin{cases} z = \sum_{j=1}^{n} c_j x_j & (1)\ . \\ \forall i = 1,..,m : \sum_{j=1}^{n} a_{ij} x_j \leq, = o\text{ù} \geq b_i & (2)\ . \\ \forall j = 1,..,n : x_{ij} \geq 0 & (3)\ . \end{cases}$$

(1) : fonction objective à minimiser où à maximiser.
(2) : $m$ contraintes linéaires.
(3) : Contraintes de positivité.

[4][1]

### 1.4.2 Formes matricielles classiques et conventions

Notons par $x = (x_1, x_2, ..., x_n)^T$ le vecteur des variables. $b = (b_1, b_2, ..., b_m)^T$ le second membre des contraintes , $c = (c_1, c_2, ..., c_n)^T$ le vecteur coût où profit associé aux variables et $A$ la matrice ayant comme éléments $a_{ij} \quad {\scriptstyle 1\leq i\leq m} \quad {\scriptstyle 1\leq j\leq n}$.

$$\text{Forme canonique} : \begin{cases} \max z = cx \\ Ax \leq b \\ x \geq 0 \end{cases} .$$

$$\text{Forme standard} : \begin{cases} \max z = cx \\ Ax = b \\ x \geq 0 \end{cases} .$$

La forme canonique avec des contraintes $\leq$ s'utilise dans la représentation graphique, et la forme standard avec des contraintes égalité s'utilise dans la résolution algébrique.

## 1.5 Complexité algorithmique

La complexité (temporelle) d'un algorithme est le nombre d'opérations élémentaires (affectations, comparaisons, opérations arithmétiques) effectuées par un algorithme. Ce nombre s'exprime en fonction de la taille $n$ des données.

On s'intéresse au temps exact quand c'est possible, mais également au temps moyen (que se passe-t-il si on moyenne sur toutes les exécutions du programme sur des données de taille $n$), au cas le plus favorable, ou bien au cas le pire. On dit que la complexité de l'algorithme est $O(f(n))$ où $f$ est d'habitude une combinaison de polynômes, logarithmes ou exponentielles.

Ceci reprend la notation mathématique classique, et signifie que le nombre d'opérations effectuées est borné par $cf(n)$ où $c$ est une constante, lorsque $n$ tend vers l'infini.

Considérons le comportement à l'infini de la complexité est justifié par le fait que les données des algorithmes sont de grande taille et qu'on se préoccupe surtout de la croissance de cette complexité en fonction de la taille des données. Une question systématique à poser est : que devient le temps de calcul si on multiplie la taille des données par 2 ? De cette façon, on peut également comparer des algorithmes entre eux.[6]

### 1.5.1 Notation O(.)

La théorie de la complexité algorithmique vise à répondre à ces besoins. Elle permet :

1. De classer les problèmes selon leur difficulté ;
2. De classer les algorithmes selon leur efficacité ;
3. De comparer les algorithmes sans devoir les implémenter.

Comme on l'a vu dans la section précédente, les calculs à effectuer pour évaluer le temps d'exécution d'un algorithme peuvent parfois être longs et pénibles. De plus, le degré de précision qu'ils requièrent est souvent inutile. On aura donc recours à une approximation de ce temps de calcul, représentée par la notation O(.) .[15]

### 1.5.2 Calcule de la complixité d'un algorithme

La notation O(.) va nous permettre de quantifier l'efficacité d'un algorithme sous certaines hypothèses :

**Définition :** La complexité d'un algorithme est la mesure **asymptotique** de son temps d'exécution **dans le pire des cas**. Il s'exprime à l'aide de la notation O(.) en fonction de la taille des données reçues en entrée .[17]

Les deux précisions sur le caractère de cette mesure important :

1. Asymptotique signifie que l'on s'intéresse à des données très grandes ; en effet, les petites valeurs ne sont pas assez informative .
2. "Dans le pire des cas " signifie que l'on s'intéresse à la performance de l'algorithme dans les situations où le problème prend le plus de temps à résoudre ; on veut être sur que l'algorithme ne prendra jamais plus de temps que ce qu'on a estimé .

Le calcul de la complexité d'un algorithme s'effectue de manière similaire à celui du temps d'exécution, si ce n'est qu'on remplace maintenant les unités de temps par les approximations fournies par la notation O(.) . Plus précisément :

- Chaque instruction basique (affectation d'une variable, comparaison, +,/ ,*,..... ) consomme un temps constant représenté par la notation O(1) ;
- Chaque itération d'une boucle rajoute la complexité de ce qui est effectué dans le corps de cette boucle ;
- Chaque appel de fonction rajoute la complexité de cette fonction ;
- Pour obtenir la complexité de l'algorithme, on additionne le tout.

On aura aussi recours aux simplifications suivantes :

1. On oublie les constantes multiplicatives (elles valent 1) ;
2. On annule les constantes additives ;

3. On ne retient que les termes dominants (veut dire la grande puissance).

Dans ce chapitre, nous avons rappelé quelques notions de base concernant la théorie de graphe, la programmation linéaire, et la complexité algorithmique qui seront utilisées par la suite .

# CHAPITRE 2

# PROBLÈME DE TRANSPORT

Ce chapitre a l'objectif de présenter et de modéliser le problème de transport par ses différentes formulations, Il s'agit d'un type de problème de programmation linéaire qui peut être énoncer comme suit : " Comment transporter aux moindres coûts entre $m$ origines $X_i = \{x_{1,.....,X_m}\}$ et $n$ destinations $Y_i = \{Y_{1,.....,Y_n}\}$ . Les disponibilités $a_i(i = 1, ..., m)$ existantes aux origines $x_i(i = 1, ..., m)$ , afin de satisfaire les demandes $b_j(j = 1, ..., n)$ des destinations $y_j(j = 1, ..., n)$ .étant données $m \times n$ coûts de transport $c_{ij}$.

## 2.1 Positionnement du problème

Dans un problème de transport, nous avons certaines origines, ce qui peut représenter les usines où nous avons produit des articles et fourni une quantité requise de produits à un certain nombre de destinations. Cela doit être fait de manière à maximiser le profit où à minimiser le coût. Ainsi, nous avons les lieux de production comme origines et les lieux d'approvisionnement comme destinations.

## 2.2 Modélisation

Supposons qu'une entreprise ait $m$ entrepôts et $n$ points de vente, un seul produit doit être expédié des entrepôts aux points de vente. Chaque entrepôt (origine) a un niveau d'approvisionnement donné (disponibilité), et chaque point de vente (destination) a un niveau de demande donné. On nous donne également le coût de transport entre chaque paire d'entrepôt et de destination, telles que :

- La disponibilité de chaque entrepôt $i$ est : $a_i$ unité, ou $i = 1, 2, 3..., m$.
- La demande de chaque destination $j$ est : $b_i$ unité, ou $j = 1, 2, 3..., n$.
- Le coût de transport d'une unité du produit de l'entrepôt $i$ à la destination $j$ est égal à $c_{ij}$ unité.

Où $i \in \{1, 2, 3..., m\}$ et $j \in \{1, 2, 3..., n\}$ Le coût total d'une expédition est linéaire en taille d'expédition.

### 2.2.1 Variables de décision

Les variables du modèle de programmation linéaire (PL) du problème de transport sont des entiers naturels représentant des unités transportées d'une source vers une destination. Les variables de décision sont les suivantes :

$x_{ij}$ : La quantité à transporter de la source $i$ vers la destination $j$, où $i \in \{1, 2, 3...m\}$ et $j \in \{1, 2, 3, ...n\}$.

### 2.2.2 Fonction objective

le problème consiste à déterminer les quantités $x_{ij}$ à transporter de façon que le coût total de transport $\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$ soit minimal.

La fonction objective contient des coûts associés à chacune des variables. C'est une minimisation de problème. Puisque nous supposons que la fonction coût total est linéaire, Le coût total de cette expédition est donné par $c_{ij} \times x_{ij}$. En sommant sur tout $i$ et $j$, on obtient le coût global de transport pour tous les entrepôts.[12]

### 2.2.3 Contraintes

Les contraintes sont les conditions qui obligent à satisfaire la demande et épuiser la disponibilité. Dans un Problème de transport, il existe une contrainte pour chaque sommet. Posons : $a_i$ désigne une capacité d'une source (disponibilité) et $b_j$ désigne le besoin d'une destination (demande).

Les contraintes sont :

- La disponibilité à chaque source doit être épuisée :$\sum_{j}^{n} x_{ij} = a_i, i \in \{1, \ldots, m\}$.
- La demande à chaque destination doit être satisfaite : $\sum_{i}^{m} x_{ij} = b_j, j \in \{1, \ldots, n\}$.
- La non négativité des quantités :$x_{ij} \geq 0 \forall i \in \{1, \ldots, m\}; \forall j \in \{1, \ldots, n\}$

### 2.2.4 Formulation mathématique

$$\begin{cases} \min z = \sum_{i}^{m}\sum_{j}^{n} c_{ij}x_{ij} \quad . \\ \quad \sum_{j}^{n} x_{ij} = a_i \qquad \forall i \in \{1, ...., m\} \ . \\ \quad \sum_{i}^{m} x_{ij} = b_j \qquad \forall j \in \{1, ...., n\}. \\ \quad a_i \in \mathbb{R}^+ \qquad \forall i \in \{1, ...., m\} \ . \\ \quad b_i \in \mathbb{R}^+ \qquad \forall j \in \{1, ...., n\}. \\ \quad x_{ij} \in \mathbb{R}^+ \qquad \forall (i,j) \in \{1, ...., m\} \times \{1, ...., n\}. \end{cases}$$

Il s'agit d'un programme linéaire avec $m \times n$ variables de décision, $m + n$ contraintes fonctionnelles et $m \times n$ contraintes non négatives.
$m$ : Nombre de sources.
$n$ : Nombre de destinations.
$a_i$ : Disponibilité de la $i^{ème}$ source.
$b_j$ : Demande de la $j^{ème}$ destination .
$c_{ij}$ : Coût unitaire de transport de la $i^{ème}$ source à la $j^{ème}$ destination .
$x_{ij}$ : Quantité transportée de la $i^{ème}$ source à la $j^{ème}$ destination .[14]

Une condition nécessaire et suffisante pour l'existence d'une solution réalisable au problème transport est que :

$$\sum_{i}^{m} a_i = \sum_{j}^{n} b_j \qquad \forall (i,j) \in \{1, ...., m\} \times \{1, ...., n\}$$

.

## 2.3 Tableau de transport

Le modèle d'un problème de transport peut être représenté sous forme de tableau concis avec tous les paramètres pertinents.

Le tableau de transport (Un problème de transport typique est représenté sous forme de matrice standard), où la disponibilité d'approvisionnement ($a_i$) à chaque source est affichée dans la colonne droite du tableau, et les demandes de destination ($b_j$) sont affichées dans la ligne inférieure.

Chaque cellule représente une voie, le coût de transport unitaire ($c_{ij}$) est indiqué dans le coin supérieur droit de la cellule, la quantité de matériel transporté est affichée au centre de la cellule, le tableau de transport exprime implicitement les contraintes de l'offre, de la demande et le coût de transport entre chaque source et destination.[13]

la figure 2.1 présente un tableau de problème de transport :

| Destination : → / Source : ↓ | $D_1$ | $D_2$ | $\dots D_j \dots$ | $D_n$ | Disponibilités |
|---|---|---|---|---|---|
| $S_1$ | $C_{11}$ $x_{11}$ | $C_{12}$ $x_{12}$ | | $C_{1n}$ $x_{1n}$ | $a_1$ |
| $S_2$ | $C_{21}$ $x_{21}$ | $C_{22}$ $x_{22}$ | | $C_{2n}$ $x_{2n}$ | $a_2$ |
| $\dots S_i \dots$ | | | $C_{ij}$ $x_{ij}$ | | $a_i$ |
| $S_m$ | $C_{m1}$ $x_{m1}$ | $C_{m2}$ $x_{m2}$ | | $C_{mn}$ $x_{mn}$ | $a_m$ |
| Demandes | $b_1$ | $b_2$ | $\dots b_j \dots$ | $b_m$ | $\sum a_i$ / $\sum b_j$ |

FIGURE 2.1 – Tableau de transport

## 2.4 Réseau de transport

Un réseau de transport (aussi appelé réseau de flot ) est un graphe fini, orienté sans boucle où chaque arête possède une capacité et peut recevoir un flot (ou flux). Le cumul des flots sur une arête ne peut pas excéder sa capacité. Les sommets sont alors appelés des nœuds et les arêtes des arcs. Pour qu'un flot soit valide, il faut que la somme des flots atteignant un nœud soit égale à la somme des flots quittant ce nœud, sauf s'il s'agit d'une source (qui n'a pas de flot entrant), ou d'un puits (qui n'a pas de flot sortant). Un réseau peut être utilisé pour modéliser le trafic dans un réseau routier, la circulation de fluides dans des conduites, la distribution d'électricité dans un réseau électrique, ou toutes autres données transitant à travers un réseau de nœuds.

Graphiquement, le problème du transport est souvent visualisé comme un réseau avec $m$ sommets sources, $n$ sommets destinations et un ensemble de $m \times n$ "arcs orientés " Ceci est représenté dans la figure 2.2

Dans la figure 2.2 , il y a $x_1 \dots x_m$ sources et $y_1 \dots y_n$ destinations. Les arcs orientés montrent des flux de transport de source vers destination. Chaque destination est liée à chaque source par un arc, le nombre $(c_{11} \dots c_{mn})$ au-dessus de chaque arc représente le coût du transport sur cette route. Les problèmes avec la structure ci-dessus se posent dans de nombreuses applications. Par exemple, les sources pourraient être représenté des entrepôts, des puits,. . . etc , et les destinations pourraient représenté des populations, des clients,...etc.

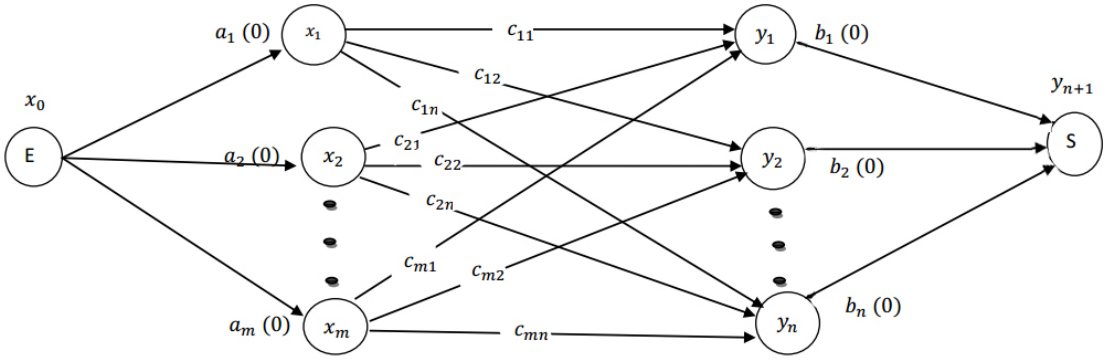



FIGURE 2.2 – Réseau de transport

• **La dégénérescence** existe dans un problème de transport lorsque le nombre de cellules remplies est inférieur à $(m + n - 1)$.

La dégénérescence peut être observée soit lors de l'attribution initiale lorsque la première entrée dans une ligne où une colonne satisfait à la fois aux exigences de la ligne et de la colonne où lors de l'application d'une méthode de résolution de problème de transport, lorsque les valeurs ajoutées et soustraites sont égales.[13]

Le transport avec m-origines et n-destinations peut avoir $(m+n-1)$ variables de base positives, sinon la solution de base dégénèrera, Donc à chaque fois que le nombre de cellules basiques est inférieur à $m + n - 1$, le problème du transport est dégénéré.

Pour résoudre la dégénérescence, les variables positives sont augmentées par autant de variables à valeur nulle que nécessaire pour compléter les $(m + n - 1)$ variables de base.

• **Si :**

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

Ce problème du transport est connu comme un problème de transport non équilibré . On distingue deux Cas :[4]

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

Où

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

– (a) $\sum_{i}^{m} a_i > \sum_{j}^{n} b_j$ dans ce cas il faut introduire une destination fictive $y_{n+1}$ de coût de transport égale à zéro entre $x_i$ et $y_{n+1}(i = 1, \ldots, m)$ dont la demande :

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

– (b) $\sum_{i}^{m} a_i < \sum_{j}^{n} b_j$ dans ce cas il faut introduire une source fictive $x_{n+1}$ de coût de transport égale à zéro entre $y_j$ et $x_{m+1}(i = 1, \ldots, m)$ dont disponibilité :

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

## 2.5 Problème de flot maximum à coût minimum

Considérons un réseau $G = (V, E, U, C)$ un réseau avec : $V$=ensemble de nœuds $|V| = n$.
$E$= ensemble des arcs $|E| = m$.
$U$ : capacité des arcs , $U \in \mathbb{R}^m$.
$C$ : coûts de transport sur les arcs .

Le problème de flot à coûts minimum consiste à acheminer (transporter) une quantité de flot $f$ d'une source $s$ vers une destination $t \in V$ de sorte que le cout totale du transport soit minimum .

Ce problème se modélisé par un programme linéaire comme suit :

$$\begin{cases} \min z = \sum\limits_{(i,j)\in E} c_{ij}x_{ij} \\ \sum\limits_{j\in\Gamma^{+}_{(i)}} x_{ij} - \sum\limits_{j\in\Gamma^{-}_{(i)}} x_{ij} = b_i = \begin{cases} f & si\ i = s \\ -f & si\ i = t \\ 0 & sinon \end{cases} \\ 0 \leq x_{ij} \leq U_{ij} \qquad\qquad (i,j) \in E \end{cases}$$

$x_{ij}$ : La quantité ( flot ) transporté de $i$ vers $j$ .

On peut présenter un problème du transport sous la forme matricielle comme suit :

$$\begin{cases} \min z = c^T x \\ Ax = b \\ 0 \leq x \leq \mu \end{cases}$$

Où $\mu \in \mathbb{R}^n, b \in \mathbb{R}^m, A \in \mathbb{R}^{n*m}$ et la matrice d'incidence sommets arcs associe au réseau , et $c^T$ est la transposée de vecteur des coûts .

### 2.5.1 Cas particulier d'application du problème de flot à coût minimum

- **problème de flot maximum**

– On ajoute un arc fictif de $t$ vers $s$ , avec $C_{ts} = -1$ , $U_{ts} = +\infty$.
– Pour les arcs $(i, j) \in E$ , on a $C_{ij} = 0$.
– $b_i = 0$ , $\forall i \in V$

- **Problème classique de transport** Soit $G = (V_1 \bigcup V_2, E)$ un graphe

biparti, Où $V_1$ est un ensemble de fournisseurs qui desservent un ensemble de clients : $E = \{(i, j), i \in V_1 etj \in V_2\}$.

On a $b_i > 0$ pour tous nœud $i \in V_1$ et $b_j < 0$ pour tous nœud $j \in V_2$ .

## 2.6 Problème d'affectation

### 2.6.1 Présentation

Il s'agit d'un cas particulier du problème de transport (Problème de transport équilibré )avec $n$ entrepôts et $n$ magasins, et où la demande associée à chaque destination égale à 1. Le problème consiste à affecter les éléments d'un ensemble à ceux d'un autre ensemble de sorte que la somme des coûts des affectations soit minimale.

### 2.6.2 Formulation mathématique

Le programme à résoudre est :

$$\begin{cases} \min z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} & \\ \quad \sum_{j=1}^{n} x_{ij} = 1 & \forall i = \{1, ...., m\} \\ \quad \sum_{i=1}^{m} x_{ij} = 1 & \forall j = \{1, ...., n\} \\ \quad x_{ij} \in \{0, 1\} & \forall i \in \{1, ...., m\} \text{ et } \forall j \in \{1, ...., n\} \end{cases}$$

$x_{ij} = 1$ si $i$ est affecté à $j$ .
$x_{ij} = 0$ si $i$ n'est pas affecté à $j$.
$C_{ij}$ : coût d'affectation de $i$ à $j$.

Les contraintes de ce problème se retrouvent dans de nombreuses applications mettant en jeu des problèmes d'allocation de ressources. Elles sont généralement appelées "contraintes d'affectation ".

En théorie des graphes, on peut se ramener à un "problème de couplage dans un graphe biparti" [10].

En associant $X_1$ à l'ensemble des tâches, de cardinalité $m$ et $X_2$ à l'ensemble des agents, de cardinalité $n$, une arête $(i, j)$ dans le graphe G (avec $i \in X_1$ et $j \in X_2$) représente la possibilité d'affecter la tâche $i$ à l'agent $j$ ; on associe le poids $c_{i,j}$ à chaque arête $(i, j)$ de $G$. Le poids d'un couplage étant défini comme la somme des poids de ses arêtes, le problème d'affectation revient alors à chercher un couplage de cardinalité $m$ de poids minimal dans le graphe G.

Le cas particulier où $X_1$ et $X_2$ sont de même cardinalité (correspondant au cas $n = m$ pour le problème d'affectation) est fréquemment étudié ; on s'intéresse alors à la recherche d'un couplage de cardinalité maximum. Si on considère des ensembles $X_1$ et $X_2$ de cardinalité $n$ et s'il existe $n^2$ arêtes dans le graphe $G$ (le graphe biparti est complet), alors le couplage maximum est de cardinalité $n$ et il est appelé " couplage parfait ". On peut étendre ce problème à celui de la recherche d'un couplage maximum de poids minimal dans G.

Le problème d'affectation, ou de couplage dans un graphe biparti, peut être modélisé comme un problème de flot maximum à coût minimal dans lequel les capacités des arcs sont toutes égales à 1. C'est un problème classique de la théorie des graphes qui revient à maximisé le débit à faire passer à travers un réseau, pour un moindre coût. Des algorithmes simples et efficaces existent pour résoudre ce problème ; en particulier l'algorithme de Busaker et Gowen qui part d'un flot nul et qui l'augmente progressivement par recherche de "chaînes augmentantes " .

La "méthode Hongroise", est un algorithme dual qui s'appuie sur une modélisation du problème d'affectation sous forme d'un programme linéaire, mais qui peut être vu comme une variante de l'algorithme de Busaker et Gowen, spécialisée pour la structure biparti du graphe. Du fait de sa grande efficacité sur ce type de problème, c'est l'algorithme de référence en Recherche Opérationnelle pour résoudre le problème d'affectation. Son principe est basé sur le fait que les couplages de poids minimal dans le graphe du problème primal sont exactement les couplages de cardinalité maximum dans le graphe du problème dual (voir par exemple [10, 15] pour une présentation détaillée de la méthode).

### 2.6.3 Modélisation par un problème de flot maximum

Le problème d'affectation est représenté par un graphe biparti $G_1 = (S_1, A_1)$ avec $S_1 = X \cup Y$ où $X = L_i,\ \ 1 \leq i \leq m$ et $\quad Y = c_i,\ \ 1 \leq j \leq n \quad$ et $A_1 = \{(L_i, C_j)$ : la tâche $C_j$ est permise pour la machine $L_i$ $\}$.

Le problème de l'affectation peut se modéliser par la recherche d'un flot maximum à travers le réseau $G = (S, A, c)$ obtenu à partir du graphe biparti $G_1 = (S, A)$ comme suit :

- On ajoute une source $s$ et des arêtes $(s, L_i)$ reliant $s$ aux offres $L_i$ avec des capacités $c(s, L_i) = a_i = 1$.
- On ajoute un puits $p$ et des arêtes $(C_j, p)$ reliant les demandes $C_j$ à $t$ avec des capacités $c(C_j, p) = b_i = -1$ .
- Pour chaque arête $(L_i, C_j)$ entre $Li$ et $Cj$ , on considère une capacité infinie $(+\infty)$.

On a : $S = S_1 \cup s, p \quad$ et $\quad A = A_1\{(s, L_i), 1 \leq i \leq m\} \cup \{(C_j, p), 1 \leq j \leq n\}$

La figure 2.3 présente une modélisation du problème d'affectation par le flot maximum

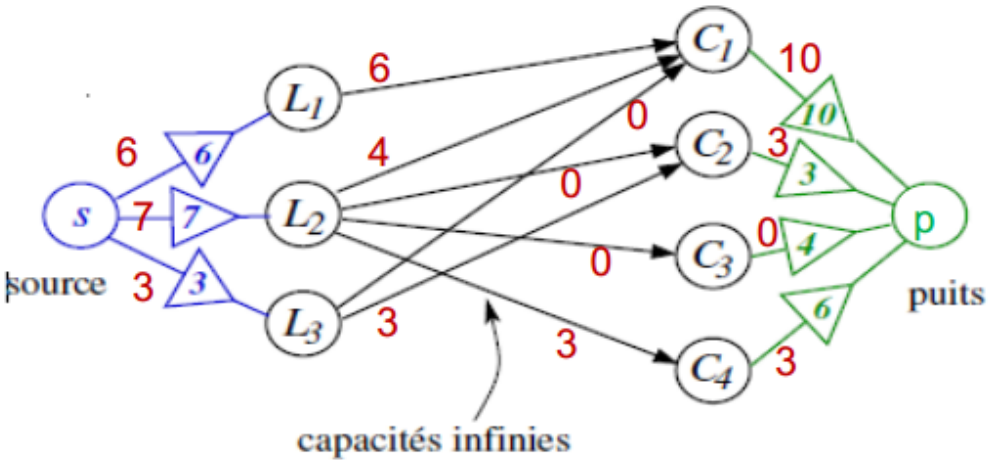



FIGURE 2.3 – Modélisation du problème d'affectation par le flot maximum

On peut déterminer le flot maximum sur ce réseau par application de l'algorithme de Ford-Fulkerson. Le maximum d'affectation possible est 16.

### 2.6.4 Problème d'affectation quadratique

Il existe des problèmes appartenant à de nombreux domaines, aussi variés que l'électronique, économie, l'informatique,...etc, pour lequel la fonction de cout est quadratique, le problème d'affectation "affectation quadratique ", c'est un problème NP complet et donc beaucoup plus difficile à résoudre que l'affectation linéaire .

#### Programme quadratique équivalent

Résoudre un problème d'affectation quadratique, revient à résoudre le programme quadratique suivant : Le problème d'affectation quadratique est noté $A, Q$

$$\begin{cases} \min z = \sum_{i\in I}\sum_{j\in J}\sum_{k\in I}\sum_{l\in J} c_{ijkl}\ x_{ij}\ x_{kl} & \\ \quad \sum_{j\in J} x_{ij} = 1 & \forall i \in I \\ \quad \sum_{i\in I} x_{ij} = 1 & \forall j \in J \\ \quad x_{ij} \in \{0,1\} & \forall\forall i \in I et \forall j \in J \end{cases}$$

$c_{ijkl}$ représente le cout lié à l'affectation respective de i en j et de k en l .

En économie le problème fut posé sous le nome anglais " indivisible ressources allocation problème. Il s'agit d'implanter diverses unités de production à des emplacement différents déjà déterminés .Des quantités fixées de produits doivent circuler entre ces usines, et leur coût de transport dépend de la distance parcourue. L'implantation des usines cherchées est évidemment celle qui assurera un cout de transport minimale :

$I$ est l'ensemble des n usines .

$J$ celui des n emplacements possibles .

$x_{ij} = 1$ Si l'usine i est implantée en j , $x_{ij} = 0$ sinon .

$f_{ik}$ la quantité de produit à transporter de l'usine $i$ à l'usine $k$ .

$d_{jl}$ le cout de transport d'une unité de produit de l'emplacement $j$ à l'emplacement $l$ .

$c_{ijkl} = f_{ik}.d_{jl}$

$$\begin{cases} \min z = \sum\limits_{i\in I}\sum\limits_{j\in J}\sum\limits_{k\in I}\sum\limits_{l\in J} c_{ijkl}x_{ij}x_{kl} & . \\ \sum\limits_{j\in J} x_{ij} = 1 & \forall i \in I. \\ \sum\limits_{i\in I} x_{ij} = 1 & \forall j \in J\ . \\ x_{ij} \in \{0,1\} & \forall\forall i \in I et \forall j \in J\ . \end{cases}$$

# CHAPITRE 3

# MÉTHODES DE RÉSOLUTION DU PROBLÈME DE TRANSPORT

Le problème du transport est un problème linéaire que peut être représenté sous forme d'un graphe et qu'on peut le résoudre en utilisant les différentes méthodes de résolution des problèmes linéaires qu'on va présenter par la suite.

Dans ce chapitre nous allons présenter deux méthodes graphique (Stepping-Stone, distribution modifiée) pour la recherche de la solution optimale et les méthodes :Coin Nord-Ouest, coût minimum, approximation de vogel pour la recherche de la solution de base d'un problème de transport.

## 3.1 Structure de la résolution de problème de transport

Considérons un problème de transport impliquant $m$ origines et $n$ destinations. Etant donné que la somme des disponibilités d'origine est égale à la somme des demandes de destination, une solution réalisable existe toujours. La $(m+n)$ $i^{ème}$ contrainte est redondante et peut donc être supprimée. Cela signifie également qu'une solution de base réalisable pour un problème de transport peut avoir au plus $(m+n-1)$ composants strictement positifs, Sinon la solution dégénérera. Il est toujours possible d'assigner une solution réalisable initiale à un problème de transport. De telle sorte que les exigences des destinations soient satisfaites. Cela peut être réalisé soit par une inspection, soit par des règles simples. Nous commençons par imaginer que la table de transport est vide, c'est-à-dire initialement tout $x_{ij} = 0$ . Les procédures les plus simples pour l'allocation initiale seront discutées dans la section suivante.[4]

### 3.1.1 Solution de base réalisable

**Définition**

On appelle solution de base réalisable d'un programme de transport, une solution admissible comportant $M = (m + n - 1)$ $x_{ij} > 0$, c'est-à-dire qu'une solution de base comporte $(m.n - M)$ zéros.

Le graphe d'une solution de base est un graphe connexe sans cycle, c'est-à-dire un arbre comportant $N = m + n$ sommets soit $M = N - 1$ arcs.

Si le nombre d'allocations dans les solutions de bases réalisables est inférieur à $(m + n - 1)$, on appelle une solution de base dégénérée.

**Interprétation en termes de théories de graphes de la notion de solution de base**

Imaginons que nous disposons d'un algorithme qui permet à toute étape de satisfaire une demande ou épuiser une disponibilité.

À chaque étape de l'algorithme $\varphi_{ij} = \min\{a', b'\}$ $(a'_i, b'_j \neq 0, \varphi_{ij} \in \mathbb{N})$ avec :
$a'$ = disponibilité restante dans $x_i$.
$b'$ = demande insatisfaisante dans $y_j$.

Si $a'_i \neq b'_j$ (sauf la dernière étape où on a : $a'_i = b'_j$ ) nous aurons choisi $(m = n - 1)$ flux $\varphi_{ij}$ non nuls et nous obtenons une solution de base :
$(m \times n - (m + n - 1) = (m - 1)(n - 1))$ flux nuls.

### 3.1.2 Solution optimale

Une solution réalisable (pas nécessairement de base) est considérée optimale si elle minimise le coût total du transport.

### 3.1.3 Organigramme de résolution pour le problème de transport

On résume la résolution de problème de transport sous forme d'organigramme suivant :

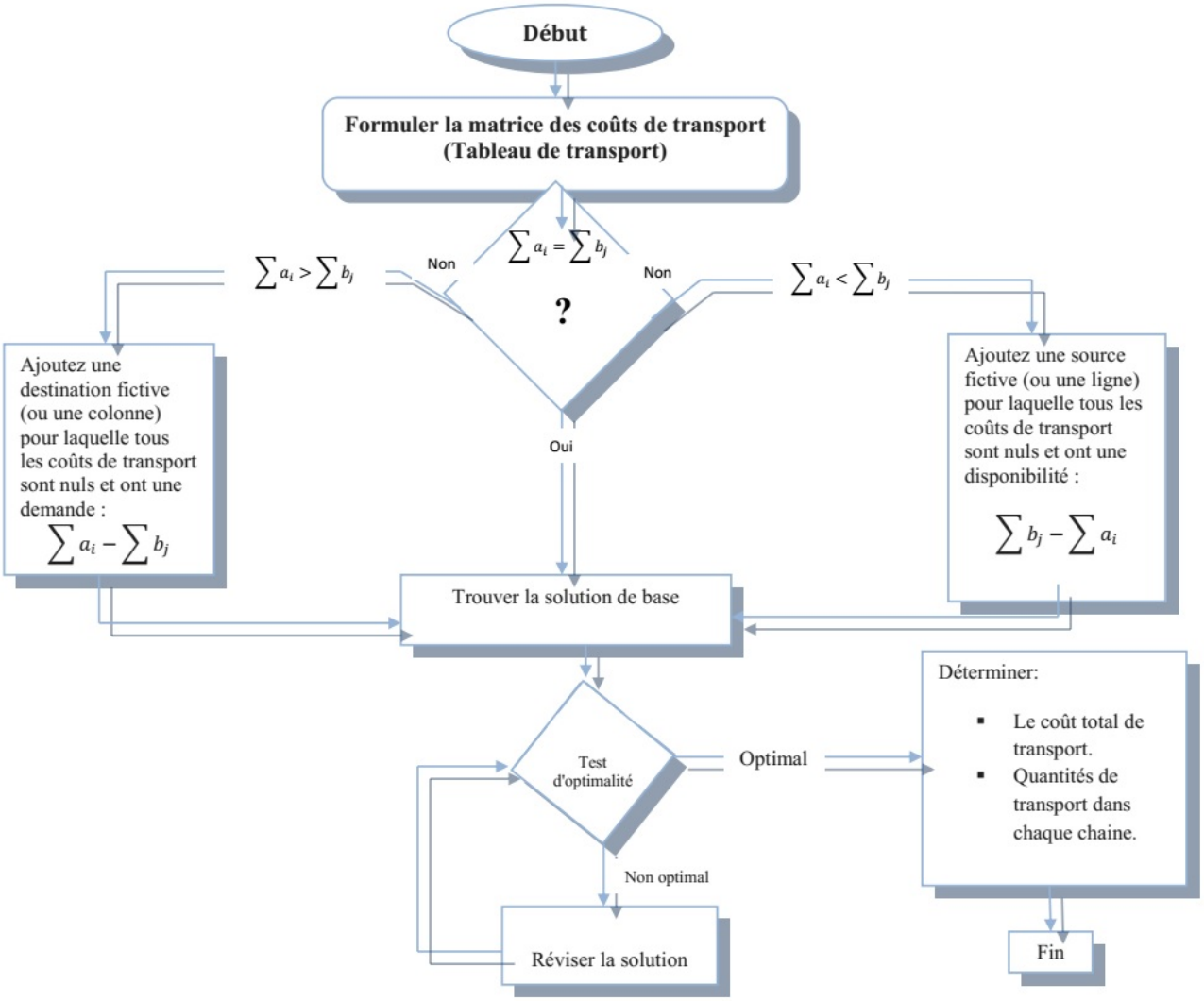



FIGURE 3.1 – Organigramme de la résolution de problème de transport

**Description sommaire d'organigramme de résolution :**

1. Tout d'abord, le problème est formulé comme un tableau de transport.
2. Vérifiez si un modèle de transport est équilibré ?
3. Sinon, ajoutez une source fictive ou une destination fictive pour équilibrer le modèle de transport.
4. Trouvez la solution de base initiale du problème de transport.
5. Vérifiez si la solution est optimale ? Si la solution n'est pas optimale, revenir à 4.

6. Lorsque la solution optimale est obtenue, nous calculons le coût total du transport et nous avons également transporté la quantité respective demandée à son destinataire.

### 3.1.4 Algorithme général de résolution de problème de transport

Les modèles de transport ne commencent pas à l'origine où toutes les valeurs de décision sont nulles, Ils doivent plutôt recevoir une solution de base réalisable initiale.

L'algorithme de résolution à un problème de transport peut se résumer en étapes suivantes :

**Étape 1 :** Formuler et configurer le problème sous la forme matricielle : La formulation du problème de transport est similaire à la formulation du problème PL. Ici, la fonction objective est le coût total du transport et les contraintes sont l'offre et la demande disponibles à chaque source et destination, respectivement.

**Étape 2 :** Obtenir une première solution de base réalisable : Cette solution de base initiale peut être obtenue en utilisant l'une des méthodes suivantes :
- Méthode de Coin Nord-Ouest.
- Méthode du Coût Minimum.
- Méthode d'Approximation de Vogel.

La solution obtenue par l'une des méthodes ci-dessus doit satisfaire les conditions suivantes :

1. La solution doit être réalisable, c'est-à-dire qu'elle doit satisfaire toutes les contraintes de l'offre et de la demande.
2. Le nombre d'attribution positive (les cases allouées) doit être égal à m + n - 1, où m est le nombre de lignes et n est le nombre de colonnes.

La solution qui satisfait les conditions mentionnées ci-dessus est appelée une solution de base non dégénérée.

**Étape 3 :** Tester la solution de base initiale pour l'optimalité : L'utilisation de l'une des méthodes suivantes pour tester l'optimalité de la solution de base initiale obtenue :
- Méthode Stepping Stone.
- Méthode de distribution modifiée.
- Méthode de simplexe adapté .

**Si** la solution est optimale arrêtez, **sinon** déterminez une nouvelle solution améliorée.

**Étape 4 :** Mise à jour de la solution. Répétez l'étape 3 jusqu'à atteindre la solution optimale.

## 3.2 Méthode de détermination de la solution de base

### 3.2.1 Méthode du COIN NORD-OUEST

**Présentation**

La méthode du coin nord-ouest est une méthode facile mais elle n'a pas de sens économique. Puisqu'elle consiste à affecter au coin nord-ouest de chaque grille la quantité maximum possible sans se préoccuper de l'importance du coût.[7]

**Principe**

**Étape 1 :** Localiser la cellule $(p; q)$ qui se trouve dans le coin nord-ouest, c'est à-dire en haut à gauche, de la partie non-éliminée du tableau de transport.

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

$$a'_p = a_p - x_{pq}$$

$$b'_p = b_p - x_{pq}$$

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

**Étape 3 :**

1. Si $a'_p = 0$ et $b'_q > 0$, cela signifie que l'origine $p$ a été "vidée". Il faut donc éliminer la ligne p du tableau.
2. Si $b'_q = 0$ et $a'_p > 0$, cela signifie que la destination q est entièrement satisfaite et qu'il reste des marchandises dans le dépôt $p$. Il faut donc éliminer la colonne $q$ du tableau.
3. Si $a'_p = 0$ et $b'_q = 0$, nous nous trouvons dans un cas dégénéré. On élimine alors la ligne $p$, à moins qu'elle ne soit la seule ligne restante du tableau ; auquel cas il faut éliminer la colonne $q$.

**Étape 4 :**

1. S'il reste un total de deux ou plusieurs lignes et colonnes non encore éliminées, reprendre à l'étape 1.
2. S'il ne reste qu'une ligne non éliminée, la solution réalisable de base initiale est déterminée par les cellules entourées.

• Appliquons la méthode de Coin nord-ouest au problème de transport donné par la table 3.1.

| Origines \ Destinations | 1 | 2 | 3 | Offre |
|---|---|---|---|---|
| 1 | 25 | 17 | 16 | 350 |
| 2 | 24 | 18 | 14 | 550 |
| Demande | 300 | 300 | 300 | 900 |

TABLE 3.1 – Problème de transport initial

**Itération 1 :**

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

**Étape 2**

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

**Étape 3** $\acute{a}_1 = 50$ et $\acute{b}_1 = 0$ , on élimine donc la colonne 1.

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

| Origines \ Destinations | ~~1~~ | 2 | 3 | Offre |
|---|---|---|---|---|
| 1 | ~~25~~ 300 | 17 | 16 | 350 50 |
| 2 | ~~24~~ | 18 | 14 | 550 |
| Demande | ~~300~~ 0 | 300 | 300 | 900 |

**Itération 2 :**

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

**Étape 2**

$$x_{12} = \min\{a_1; b_2\} = \min\{50; 300\} = 50$$

$$a'_1 = a_1 - x_{12} = 50 - 50 = 0$$

$$b'_2 = b_2 - x_{12} = 300 - 50 = 250$$

**Étape 3** $a'_1 = 50$ et $b'_2 = 250$ , on élimine donc la ligne 1.

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

| Origines \ Destinations | 1 | 2 | 3 | Offre |
|---|---|---|---|---|
| 1 | 25 300 | 17 50 | 16 | 350 0 |
| 2 | 24 | 18 | 14 | 550 |
| Demande | 300 0 | 300 250 | 300 | 900 |

**Itération 3 :**

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

**Étape 2**

$$x_{22} = \min\{a_2; b_2\} = \min\{550; 250\} = 250$$

$$a'_2 = a_2 - x_{22} = 550 - 250 = 300$$

$$b'_2 = b_2 - x_{22} = 250 - 250 = 0$$

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

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

| Origines \ Destinations | 1 | 2 | 3 | Offre |
|---|---|---|---|---|
| 1 | 25<br>300 | 17<br>50 | 16 | 350<br>0 |
| 2 | 24 | 18<br>250 | 14 | 550<br>300 |
| Demande | 300<br>0 | 300<br>0 | 300 | 900 |

**Itération 4 :**

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

**Étape 2**

$$x_{23} = \min\{a_2; b_3\} = \min\{300; 300\} = 300$$
$$a'_2 = a_3 - x_{23} = 300 - 300 = 0$$
$$b'_2 = b_3 - x_{23} = 300 - 300 = 0$$

**Étape 3** $a'_2 = 300$ et $b'_3 = 0$ , la ligne 2 est la dernière ligne, on élimine donc la colonne 3.

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

| Origines \ Destinations | 1 | 2 | 3 | Offre |
|---|---|---|---|---|
| 1 | 25<br>300 | 17<br>50 | 16 | 350<br>0 |
| 2 | 24 | 18<br>250 | 14<br>300 | 550<br>0 |
| Demande | 300<br>0 | 300<br>0 | 300<br>0 | 900 |

Nous pouvons calculer le coût de transport total

$$z = \sum_{j=1}^{n}\sum_{i=1}^{m} c_{ij}x_{ij} = c_{11}x_{11}+c_{12}x_{12}+c_{22}x_{22}+c_{23}x_{23} = 25(300)+17(50)+18(250)+14(300) = 17050$$

### 3.2.2 Méthode de coût minimum

**Définition**

La méthode du Coût Minimum est une méthode pour calculer une solution de base réalisable d'un problème de transport où les variables de base sont choisies en fonction du coût unitaire du transport. La méthode du coût minimum trouve une meilleure solution de départ en se concentrant sur les coûts de transport les moins chers.[7]

**Principe**

cette méthode ne diffère de la précédente que par le critère appliqué à l'étape (1), exposée ici , les étapes (2), (3) et (4) restant les mêmes.

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

- La méthode commence par affecter autant que possible à la case avec le coût unitaire de transport le plus petit. Ensuite, la ligne où la colonne satisfaite est dépassée et les montants de l'offre et de la demande sont ajustés en conséquence. Si à la fois une ligne et une colonne sont satisfaites simultanément, une seule est décalée, la même que dans la méthode du Coin Nord-Ouest, Ensuite, recherchez la case non décalée avec le coût unitaire le plus petit et répétez le processus jusqu'à ce qu'une ligne où une colonne exactement soit laissée hors traitement.

• Appliquons la méthode de matrice minimale au problème de la table 3.1

**Itération 1 :**

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

**Étape 2** Calculons $x_{23},\ b'_2\ \ et\ \ b'_3$.

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

**Étape 3** Comme $a'_2 = 250$ et $b'_3 = 0$ , il faut éliminer la colonne 3.

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

le tableau après cette première itération.

| Origines \ Destinations | 1 | 2 | 3 | Offre |
|---|---|---|---|---|
| 1 | 25 | 17 | 16 | 350 |
| 2 | 24 | 18 | 14<br>300 | 550<br>250 |
| Demande | 300 | 300 | 300<br>0 | 900 |

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

**Itération 2 :**

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

**Étape 2**

$$x_{12} = \min\{a_1; b_2\} = \min\{350; 300\} = 300$$
$$a'_1 = a_1 - x_{12} = 350 - 300 = 50$$
$$b'_2 = b_2 - x_{12} = 300 - 300 = 0$$

**Étape 3** $a'_1 = 50$ et $b'_2 = 0$, il faut donc éliminer la colonne 2.

**Étape 4** Comme il reste deux lignes et une colonne, nous n'avons pas terminé, nous cherchons alors le troisième $x_{pq}$ à entrer dans la base.

le tableau après cette deuxième itération.

| Origines \ Destinations | 1 | 2 | 3 | Offre |
|---|---|---|---|---|
| 1 | 25 | 17<br>300 | 16 | 350<br>50 |
| 2 | 24 | 18 | 14<br>300 | 550<br>250 |
| Demande | 300 | 300<br>0 | 300<br>0 | 900 |

**Itération 3 :**

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

**Étape 2**

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

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

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

le tableau après cette troisième itération.

| Origines \ Destinations | 1 | 2 | 3 | Offre |
|---|---|---|---|---|
| 1 | 25 | 17<br>300 | 16 | 350<br>50 |
| 2 | 24<br>250 | 18 | 14<br>300 | 550<br>0 |
| Demande | 300<br>50 | 300<br>0 | 300<br>0 | 900 |

**Itération 4 :**

**Étape 1** Le seul $x_{pq}$ qui peut encore entrer dans la base est $x_{11}$ .

**Étape 2**

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

**Étape 3** $a'_1 = 0$ et $b'_1 = 0$ , la ligne 1 est la dernière des lignes, on élimine donc la colonne 1.

**Étape 4** Il ne reste que la ligne 1, nous avons donc terminé !

| Origines \ Destinations | 1 | 2 | 3 | Offre |
|---|---|---|---|---|
| 1 | 25<br>50 | 17<br>300 | 16 | 350<br>0 |
| 2 | 24<br>250 | 18 | 14<br>300 | 550<br>0 |
| Demande | 300<br>0 | 300<br>0 | 300<br>0 | 900 |

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

$$z = \sum_{j=1}^{n}\sum_{i=1}^{m} c_{ij}x_{ij} = c_{11}x_{11}+c_{12}x_{12}+c_{21}x_{21}+c_{23}x_{23} = 25(50)+17(300)+24(250)+14(300) = 16550$$

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

### 3.2.3 Méthode d'Approximation de Vogel

#### Présentation

Cette méthode est basée sur le calcul des regrets. Le regret associé à une ligne ou à une colonne est la différence entre le coût minimum et le coût immédiatement supérieur dans cette ligne ou dans cette colonne. C'est une mesure de la priorité à accorder aux transports de cette ligne ou de cette colonne, car un regret important correspond à une pénalisation importante si on n'utilise pas la route de coût minimum.[12]

La méthode de Vogel fournit, en général, une solution très proche de l'optimum ; le nombre de changements de base nécessaires pour arriver à une solution optimale est peu élevé (il arrive même assez fréquemment que la solution donnée par cette règle soit optimale).

#### Principe

D'abord, on calcule pour chaque rangée, ligne ou colonne, la différence entre le coût le plus petit avec celui qui lui est immédiatement supérieur.

Ensuite on affecte à la relation de coût le plus petit correspondant à la rangée présentant la différence maximum la quantité la plus élevée possible. Ce qui sature une ligne ou une colonne.

Et on reprendre le processus jusqu'à ce que toutes les rangées soient saturées.

**Remarques :**

- La Méthode d'Approximation de Vogel (VAM) est connue aussi par :l'heuristique de Balas-Hammer /la Méthode de la différence maximum / la méthode de pénalité unitaire /la méthode des regrets maximaux successifs.
- La méthode VAM est basée sur la notion de coût de pénalité où de regret.
- Un coût de pénalité est la différence entre le coût de case le plus grand et le plus important d'une rangée (ligne où colonne).
- Dans VAM, la première étape consiste à développer un coût de pénalité pour chaque source et destination.
- Le coût de pénalité est calculé en soustrayant le coût de case minimum du coût de case supérieur suivant dans chaque rangée.

### L'algorithme d'Approximation de Vogel

$\Delta_l$ représente la différence entre le coût minimum et celui immédiatement supérieur sur une ligne. $\Delta_c$ représente la différence entre le coût minimum et celui immédiatement supérieur sur une colonne.

1. Calculer les différences $\Delta_l$ et $\Delta_c$ pour chaque ligne et colonne.
2. Sélectionner la ligne ou la colonne ayant le $\Delta_l$ ou $\Delta_c$ maximum.
3. Choisir dans cette ligne ou colonne le coût le plus faible.
4. Attribuer à la relation $(i, j)$ correspondante le maximum possible de matière transportable de façon à saturer soit la destination soit la disponibilité.
5. calculer la quantité résiduelle soit demande soit en disponibilité.
6. Eliminer la ligne ou la colonne ayant sa disponibilité ou demande satisfaite.
7. SI nombre de lignes ou colonnes $> 2$ retour en 2. SINON affecter les quantités restantes aux liaisons.

• Appliquons la méthode de d'approximation de Vogel au problème de la table 3.1

**Itération 1 :**

**Étape 1**

- le différence entre le coût de la ligne 1 $\quad \Delta_l = 17 - 16 = 1$
- le différence entre le coût de la ligne 2 $\quad \Delta_l = 18 - 14 = 4$
- le différence entre le coût de la colonne 1 $\quad \Delta_c = 25 - 24 = 1$
- le différence entre le coût de la colonne 2 $\quad \Delta_c = 18 - 17 = 1$
- le différence entre le coût de la colonne 3 $\quad \Delta_c = 16 - 14 = 2$

**Étape 2**

$$max\{\Delta_{c1}, \Delta_{c2}, \Delta_{c3}\} = max\{1, 1, 2\} = 2$$

on sélection la colonne 3.

**Étape 3** Le coût le plus faible da la colonne 3 $\min\{16, 14\} = 14$

donc le coût le plus faible 14 de la ligne 2.

**Étape 4**

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

**Étape 5** $b'_3 = 0$ , on élimine donc la colonne 3

**Étape 6** Il reste un ligne et deux colonne, nous n'avons pas terminé, on révient à l'etape 1.

| Destinations / Origines | 1 | 2 | ~~3~~ | Offre | $\Delta_l$ |
|---|---|---|---|---|---|
| 1 | 25 | 17 | ~~16~~ | 350 | 1 |
| 2 | 24 | 18 | ~~14~~ 300 | 550 250 | 2 |
| Demande | 300 | 300 | ~~300~~ 0 | 900 | |
| $\Delta_c$ | 1 | 1 | 2 | | |

**Itération 2 :**

**Étape 1**

- le différence entre le coût de la ligne 1 $\Delta_l = 25 - 17 = 8$
- le différence entre le coût de la ligne 2 $\Delta_l = 24 - 18 = 6$
- le différence entre le coût de la colonne 1 et 2 il reste le même.

**Étape 2**

$$max\{\Delta_{c1}, \Delta_{c2}\} = max\{1, 1\} = 1$$

on sélection la colonne 1.

**Étape 3** Le coût le plus faible da la colonne 1 $\min\{25, 24\} = 24$

donc le coût le plus faible 24 de la ligne 2.

**Étape 4**

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

**Étape 5** $b'_2 = 0$ , on élimine donc la ligne 2

| Origines \ Destinations | 1 | 2 | 3 | Offre | $\Delta_l$ |
|---|---|---|---|---|---|
| 1 | 25 | 17 | 16 | 350 | 8 |
| 2 | 24<br>250 | 18 | 14<br>300 | 550<br>0 | 6 |
| Demande | 300<br>50 | 300 | 300<br>0 | 900 | |
| $\Delta_c$ | 1 | 1 | | | |

**Étape 6** Le nombre de ligne inférieure à 2 donc effecter les quantités restantes aux liaison $x_{11} = 50$ et $x_{12} = 300$ ,nous avons donc terminé.

| Origines \ Destinations | 1 | 2 | 3 | Offre |
|---|---|---|---|---|
| 1 | 25<br>50 | 17<br>300 | 16 | 350<br>0 |
| 2 | 24<br>250 | 18 | 14<br>300 | 550<br>0 |
| Demande | 300<br>0 | 300<br>0 | 300<br>0 | 900 |

Nous pouvons maintenant calculer le coût de transport total :

$$z = \sum_{j=1}^{n}\sum_{i=1}^{m} c_{ij}x_{ij} = c_{11}x_{11}+c_{12}x_{12}+c_{21}x_{21}+c_{23}x_{23} = 25(50)+17(300)+24(250)+14(300) = 16550$$

On obtient par hasard la même solution réalisable de base initiale qu'avec la deuxième méthode proposée. Ceci est à considérer comme une exception due aux dimensions réduites du problème choisi comme exemple.

## 3.3 Méthode d'optimisation de la solution de base

### 3.3.1 Méthode de Stepping-Stone

L'algorithme du Stepping-Stone sera un algorithme itératif (donc par étapes successives) visant à améliorer (donc faire baisser le coût global) une solution de base.[7]

Il nous faut donc une solution de départ pour démarrer l'algorithme.

**Déroulement de l'algorithme**

L'algorithme consiste à modifier la solution pour une qui soit meilleure, donc à rendre non vide une case vide du tableau des quantités.

On appelle :
$T_i \ll u_i \gg$ : potentiel origine.
$T_j \ll v_j \gg$ : potentiel destination.
$\partial_{ij}$ : coût marginal de la liaison $(x_i, x_j)$.

**Algorithme de stepping stone**

1. Déterminer une solution de base initiale.
2. Calculer les coûts marginaux $\delta_{ij} = c_{ij} - (t_j t_i)$ avec $t_j - t_i = \theta$, $I_j$ la tension de l'arc $(i, j)$, ti s'appelle le potentiel du sommet $i$ de l'arc $(i, j)$. Si tous les coûts marginaux sont positifs où nuls alors FIN. La solution est optimal, sinon passer à 3.
3. Pour tous les coûts marginaux négatifs, chercher la chaine de substitution et déterminer la quantité maximum qui peut être déplacé et passer à (4).Alors le gain correspondant est égale au produit de cette quantité par le coût marginale.
4. Retenir la chaine de substitution qui réalise la plus grande diminution du coût de transport, l'effectuer et revenir à (2).

**Remarque :** Le coût marginale : la quantité (positive ou négative) qui s'ajout au coût globale lorsqu'on veut transporter une unité sur un arc de flux nul.

**Comment déterminer les potentiels ?**

- On utilisera le tableau des coûts limité aux cases où la quantité transitée est non nulle.
- On déterminera les potentiels de proche en proche : on commencera par une destination, puis une origine, puis une destination.

**Comment déterminer les $\delta$ ?**

Pour chaque case nulle, on calculera $\delta$ en ajoutant au coût unitaire de la case le potentiel d'origine associée et en retranchant le potentiel de la destination correspondante.

**Comment déterminer les quantités à transporter$(q)$ ?**

- On déterminera les quantités qu'on peut ajouter aux cases vides uniquement pour celles dont le $\delta$ est négatif, il ne sert à rien, en effet, de remplir une case qui fait augmenter le coût.

– Pour remplir une case vide, il faut diminuer une case pleine, donc constituer un circuit de cases pleines qu'on vide et remplit alternativement.

**Résumé d'heuristique de Stepping-Stone**

1. Déterminez les chemins d'accès et les changements de coûts pour chaque case vide dans le tableau.
2. Allouer autant que possible à la case vide avec la plus forte diminution nette des coûts.
3. Répétez l'étape 1 et 2 jusqu'à ce que toutes les cases vides aient des changements de coûts positifs qui indiquent une solution optimale.

• Appliquons la méthode de Stepping-Stone au problème de la table 3.1

**Étape 1** Tableau de transport initial (table 3.1).

**Étape 2** À l'aide de la méthode du coin nord-ouest, nous avions obtenu la solution réalisable de base initiale suivante :

| Origines \ Destinations | 1 | 2 | 3 | Offre |
|---|---|---|---|---|
| 1 | 25 * 300 | 17 * 50 | 16 (3) | 350 |
| 2 | 24 (-2) | 18 * 250 | 14 * 300 | 550 |
| Demande | 300 | 300 | 300 | 900 |

**Étape 3** Calculons maintenant les valeurs de boucles pour les cellules hors de la base $(1;3)$ et $(2;1)$.

$$v(1;3) = c_{13} - c_{23} + c_{22} - c_{12} = 16 - 14 + 18 - 17 = 3$$

.

$$v(2;1) = c_{21} - c_{11} + c_{12} - c_{22} = 24 - 25 + 17 - 18 = -2$$

. Ces valeurs sont du reste reportées dans le tableau ci-dessus, à l'endroit réservé aux $x_{ij}$. C'est ainsi que l'on procédera lors de tout calcul fait à la main. On remarque que $v(2;1) < 0$ ; par conséquent, la solution actuelle n'est pas optimale.

**Étape 4** On peut améliorer la solution en faisant entrer $(2;1)$ dans la base. Opérons donc le changement de base tel qu'il est décrit dans la méthode d'entrée.

1.

$$x'_{21} = \min\{x_{11}, x_{22}\} = \min\{300, 250\} = 250$$

2.

$$x'_{12} = x_{12} - x'_{21} = 50 + 250 = 300$$
$$x'_{11} = x_{11} - x'_{21} = 300 - 250 = 50$$
$$x'_{22} = x_{22} - x'_{21} = 250 - 250 = 0$$

3. La cellule $(2;1)$ entre dans la base ; la cellule $(2;2)$ en sort.

Voici le tableau obtenu après le changement de base :

| Origines \ Destinations | 1 | 2 | 3 | Offre |
|---|---|---|---|---|
| 1 | 25 * 50 | 17 * 300 | 16 (1) | 350 |
| 2 | 24 * 250 | 18 (2) | 14 * 300 | 550 |
| Demande | 300 | 300 | 300 | 900 |

Le calcul des valeurs de boucle pour $(1;3)$ et $(2;2)$ nous a permis d'obtenir les résultats reportés dans le tableau . On voit que $v(1;3) = 1$et $v(2;2) = 2$. Comme toutes deux sont non-négatives, la solution actuelle est la solution réalisable de base optimale du problème. Une remarque concernant la boucle $b(1;3)$s'impose. La cellule $(1;2)$ ne fait en aucun cas partie de cette boucle, par conséquent, il n'y a pas trois cellules consécutives sur la ligne 1, contrairement à ce que l'on pourrait croire. Lors de la recherche d'une boucle, on peut ignorer toute cellule de base unique sur une ligne ou une colonne.

Revenons à notre solution ; nous remarquons qu'elle est la même que la solution réalisable de base initiale trouvée, soit par la méthode coût minimum, soit par la méthode Vogel, pour lesquelles $z = 16550$. Ceci montre, sur ce petit problème, que ces deux méthodes sont plus efficaces que celle du coin nord-ouest. Il est bien clair que pour des problèmes plus grands que celui envisagé ici, ni la méthode de la matrice minimale, ni la méthode Vogel ne donnent directement la solution réalisable de base optimale.

### 3.3.2 Méthode de Distribution Modifiée

#### Définition

La Méthode de distribution modifiée (où des penalties) : est une version modifiée de la méthode de stepping stone dans laquelle les équations mathématiques

remplacent les chaines de substitutions. Cette méthode est plus pratique que stepping stone.

En appliquant la méthode MODI, nous commençons par une solution initiale obtenue en utilisant les méthodes citées à la section précédente. Ensuite, nous devons calculer une valeur $u_i$ pour chaque ligne $i$ et $v_j$ pour chaque colonne $j$ dans la table de transport.[4]

**Les étapes de la méthode de distribution modifiée**

1. Pour calculer les valeurs $u_i$ et $v_j$ pour chaque ligne et chaque colonne, réduire les équations : $u_i + v_j = c_{ij}$.
2. Une fois que toutes les équations ont été écrites, définissez l'une des deux variables $u_i$ ou $v_j$ à zéro, et résolvez le système d'équations pour toutes les valeurs $u_i$ et $v_j$.
3. Calculez l'indice d'amélioration $\Delta_{ij}$ pour chaque cellule inutilisé par l'amélioration de la formule :$\Delta_{ij} = c_{ij} - u_i - v_j$.
4. Transférer la plus grande quantité possible à la cellule qui a $\Delta_{ij}$ le plus négatif en créant un cycle qui satisfait la demande et la disponibilité de chaque rangé.
5. Répétez les étapes 2 à 4 jusqu'à ce qu'il n'y ait pas de $\Delta_{ij}$ négatif.
6. Calculez le coût total en multipliant chaque allocation ($x_{ij}$) par son spécifique coût ($c_{ij}$).

**Résumé des étapes de La méthode de distribution modifiée (MODI)**

1. Développer une solution initiale.
2. Calculez les valeurs ui et vj pour chaque ligne et chaque colonne.
3. Calculer l'indice d'amélioration $\Delta_{ij}$ , pour chaque case vide.
4. Affecter autant que possible à la case vide qui entraînera la plus forte diminution du coût ( $\Delta_{ij}$ le plus négatif). Répétez les étapes 2 à 4 jusqu'à ce que toutes les valeurs $\Delta_{ij}$, soient positives où nulles.

• Appliquons la méthode de distribution modifiée au problème suivant :

| | $v_j$ | $v_1$ | $v_2$ | $v_3$ | |
|---|---|---|---|---|---|
| $u_i$ | Destinations / Origines | 1 | 2 | 3 | Offre |
| $u_1$ | 1 | 25 300 | 17 50 | 16 | 350 |
| $u_2$ | 2 | 24 | 18 250 | 14 300 | 550 |
| | Demande | 300 | 300 | 300 | 900 |

**Itération 1 :**

**Étape 1** Calculer pour toutes les cellules allouées :
$u_i + v_j = c_{ij}$ : Coût de transport unitaire pour la case $ij$.

$x_{11} : u_1 + v_1 = 25$
$x_{12} : u_1 + v_2 = 17$
$x_{22} : u_2 + v_2 = 18$
$x_{23} : u_2 + v_3 = 14$

**Étape 2** On met donc $u_1 = 0$ et on obtient :

$v_1 = 25, \quad v_2 = 17, \quad u_2 = 1, \quad v_3 = 13.$

**Étape 3** Utilisez la formule des penalties pour évaluer toutes les cellules vides : $c_{ij} - u_i - v_j = \Delta_{ij}$

$x_{13} : \Delta_{13} = c_{13} - u_1 - v_3 = 16 - 0 - 13 = 3$
$x_{21} : \Delta_{21} = c_{21} - u_2 - v_1 = 24 - 1 - 25 = -2$

**Étape 4**

1.
$$x'_{21} = \min\{x_{11}, x_{22}\} = \min\{300, 250\} = 250$$

2.
$$\acute{x}_{12} = x_{12} - x'_{21} = 50 + 250 = 300$$
$$x'_{11} = x_{11} - x'_{21} = 300 - 250 = 50$$
$$x'_{22} = x_{22} - x'_{21} = 250 - 250 = 0$$

3. La cellule $(2; 1)$ entre dans la base ; la cellule $(2; 2)$ en sort.

Voici le tableau obtenu après le changement de base :

| | $v_j$ | $v_1$ | $v_2$ | $v_3$ | |
|---|---|---|---|---|---|
| $u_i$ | Destinations / Origines | 1 | 2 | 3 | Offre |
| $u_1$ | 1 | 25 50 | 17 300 | 16 | 350 |
| $u_2$ | 2 | 24 250 | 18 | 14 300 | 550 |
| | Demande | 300 | 300 | 300 | 900 |

**Itération 2 :**

**Étape 1**

$x_{11} : u_1 + v_1 = 25$
$x_{12} : u_1 + v_2 = 17$
$x_{21} : u_2 + v_1 = 24$
$x_{23} : u_2 + v_3 = 14$

**Étape 2** On met donc $u_1 = 0$ et on obtient :

$v_1 = 25, \quad v_2 = 17, \quad u_2 = -1, \quad v_3 = 15.$

**Étape 3**

$x_{13} : \Delta_{13} = c_{13} - u_1 - v_3 = 16 - 0 - 15 = 1$
$x_{22} : \Delta_{22} = c_{22} - u_2 - v_2 = 18 + 1 - 17 = 2$

Toutes les valeurs $\Delta_{ij}$ sont positives où nuls, donc la solution obtenue est optimale.

$$z = \sum_{j=1}^{n}\sum_{i=1}^{m} c_{ij}x_{ij} = c_{11}x_{11}+c_{12}x_{12}+c_{21}x_{21}+c_{23}x_{23} = 25(50)+17(300)+24(250)+14(300) = 16550$$

CHAPITRE 4

# RÉSOLUTION DU PROBLÈME D'AFFECTATION, ET DE FLOT MAXIMUM

Dans ce chapitre, nous allons aborder dans un premier temps le problème d'affectation avec l'algorithme de résolution hongrois, et ensuite nous allons présenter le problème de flot et plus précisément le problème de flot maximum à coût minimal.

## 4.1 Résolution du problème d'affectation

### 4.1.1 Algorithme hongrois

Soit $V = (v_{ij})_{1 \leq i,j \leq n}$ la matrice des coûts associée à un problème d'affectation de coût minimal .

**Phase 1 : Réduction initiale**

- Soustraire l'élément minimum de la ligne $i$ de chaque élément de la ligne $i$, pour tout $i = 1, ...., n$.
- Soustraire l'élément minimum de la colonne $j$ de chaque élément de la colonne $j$, pour tout $j = 1, ...., n$.

**Phase 2 : Marquage des zéros**

Prendre la ligne contenant le moins de zéros ; encadrer le premier zéro de cette ligne et barrer les autres zéros de la ligne et de la colonne du zéro encadré. Refaire cette opération jusqu'à impossibilité d'encadrer un zéro.

**Phase 3 : Recherche d'une solution optimale**

1. procédure de marquage des lignes et des colonnes :
   - marquer les lignes ne contenant aucun 0 encadré (s'il n'y en a pas : FIN) ;
   - marquer toute colonne qui a un 0 barré sur une ligne marquée ;
   - marquer toute ligne qui a un 0 encadré dans une colonne marquée ;
   - retour à b et c jusqu'à ce qu'il n'y ait plus de ligne ou de colonne à marquer.
2. rayer chaque ligne non marquée et chaque colonne marquée.
3. réduction : choisir le plus petit élément p du tableau non rayé, l'ajouter aux colonnes non rayées et le soustraire aux lignes rayées.
4. retour à phase 2.

• Soit le problème d'affectation donné par la table 4.1

| | | | |
|---|---|---|---|
| 7 | 2 | 1 | 9 |
| 9 | 6 | 9 | 5 |
| 8 | 8 | 3 | 1 |
| 7 | 9 | 4 | 2 |

TABLE 4.1 – Problème d'affectation

**Itération 1 :**

Phase 1 : réduction initiale

| | | | | min |
|---|---|---|---|---|
| 7 | 2 | 1 | 9 | 1 |
| 9 | 6 | 9 | 5 | 5 |
| 8 | 8 | 3 | 1 | 1 |
| 7 | 9 | 4 | 2 | 2 |

=Réduction des lignes⇒

| | | | |
|---|---|---|---|
| 6 | 1 | 0 | 8 |
| 4 | 1 | 4 | 0 |
| 7 | 7 | 2 | 0 |
| 5 | 7 | 2 | 0 |

| | | | | |
|---|---|---|---|---|
| | 6 | 1 | 0 | 8 |
| | 4 | 1 | 4 | 0 |
| | 7 | 7 | 2 | 0 |
| | 5 | 7 | 2 | 0 |
| min | 4 | 1 | 0 | 0 |

=Réduction des colonnes⇒

| | | | |
|---|---|---|---|
| 2 | 0 | 0 | 8 |
| 0 | 0 | 4 | 0 |
| 3 | 6 | 2 | 0 |
| 1 | 6 | 2 | 0 |

Phase 2 : marquage des zéros

| | | | |
|---|---|---|---|
| 2 | [0] | ~~0~~ | 8 |
| [0] | ~~0~~ | 4 | ~~0~~ |
| 3 | 6 | 2 | [0] |
| 1 | 6 | 2 | ~~0~~ |

Phase 3 : recherche d'une solution optimale

1) Marquage des lignes et colonnes

| | | | | |
|---|---|---|---|---|
| | | | X | |
| 2 | [0] | ~~0~~ | 8 | |
| [0] | ~~0~~ | 4 | ~~0~~ | |
| 3 | 6 | 2 | [0] | X |
| 1 | 6 | 2 | ~~0~~ | X |

2) Rayure des lignes et colonnes

| | | | | |
|---|---|---|---|---|
| | | | X | |
| ~~2~~ | ~~[0]~~ | ~~0~~ | ~~8~~ | |
| ~~[0]~~ | ~~0~~ | ~~4~~ | ~~0~~ | |
| 3 | 6 | 2 | [0] | X |
| 1 | 6 | 2 | ~~0~~ | X |

3) Réduction

| | | | |
|---|---|---|---|
| 2 | 0 | 0 | 9 |
| 0 | 0 | 4 | 1 |
| 2 | 5 | 1 | 0 |
| 0 | 5 | 1 | 0 |

**Itération 2 :**

Phase 2 : marquage des zéros

| | | | |
|---|---|---|---|
| 2 | ~~0~~ | [0] | 9 |
| ~~0~~ | [0] | 4 | 1 |
| 2 | 5 | 1 | [0] |
| [0] | 5 | 1 | ~~0~~ |

=Fin⇒

| | | | |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 |

Une affectation de coût minimal est : $1 + 6 + 1 + 7 = 15$

## 4.2 Résolution du Problème de flot

Nous présentons maintenant les algorithmes de flots, qui vont nous permettre de traduire de manière efficace le résultat théorique d'intégrité des flots. Nous commençons par le problème du flot de valeur maximum, car l'algorithme est plus simple à présenter dans ce cas. Nous cherchons donc un flot de valeur maximum $v$ d'une source $s$ à un puits $p$, qui soit admissible, c'est-à-dire qui vérifie la contrainte de capacité, puis nous présentons l'algorithme du flot maximum à coût minimum .[8, 5]

### 4.2.1 Algorithme de Ford Fulkerson

#### Présentation

L'algorithme de Ford Fulkerson permet de calculer un flot maximum entre un sommet source S et un sommet puit T. le principe et de chercher à chaque itération une chaine améliorant $\mu$ ou joignant S et T, si on trouve une tell chaine alors on calcule l'augmentation du flot d'une valeur $\varepsilon$ tel que

$$\varepsilon = \min\{\varepsilon_1, \varepsilon_2\}$$

avec

$$\varepsilon_1 = \min_{(i,j)\in\mu^+}\{\mu_{ij} - x_{ij}\}$$

$$\varepsilon_2 = \min_{(i,j)\in\mu^-}\{x_{ij}\}$$

En effet, on peut améliorante la valeur du flot de $\varepsilon$ unités sur les arcs avant de $\mu(\mu^+)$ et diminuer de $\varepsilon$ unités sur les arc arrières de $\mu(\mu^-)$ . L'algorithme se termine lorsque on arrive pas à trouver une chaine augmentante.

#### Procédure du marquage

pour la recherche d'une chaine améliorante, on utilisé la procédure de marquage suivant :

- Initialisation $S = \varnothing$, $\mu^+ = \mu^- = \varnothing$
- On marque le sommet s d'un " + ", $S = \{s\}$.
- On marque d'une " + " une sommet $j \notin S$ tel que

$$x_{ij} < \mu_{ij}, \quad i \in S, \quad \mu^+ = \mu^+ \cup (i,j), \quad S = S \cup \{j\}$$

.

- On marque d'un " − " un sommet $j \notin S$, tell que

$$x_{ij} > 0, \quad i \in S, \quad \mu^- = \mu^- \cup (i,j), \quad S = S \cup \{j\}$$

.

- On arrête cette procédure lorsqu'on marque le sommet puit T .

**Algorithme**

1. Initialisation : $k = 0$, partir d'un flot initial réalisable, par exemple $x_{ij} = 0$.
2. 0 l'itération k, soit $x^k$ un flot réalisable
   - trouver une chaine augmantante $\mu^k$ reliant s et t en utilisant la procédure de marquage .
   - S'il n'en n'existe pas, alors $x^k$ est un flot maximum ,arrêter l'algorithme. Sinon aller à (3).
3. Mise à jour du flot $x^{k+1}$

   soit $\varepsilon^k$ la capacité résiduelle de la chaine $\mu^k$ alors on pose

$$x_{ij}^{k+1} = \begin{cases} x_{ij}^k + \varepsilon^k & \text{si } (i,j) \in \mu^{k+} \\ x_{ij}^k - \varepsilon^k & \text{si } (i,j) \in \mu^{k-} \\ x_{ij}^k & \text{si } (i,j) \notin \mu^k \end{cases}$$

   $k = k + 1$, aller à (2).

   {la coupe minimale s est composée de tous les sommets marqués durant la dernière itération de l'algorithme .}

• On considére le réseau de la figure 4.1

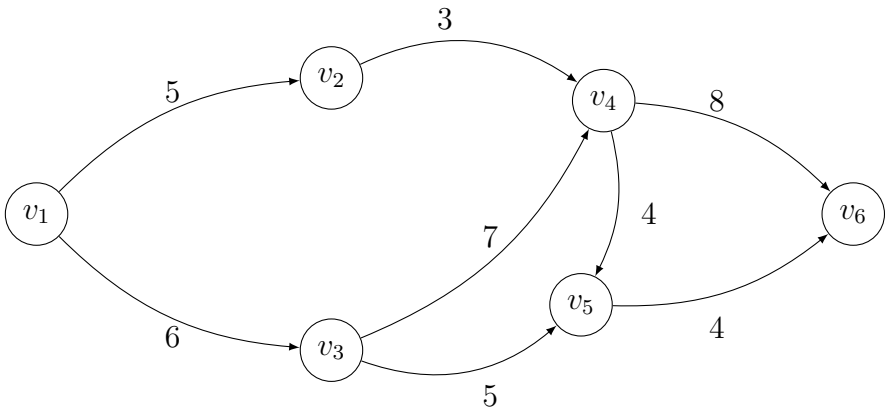



FIGURE 4.1 – Problème de folt maximum

**Initialisation**
$S = \varnothing$ , $\mu^+ = \mu^- = \varnothing$
On marque le sommet $s$ (entrée du réseau R) par le signe +, $S = \{s\}$

**Itération 1**
On pose : $\mu = \{v_1, v_2, v_4, v_6\}$
$\mu^+ = \{(v_1, v_2), (v_2, v_4), (v_4, v_6)\}$
$\mu^- = \varnothing$

$\varepsilon = \min\{(5-0),(3-0),(8-0)\} = \min\{5,3,8\} = 3$

On améliore ainsi le flot $x^0$ pour obtenir un nouveau flot $x^1 = 3$. Comme indiqué dans la figure 4.2

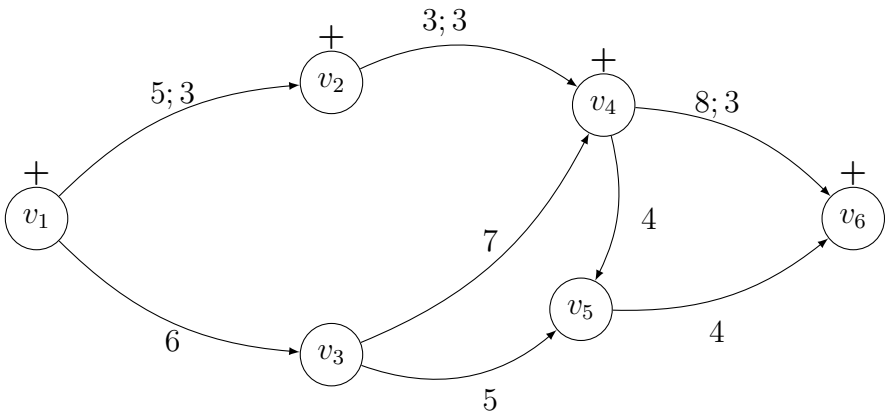



FIGURE 4.2 – Le réseau après avoir défini le nouveau flot $x^1$

**Itération 2**

On pose : $\mu = \{v_1, v_3, v_4, v_5, v_6\}$

$\mu^+ = \{(v_1, v_3), (v_3, v_4), (v_4, v_5), (v_5, v_6)\}$

$\mu^- = \varnothing$

$\varepsilon = \min\{(6-0),(7-0),(4-0),(4-0)\} = \min\{6,7,4,4\} = 4$

on améliore ainsi le flot $x^1$ pour obtenir un nouveau flot $x^2 = 3 + 4 = 7$, en ajoutant la quantité $\varepsilon$ au flot des arcs de $\mu$. Le flux des arcs n'appartenant pas à la chaine, reste inchangé. Comme indiqué dans la figure 4.3.

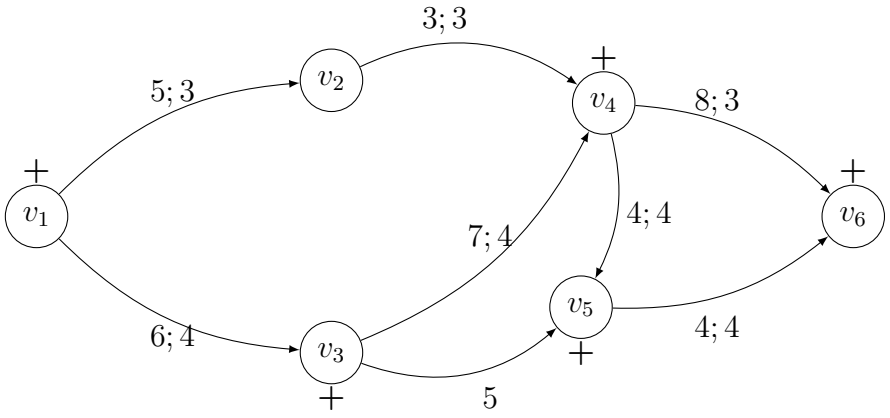



FIGURE 4.3 – Le réseau après avoir défini le nouveau flot $x^2$

**Itération 3**

On pose : $\mu = \{v_1, v_3, v_5, v_4, v_6\}$
$\mu^+ = \{(v_1, v_3), (v_3, v_5), (v_4, v_6)\}$
$\mu^- = \{(v_5, v_4)\}$
$\varepsilon^+ = \min\{(6-4), (5-0), (8-3)\} = \min\{2, 5, 5\} = 2$
$\varepsilon^- = \min\{4\} = 4$
$\varepsilon = \min\{\varepsilon^+, \varepsilon^-\} = \min\{4, 2\} = 2$

On améliore ainsi le flot $x^2$ pour obtenir un nouveau flot $x^3 = 7 + 2 = 9$, en ajoutant la quantité $\varepsilon^+$ au flot des arcs de $\mu$ et retranchant la quantité $\mu$ au flot des arcs $\varepsilon^-$. Le flux des arcs n'appartenant pas à la chaine, reste inchangé. Comme indiqué dans la figure 4.4.

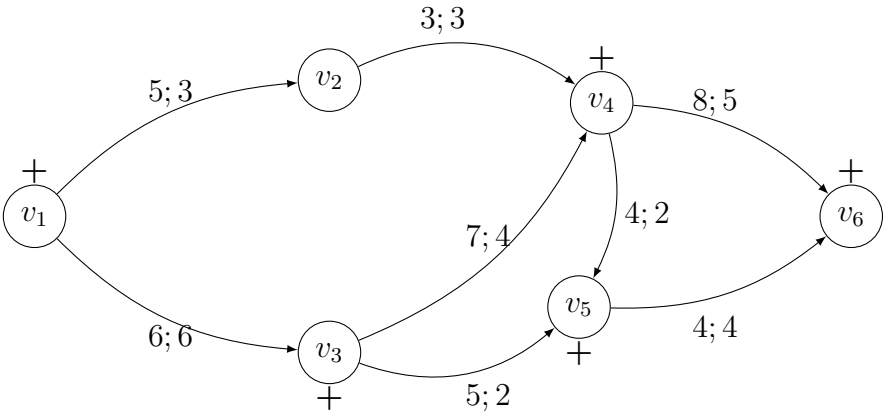



FIGURE 4.4 – Le réseau après avoir défini le nouveau flot $x^3$

**Itération 4**

Dans le réseau R, on ne peut pas marquer le sommet V . Donc le flot obtenu est maximum. Comme indiqué dans la figure 4.5.

$$x_{max} = 9$$

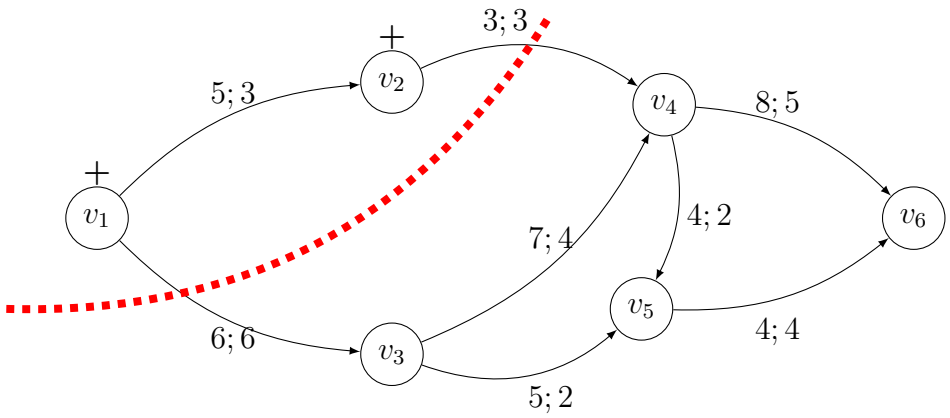



FIGURE 4.5 – Le réseau après avoir défini le flot maximum et la coupe minimal

la coupe minimale est forme des sommets marque d'un " + " ou d'un " − "
$S = \{v_1, v_2\}$ $\Gamma(S) = \{(v_1, v_3), (v_2, v_4)\}$ donc $C(S) = 6 + 3 = 9$

### 4.2.2 Méthode du simplexe réseau

**• Solution arbre**

Puisque $\sum_{i \in v} b_i = 0$ cela signifie que la matrice d'incidence $A$ du graphe $G$ a des lignes linéairement indépendantes si $G$ est un graphe orienté simple et connexe alors $rang(A) = n - 1$ et dans ce cas pour applique la méthode du simplexe, on doit supprimer une ligne de $A$ . [5]

Soit $A$ la matrice d'incidence d'une graphe orienté simple et connexe $G$ composé de $n$ sommet et de $m$ arcs .

Une sous-matrice carré $A_B$ de $A$ d'ordre $n-1$ est non singulière ($det \neq 0$) ssi les arcs associés aux colonnes de $A_B$ sont ceux d'un arbre partiel de $G$

$$G = (V, E), \quad |V| = n, \quad |E| = m$$

.

Un arbre T=(V,E) de $G$ est un graphe partiel de $G$ tel que :
– Les sommets de T sont n
– Les arcs de T sont $|E'| = n - 1$

Une solution de base réalisable pour le PL suivant

$$\begin{cases} \min \quad c'x & .....(1) \\ Ax = b \\ 0 \leq x \leq \mu \\ j = \{1, ...., n\} = j_B \cup j_N \end{cases}$$

Vérifie les contraintes tel que :

$$x_j = 0 \cup x_j, j \in j_N$$

et

$$0 \leq x_j \leq \mu_j, j \in j_B$$

(x,T) est un solution arbre du PFCM si :

$$\begin{cases} Ax = b \\ x_j = 0 \cup x_j \quad \forall (i,j) \notin T \end{cases}$$

Cela signifie que toute solution réalisable basique du PFCM correspond à un flot circulant sur les arcs d'un arbre et vice vers a

$$\begin{cases} \min \quad c'x \\ Ax = b \\ 0 \leq x \leq \mu \end{cases}$$

$$L(x, y, \delta_1, \delta_2) = c'x + y(Ax - b) + \delta_1' x - \delta_2(x - \mu)$$

$$\begin{cases} \frac{\partial L}{\partial x} = c' + y'x + \delta_1 x - \delta_2 & \\ Ax = b & 0 \leq x \leq \mu \\ \delta_{1j} x_j = 0 & \delta_{1j} \geq 0 \\ \delta_{2j}(x_j - \mu_j) = 0 & \delta_{2j} \geq 0 \\ \delta = \delta_2 - \delta_1 & \\ x_j = 0 & \delta_j \leq 0, 0 < x_j < \mu_j \\ x_j = \mu_j & \delta_j \geq 0 \\ 0 < x_j < \mu_j & \delta_j = 0, j \in j_N \end{cases}$$

• Calcul du vecteur des coût réduits

$$\delta \in \mathbb{R}, \delta = C - A'y$$

ou $y \in \mathbb{R}^n$ est le vecteur des multiplicateurs du simplexe associé aux sommets de G .

pour chaque arc $(i,j) \in E$ on a :

$$\delta_{ij} = C_{ij} - y_i + y_j$$

pour les variables de base on a :

$$\delta_{ij} = C_{ij} - y_i + y_j, \forall (i,j) \in T$$

alors on obtient un système de $(n-1)$ équations avec $n$ inconnues .

pour calculer les multiplicateur $y_i$, il suffit de fixer la valeur d'un multiplicateur (par exemple $y_t = 0$) et de calculer séquentielle-ment les autres inconnus .

**Critère d'optimalité**

soit x une solution réalisable basique du problème PFCM et T l'arbre associé . on a :

$$\begin{cases} \delta_{ij} \geq 0 & \text{pour } x_{ij} = 0 \\ \delta_{ij} \leq 0 & \text{pour } x_{ij} = \mu_{ij}, (i,j) \notin T \ .....(*) \end{cases}$$

alors x est une solution optimale du PFCM .

• Choix de la variable entrant

Dans ce cas ou les relation $\varphi$ ne sont pas vérifiées, alors on choisit une variable $x_{i_1 j_1} = 0$, alors sa valeur augment et diminue si :

$$x_{i_1 j_1} = \mu_{i_1 j_1} = 0$$

Une variable entre en base si on a :

$$\begin{cases} x_{ij} = 0 \quad et\ \delta_{ij} < 0 & \Longrightarrow \text{on augmante le flot } x_{ij} \text{ sur l'arc } (i,j) \\ x_{ij} = 0 \mu_{ij} \quad et\ \delta_{ij} > 0 & \Longrightarrow \text{on baisse le flot } x_{ij} \text{ sur l'arc } (i,j) \end{cases}$$

Calculons :

$$\begin{cases} \varepsilon_1 = \min\limits_{(i,j) \notin T} \{ \frac{\delta_{ij}}{x_{ij}=0 \ et \ \delta_{ij}<0} \} \\ \varepsilon_2 = \max\limits_{(i,j) \notin T} \{ \frac{\delta_{ij}}{x_{ij}=\mu_{ij} \ et \ \delta_{ij}>0} \} \end{cases}$$

soit $\varepsilon = \min\{\varepsilon_1, \varepsilon_2\}$ et on a 2 cas :

– Si $\varepsilon = \varepsilon_1 = \delta_{i_1 j_1} < 0$ , alors $x_{i_1 j_1}$ doit augmenter .
– Si $\varepsilon = \varepsilon_2 = \delta_{i_1 j_1} > 0$ , alors $x_{i_1 j_1}$ doit diminue .

• Choix de la variable sortante

L'ajoute d'un arc $(i_1, j_1)$ à l'arbre T crée un cycle élementaire $\varphi$, ou le flot sur les arcs de $\varphi$ change et pour les autres arcs ne changent pas .

– Si on a $x_{i_1 j_1} = 0$, alors on doit augmenter la valeur du flot des arcs de $\varphi$ ayant le meme sense que $(x_{i_1 j_1})$ noté $\varphi^+$ de $\theta$ unités et le diminue de $\theta$ unités pour les arcs ayant un sens de parcours opposé noté $\varphi^-$.
– Par contre, si $x_{i_1 j_1} = \mu_{i_1 j_1}$ alors on fait l'inverse, on diminue la quantité du flot de $\theta$ unités pour les arcs de $\varphi^+$ et on l'augmente pour les arcs de $\varphi^-$.

Le changement maximum de la valeur du flot noté $\theta$, est calculée comme suit :

$$\theta = \min\{\theta^-, \theta^+\}$$

avec ,

$$\begin{cases} \theta^- = \min\limits_{(i,j)\in\varphi^-} \{x_{ij}\} \\ \theta^+ = \min\limits_{(i,j)\in\varphi^+} \{\mu_{ij} - x_{ij}\} \end{cases}$$

La valeur du nouveau flot :

$$\begin{cases} x_{ij} + \theta & \text{Si} \quad (i,j) \in \varphi^+ \\ x_{ij} - \theta & \text{Si} \quad (i,j) \in \varphi^- \\ x_{ij} & \text{Sinon} \end{cases}$$

• Changement de base

En envoyant une quantité supplémentaire de flot de $\theta$ unités autour du cycle $\varphi$, alors il y a une variable basique notée $x_(i_0, j_0)$ qui atteindra l'une de ses bornes (0 ou $\mu_(i_0, j_0)$ ) par conséquent, l'arc $(i_0, j_0)$ est supprimé de la base (arbre T ) selon les cas suivant :

– Si $x_(i_0, j_0) \in T$, alors on pose $T = T \cup (i_0, j_0)\ (i_1, j_1)$.
– Si l'arc $(i_0, j_0) = (i_1, j_1) \notin T$, alors on pose $T = T$.

• considérons le réseau de la figure 4.6 :

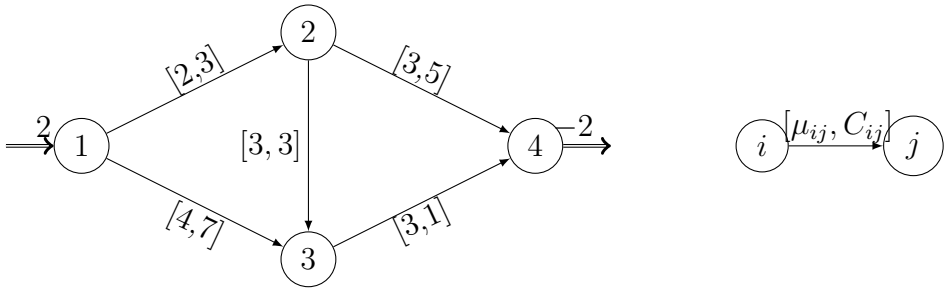
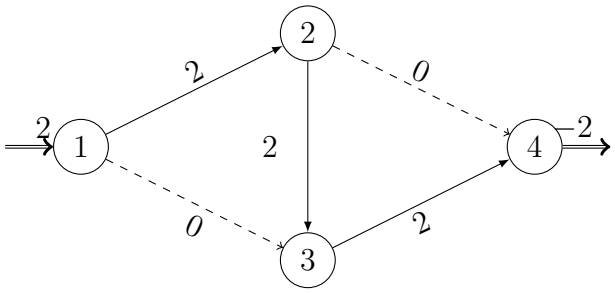



FIGURE 4.6 – Problème de flot maximum à coût minimum

La solution de base associée à la figure 4.6 est donnée par la figure 4.7




l'arbre $T$

FIGURE 4.7 – La solution arbre T

La méthode du simplexe réseau est :
**Iteration 1 :**

Calcul des potentiels :

on pose $y_4 = 0$
$\delta_{34} = C_{34} - y_3 + y_4 = 1 - y_3 + 0 = 0 \quad \Longrightarrow y_3 = 1$
$\delta_{23} = C_{23} - y_2 + y_3 = 3 - y_2 + 1 = 0 \quad \Longrightarrow y_2 = 4$
$\delta_{12} = C_{12} - y_1 + y_2 = 3 - y_1 + 4 = 0 \quad \Longrightarrow y_1 = 7$

Calcul du vecteur des coût réduits :

$\delta_{13} = C_{13} - y_1 + y_3 = 7 - 7 + 1 = 1 \ > 0 \ \ et \ x_{13} = 0$
$\delta_{24} = C_{24} - y_2 + y_4 = 5 - 4 + 0 = 1 \ > 0 \ \ et \ x_{24} = 0$

La solution courant optimal

$$Z = \sum_{(i,j)\in T} C_{ij} x_{ij} = 2 \times 3 + 2 \times 3 + 2 \times 1 = 14$$

# CHAPITRE 5

# APPLICATION ET ÉVALUATION

Le modèle de transport est un modèle linéaire, qui peut être traité par l'algorithme du simplexe. Cependant, la structure particulière de ce modèle permet de simplifier considérablement l'algorithme. Nous illustrons ici ce que devient l'algorithme lorsqu'on utilise les méthodes de la recherche de solution optimale de base de problème de transport présentées dans le chapitre précédent, en les appliquant pour faire face à quelques situations réelles afin d'expliquer leurs fonctionnements, et manipulant l'algorithme sous forme d'un programme informatique (programmé en langage C) capable de résoudre un problème de transport équilibré.

## 5.1 Langages utilisés

**C** est un langage de programmation impératif et généraliste. Inventé au début des années 1970 pour réécrire UNIX, C est devenu un des langages les plus utilisés. De nombreux langages plus modernes comme C++, C#, Java et PHP reprennent des aspects de C.

Le C est un langage de programmation de bas niveau très populaire, crée dans les années 1970 par D.Ritchie et B.W.Kernighan. Il est portable, libre, faiblement typé (peu de types de variables différents : son fonctionnement est donc proche de l'ordinateur (gain en rapidité), mais un brin plus difficile à manipuler pour le programmeur). Le C n'est sans doute pas le langage le plus facile à apprendre (notamment à cause de l'adoption du concept parfois un peu obscure des pointeurs), ni le plus récent, mais ses qualités font de lui un langage incontournable en matière de programmation.

Le fameux Hello Word :

```
#include <stdio.h>
int main()
{
  printf("hello, world\n");
  return 0;
}
```

## 5.2 Technologies utilisées

**Dev-C++** est un environnement de développement intégré (IDE) permettant de programmer en C et en C++. Développé avec Borland Delphi 6, Dev-C++ était disponible uniquement sous Microsoft Windows. Longtemps à l'abandon, le projet a été repris par un autre développeur en 2011 et est régulièrement mis à jour.

Cet IDE complet comprend entre autres un « répertoire de classes », servant à localiser facilement les fonctions, classes et membres du code source, un « répertoire de fonctions incluses », fonctionnant comme le répertoire de classes mais pour chercher dans les fichiers inclus (header), et un débogueur qui permet de surveiller l'état des variables pendant l'exécution du programme. Il souffre en revanche de l'absence d'un éditeur de ressources, ce qui rend la conception d'applications délicate si on ne fait pas appel à un outil externe.

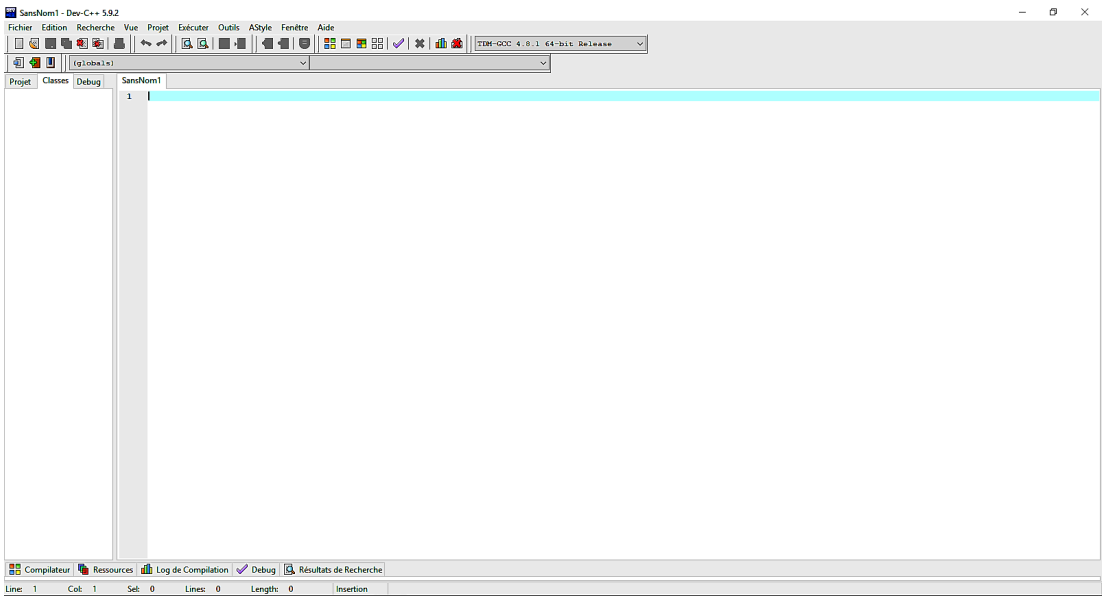



FIGURE 5.1 – Interface de Dev-C++

### 5.2.1 Fichier nouveau

Si votre programme tient dans un seul fichier et n'a pas besoin de bibliothèques particulières, vous pouvez utiliser Dev-C++ sans créer de projet. Pour cela il vous suffit de lancer Dev-C++ puis de créer un fichier source : commande **Nouveau > Fichier Source** du menu **Fichier** (beaucoup de commandes des menus s'obtiennent aussi par des boutons de la barre d'outils et/ou par des raccourcis clavier).

Enregistrez immédiatement ce fichier à l'aide des commandes **Sauvegarder** ou **Sauvegarder Sous...** du menu **Fichier**.

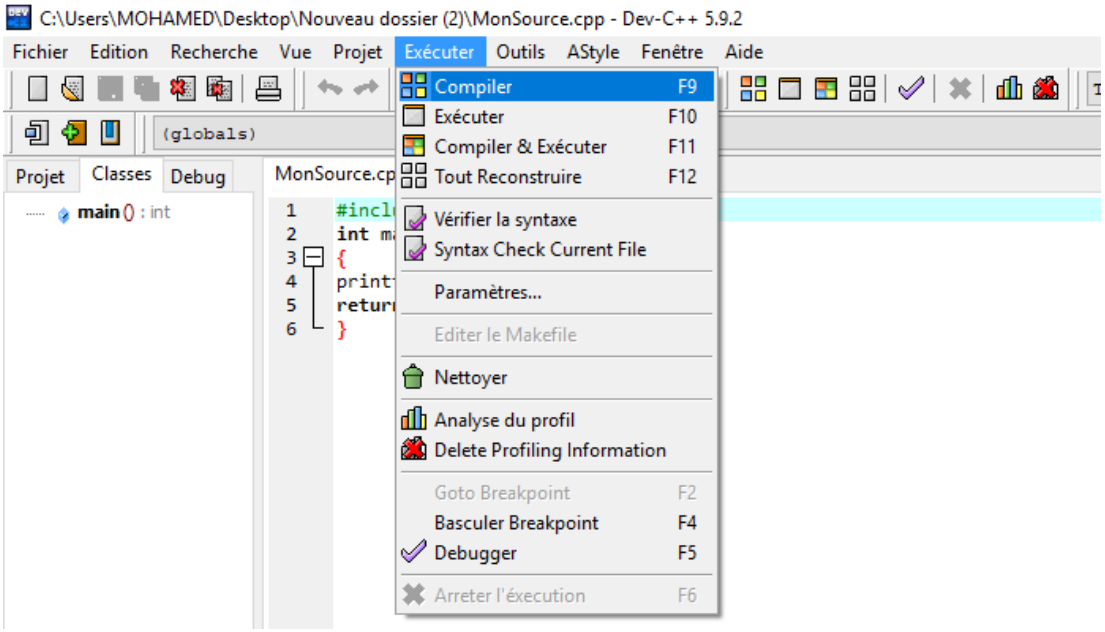
### 5.2.2 Fichier existant

Dans le cas où vous voulez travailler sur un fichier qui existe déjà, vous pouvez l'ouvrir dans Dev-C++ par la commande **Ouvrir Projet ou Fichier...** du menu **Fichier**. D'autre part, si Windows est bien configuré (c'est le cas, en principe, si l'installation s'est bien passée), les icônes des fichiers .CPP ressemblent à l'icône de la figure ci-dessous :

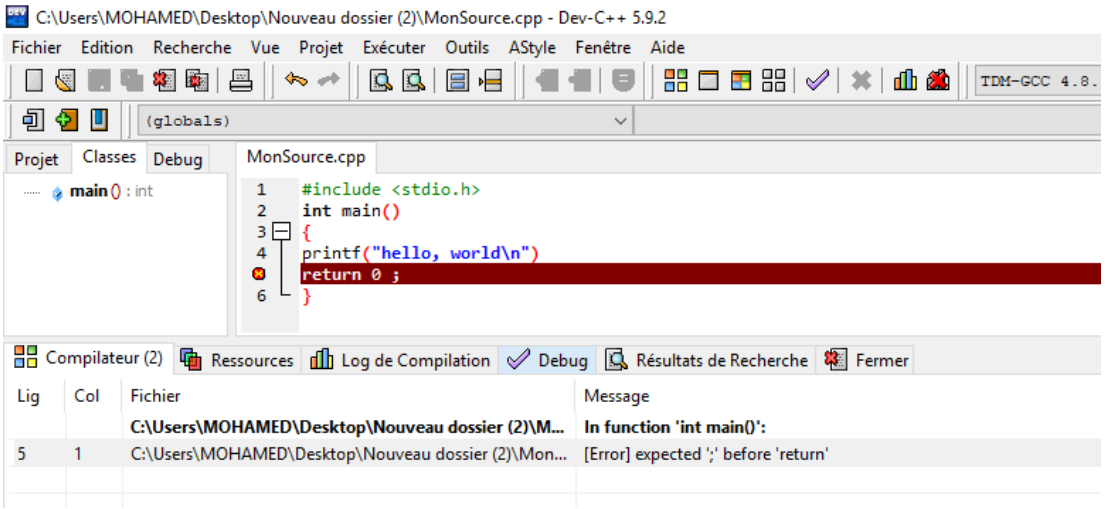

Il faut cliquer deux fois sue l'icône CPP pour lancer Dev-C++.

### 5.2.3 Fichier «compilé» et «exécuté»

Pour compiler le fichier, utilisez le menu **Exécuter>Compiler** ou cliquez sur le bouton avec la même icône en haut à gauche de la fenêtre ci-dessous.

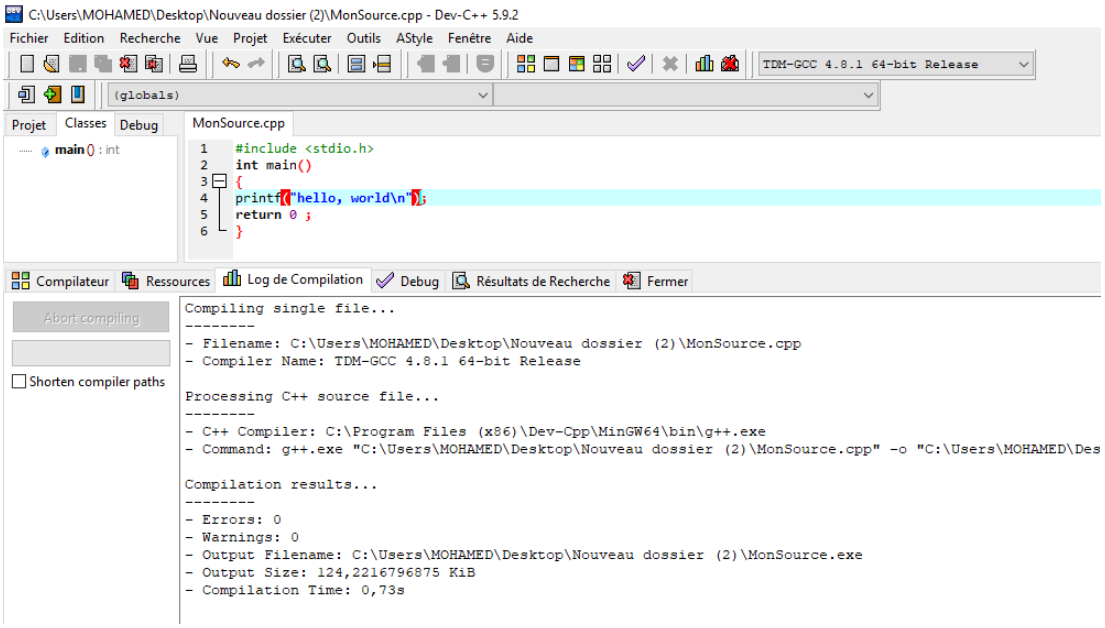



S'il y a des erreurs dans votre programme, les messages d'erreur seront affichés dans la fenêtre du compilateur (bas de la fenêtre) de la figure ci-dessous. Double-cliquez sur la première erreur : vous serez alors amenés à la ligne du programme ou se situe l'erreur.

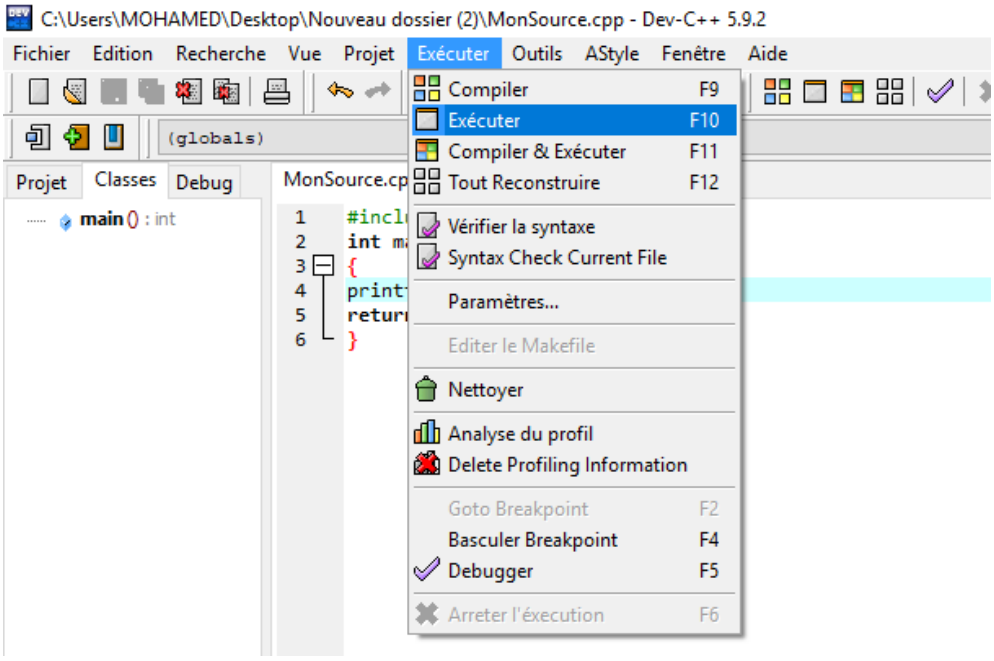



L'erreur signalé par une couleur spéciale et une marque dans la marge.

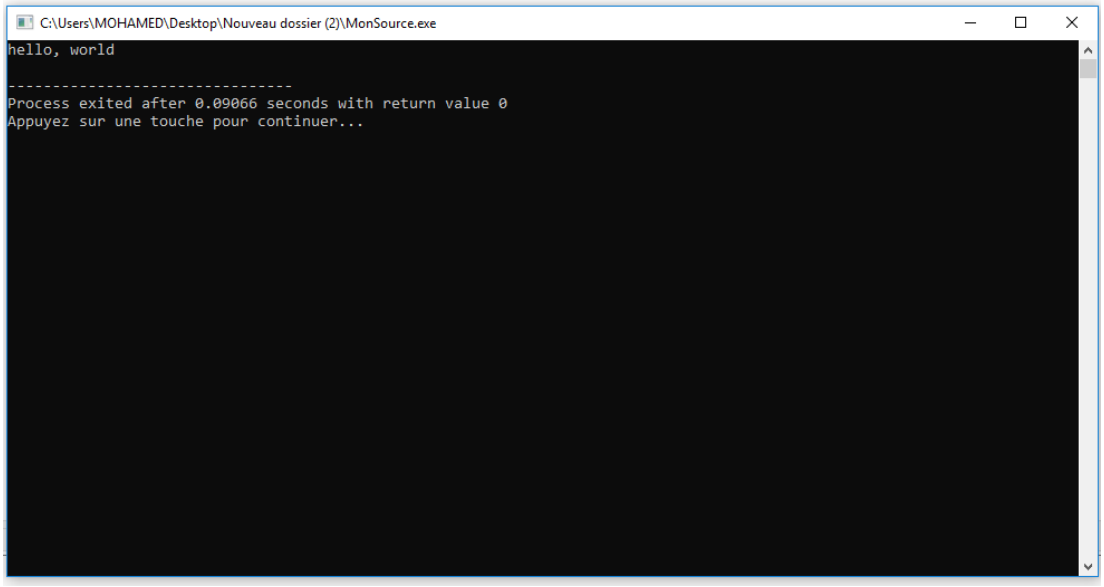
Une fois que les erreurs ont été corrigées la fenêtre de compilation Compile progresse nous informera que c'est terminé. Vous pouvez alors la fermer. Comme montré dans la fenêtre ci-dessous :




Vous pouvez alors exécuter votre programme. Pour cela utilisez le menu **Exécuter>Exécutez** ou tapez [**F10**] ou cliquez sur le bouton avec la même icône en haut à gauche. Comme indiqué dans la fenêtre ci-dessous :



La fenêtre console d'exécution apparaît alors :




Le programme se déroule dans la fenêtre ci-dessus. L'affichage avec **printf** et le saisie avec **scanf**. À la fin de l'exécution du programme, s'affichera «Appuyez sur une touche pour continuer....». Appuyez sur une touche, la fenêtre d'exécution se fermera alors et vous reviendrez à votre programme.

Pour compiler et exécuter directement, tapez [**F9**].

## 5.3 Modélisation de problème de transport et résolution

### 5.3.1 Problématique :

Le transport de blé implique des matériels et des qualifications spécifiques selon les phases de sa transformation. Celle qui va du champ de l'agriculteur au moulin de la coopérative et celle qui va du moulin aux usines de transformation des industriels.

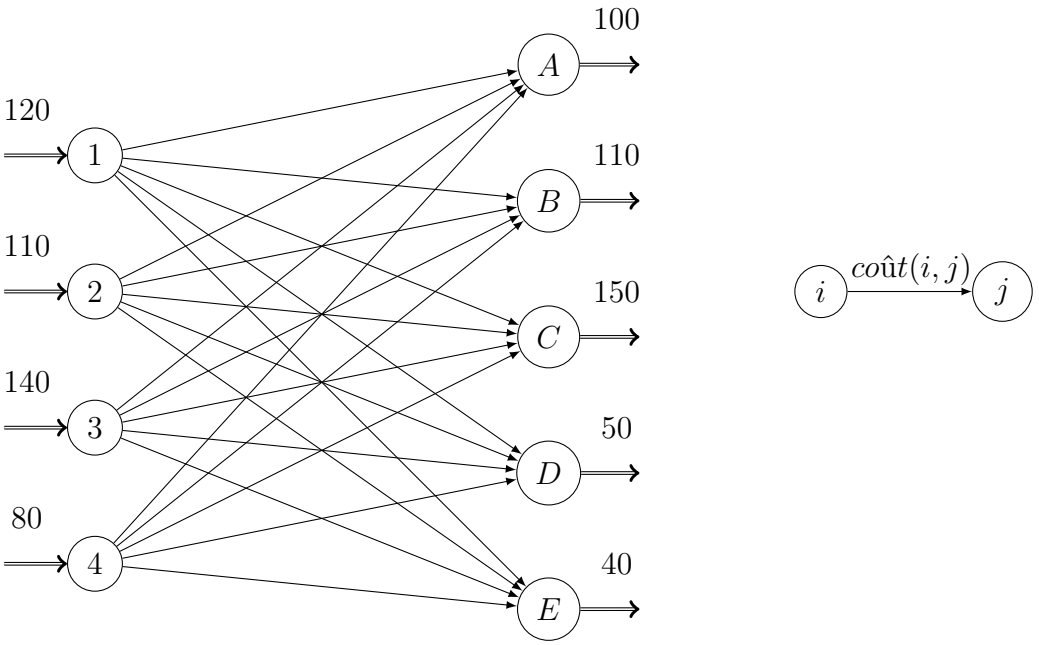
Le problème est de déterminer combien de tonnes de blé à transporter de chaque élévateur à grain à chaque moulin, afin de minimiser le coût total du transport.

### 5.3.2 Modélisation

la variable de décision $x_{ij}$ représente le nombre de tonnes de blé transporté par chaque élévateur $i$ ($o\grave{u} \quad i = 1, 2, 3, 4$), à chaque moulin $j$ ($o\grave{u} \quad j = A, B, C, D, E$). La fonction objective représente le coût de transport total pour chaque itinéraire, Chaque terme dans la fonction objective reflète le coût du tonnage transporté pour une route.

| m / é | A | B | C | D | E | Offre |
|---|---|---|---|---|---|---|
| 1 | 7 | 12 | 1 | 5 | 9 | 120 |
| 2 | 15 | 3 | 12 | 6 | 14 | 110 |
| 3 | 8 | 16 | 10 | 12 | 7 | 140 |
| 4 | 18 | 8 | 17 | 11 | 16 | 80 |
| Demande | 100 | 110 | 150 | 50 | 40 | 450 |

le graphe associe au cette tableau est comme suit :



### 5.3.3 Formulation mathématique

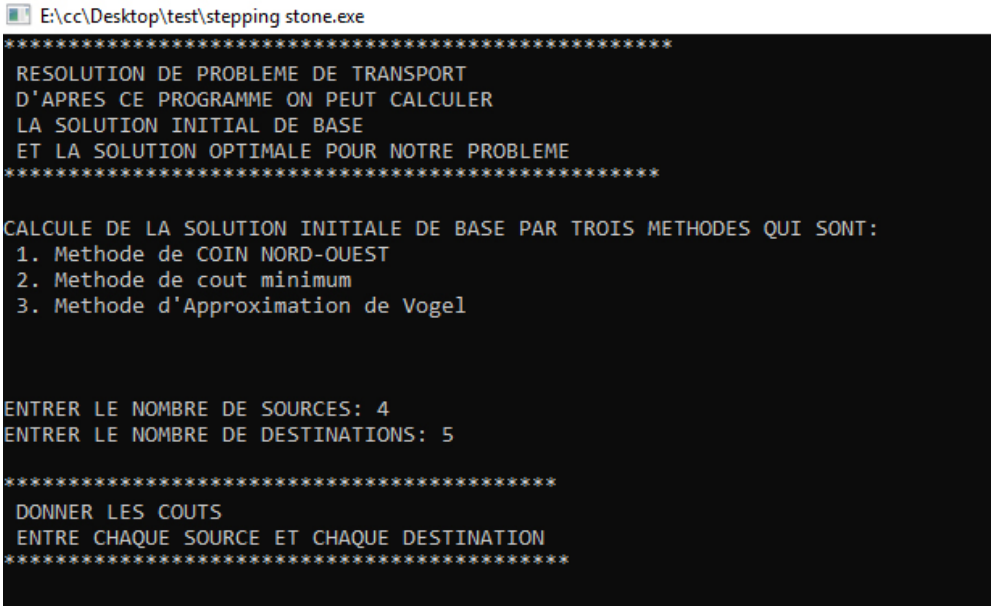
Le modèle de transport deviendra alors :

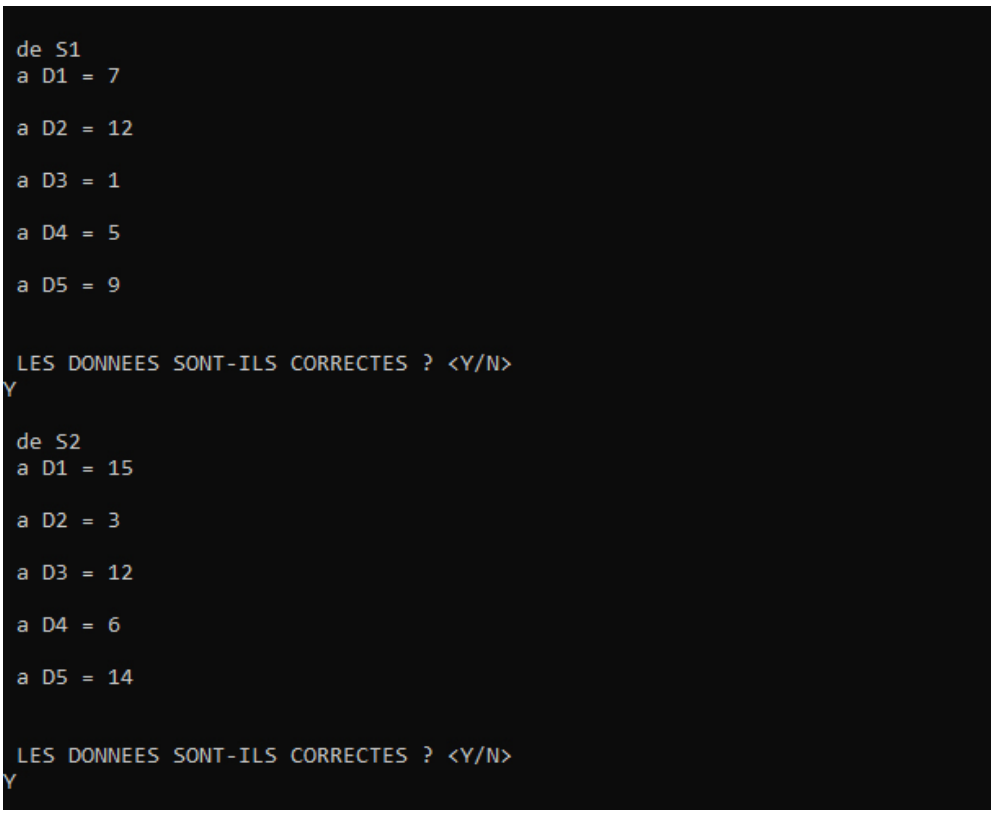
$$\begin{cases} \min z = & 7x_{11} + 12x_{12} + x_{13} + 5x_{14} + 9x_{15} + 15x_{21} + 3x_{22} + 12x_{23} + 6x_{24} + 14x_{25} \\ & +8x_{31} + 16x_{32} + 10x_{33} + 12x_{34} + 7x_{35} + 18x_{41} + 8x_{42} + 17x_{43} + 11x_{44} + 16x_{45} \\ & \\ & x_{11} + x_{12} + x_{13} + x_{14} + x_{15} = 120 \\ & x_{21} + x_{22} + x_{23} + x_{24} + x_{25} = 110 \\ & x_{31} + x_{32} + x_{33} + x_{34} + x_{35} = 140 \\ & x_{41} + x_{42} + x_{43} + x_{44} + x_{45} = 80 \\ & \\ & x_{11} + x_{21} + x_{31} + x_{41} = 100 \\ & x_{12} + x_{22} + x_{32} + x_{42} = 110 \\ & x_{13} + x_{23} + x_{33} + x_{43} = 150 \\ & x_{14} + x_{24} + x_{34} + x_{44} = 50 \\ & x_{15} + x_{25} + x_{35} + x_{45} = 80 \\ & \\ & \forall x_{ij} \geq 0 \end{cases}$$

### 5.3.4 Résolution

• L'exécution de programme : Le programme est effectué de façon à afficher tous les détails des calculs par des étapes et ne pas pour donner le résultat seulement, on va afficher quelques résultats

dans les figures suivant on entrer les données, avec suivre les instructions et dans la dernière instruction, Le programme vous demande de choisir la méthode pour déterminer la solution de base parmi les trois méthodes.


E:\cc\Desktop\test\stepping stone.exe
**************************************************
RESOLUTION DE PROBLEME DE TRANSPORT
D'APRES CE PROGRAMME ON PEUT CALCULER
LA SOLUTION INITIAL DE BASE
ET LA SOLUTION OPTIMALE POUR NOTRE PROBLEME
**************************************************
CALCULE DE LA SOLUTION INITIALE DE BASE PAR TROIS METHODES QUI SONT:
1. Methode de COIN NORD-OUEST
2. Methode de cout minimum
3. Methode d'Approximation de Vogel
ENTRER LE NOMBRE DE SOURCES: 4
ENTRER LE NOMBRE DE DESTINATIONS: 5
**********
DONNER LES COUTS
ENTRE CHAQUE SOURCE ET CHAQUE DESTINATION
**********


de S1
a D1 = 7
a D2 = 12
a D3 = 1
a D4 = 5
a D5 = 9
LES DONNEES SONT-ILS CORRECTES ? <Y/N>
Y
de S2
a D1 = 15
a D2 = 3
a D3 = 12
a D4 = 6
a D5 = 14
LES DONNEES SONT-ILS CORRECTES ? <Y/N>
Y

```
 de S3
 a D1 = 8

 a D2 = 16

 a D3 = 10

 a D4 = 12

 a D5 = 7


 LES DONNEES SONT-ILS CORRECTES ? <Y/N>
Y

 de S4
 a D1 = 18

 a D2 = 8

 a D3 = 17

 a D4 = 11

 a D5 = 16


 LES DONNEES SONT-ILS CORRECTES ? <Y/N>
Y
```

```
*********************************************
 DONNER LES OFFRES
 POUR CHAQUE SOURCE
*********************************************
 S1 = 120

 S2 = 110

 S3 = 140

 S4 = 80

 LES DONNEES SONT-ILS CORRECTES ? <Y/N>
Y


*********************************************
 DONNER LES DEMANDES
 POUR CHAQUE DESTINATION
*********************************************
 D1 = 100

 D2 = 110

 D3 = 150

 D4 = 50

 D5 = 40

 LES DONNEES SONT-ILS CORRECTES ? <Y/N>
Y
```

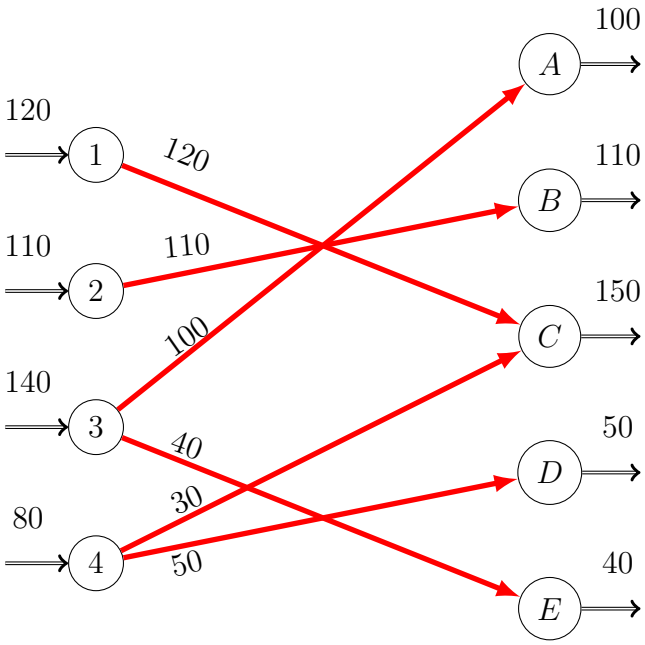
les trois figures suivant représente la solution de base avec trois méthode différant et la solution optimale avec la méthode de Stepping Stone

chaque fois que nous exécuter le programme donnée pour la solution de base, la matrice et solution réalisable ou n'est pas réalisable, le coût total et le nombre d'itération, ainsi que la solution optimale donnée l'affectation des sources à des destination et le coût totale de transport, le nombre d'itération et le temps d'exécution.

```
*** LA SOLUTION INITIALE DE BASE DE LA METHODE COIN NORD-OUEST ***

   100.000000    20.000000     0.000000     0.000000     0.000000
     0.000000    90.000000    20.000000     0.000000     0.000000
     0.000000     0.000000   130.000000    10.000000     0.000000
     0.000000     0.000000     0.000000    40.000000    40.000000


 la solution initial est realisable.
le cout total = 3950.000000

le nombre d iteration est : 13


*** LA SOLUTION OPTIMALE DE COIN NORD-OUEST ***

 DE SOURCE1 A DESTINATION3:   120.00
 DE SOURCE2 A DESTINATION2:   110.00
 DE SOURCE3 A DESTINATION1:   100.00
 DE SOURCE3 A DESTINATION5:    40.00
 DE SOURCE4 A DESTINATION3:    30.00
 DE SOURCE4 A DESTINATION4:    50.00

 LE COUT TOTALE DE TRANSPORT:     2590.0

 le nombre d iteration est : 5



**************************************************
le temp d execution est : 153.000000
```

```
*** LA SOLUTION INITIALE DE BASE DE LA METHODE MINIMUM ***

     0.000000     0.000000   120.000000     0.000000     0.000000
     0.000000   110.000000     0.000000     0.000000     0.000000
   100.000000     0.000000     0.000000     0.000000    40.000000
     0.000000     0.000000    30.000000    50.000000     0.000000


 la solution initial est realisable.
le cout total = 2590.000000

le nombre d iteration est : 6


*** LA SOLUTION OPTIMALE DE COUT MINIMUM ***

 DE SOURCE1 A DESTINATION3:   120.00
 DE SOURCE2 A DESTINATION2:   110.00
 DE SOURCE3 A DESTINATION1:   100.00
 DE SOURCE3 A DESTINATION5:    40.00
 DE SOURCE4 A DESTINATION3:    30.00
 DE SOURCE4 A DESTINATION4:    50.00

 LE COUT TOTALE DE TRANSPORT:     2590.0

 le nombre d iteration est : 1



**************************************************
le temp d execution est : 530.000000
```

```
*** LA SOLUTION INITIALE DE BASE DE LA METHODE VOGEL ***

     0.000000     0.000000   120.000000     0.000000     0.000000
     0.000000   110.000000     0.000000     0.000000     0.000000
   100.000000     0.000000     0.000000     0.000000    40.000000
     0.000000     0.000000    30.000000    50.000000     0.000000


 la solution initial est realisable.
le cout total = 2590.000000

le nombre d iteration est : 3


*** LA SOLUTION OPTIMALE DE VOGEL ***

 DE SOURCE1 A DESTINATION3:   120.00
 DE SOURCE2 A DESTINATION2:   110.00
 DE SOURCE3 A DESTINATION1:   100.00
 DE SOURCE3 A DESTINATION5:    40.00
 DE SOURCE4 A DESTINATION3:    30.00
 DE SOURCE4 A DESTINATION4:    50.00

 LE COUT TOTALE DE TRANSPORT:     2590.0

 le nombre d iteration est : 1



******************************************************
le temp d execution est : 675.000000
```



### 5.3.5 Complexité algorithmique

#### Coin du Nord-Ouest

Certainement la méthode la plus facile à mettre en œuvre et de complexité minimale, seulement en $O(m+n)$ . Elle consiste à déterminer la variable de plus petits indices où il est possible d'affecter une quantité. Sa simplicité l'a rendue très populaire. Par contre, la solution trouvée n'a aucune raison d'être prés de la solution optimale.

#### Vogel

La complexité de cette méthode est $O(mnln(mn) + (m+n)^2)$ . En effet, au début il faut ordonner, par valeurs croissantes, les $n$ coûts pour chacune des $m$ lignes et les $m$ coûts pour chacune des $n$ colonnes, ce qui donne $O(mnln(mn))$. Ensuite, à chaque itération et dans le pire des cas, il y a un calcul des $m+n$ différences sur les lignes et les colonnes, cela donne une complexité de $O(m+n)$ par itération. Comme il y a au plus $m+n$ itérations, le résultat suit.

#### Coût minimum

La méthode est facile à mettre en ouvre , elle trouve des solution de départ proche à la solution optimal , cette méthode est de complexité $O((m+n)^2)$ , la méthode consiste à déterminer le coût le plus petits de transporter une quantité .

### 5.3.6 Résultats numériques : comparaison

Nous comparons la performance des méthodes Vogel , coût minimum et la méthode du coin du Nord-Ouest

#### Caractéristiques des tests

Les tests numériques sont effectués sur un échantillon de 30 problèmes dont les dimensions sont données au tableau suivant

| offre | demande | Coin Nord-Ouest | Vogel | Coût minimum | Stepping stone |
|---|---|---|---|---|---|
| 3 | 3 | 5925 | 5125 | **4555*** | 4525 |
| 2 | 3 | 17050 | **16550*** | **16550*** | 16550 |
| 3 | 2 | **313200*** | **313200*** | **313200*** | 313200 |
| 3 | 4 | 35500 | **33620*** | **34140** | 33620 |
| 3 | 4 | 165595 | **152535*** | **152535*** | 152535 |
| 3 | 4 | 1180 | **995*** | 1080 | 995 |
| 3 | 4 | 520 | **475*** | **475*** | 475 |
| 3 | 5 | 3795 | **2810*** | 3195 | 2810 |
| 3 | 5 | 1432160 | **1323560** | 1389800 | 1316960 |
| 4 | 5 | 395 | **259*** | **259*** | 259 |
| 4 | 5 | 181721 | **101605** | **101605** | 97865 |
| 5 | 5 | 1994 | **1104*** | **1123** | 1104 |
| 5 | 8 | 3825 | **2070*** | **2070*** | 2070 |
| 8 | 8 | 4870 | **2780** | **2765*** | 2765 |
| 7 | 9 | 27184 | **9600** | 11427 | 9312 |
| 9 | 9 | 20221 | **11685** | 11976 | 11595 |
| 9 | 10 | 34862 | **11856** | 11932 | 11362 |
| 10 | 10 | 1152100 | **558100** | 683200 | 557800 |
| 11 | 12 | 1223050 | **577800** | 706300 | 552150 |
| 10 | 15 | 54300 | **19870** | 21420 | 19110 |
| 10 | 20 | 75180 | **19390** | 21470 | 19310 |
| 20 | 15 | 50030 | **17220** | 23050 | 15780 |
| 20 | 20 | 57980 | **16360** | 19860 | 16020 |
| 25 | 40 | 67714 | **11310** | 13278 | 11202 |
| 30 | 50 | 15099 | **2829** | 3704 | 2511 |
| 50 | 50 | 13586 | **1800** | 2202 | 1554 |
| 50 | 80 | 21301 | 2322 | **2310** | 2245 |
| 80 | 80 | 24488 | **1953** | 2498 | 1749 |
| 80 | 90 | 19811 | **1947** | 2419 | 1889 |
| 100 | 100 | 30372 | **2033** | 2866 | 1949 |

Certains de ces problèmes sont tirés de la littérature, d'autres ont été générés aléatoirement. Les problèmes 1 à 17 sont de petite taille puisqu'ils comportent moins de 100 variables et les problèmes 18 à 30 sont de grande taille puisqu'ils comportent au moins 100 variables.

**Comparaison**

Le tableau donne la valeur de l'objectif pour chacune des trois méthodes Vogel , coût minimum et la méthode du coin du Nord-Ouest Nous présentons en caractères gras la meilleure solution parmi celles données par les trois méthodes. Le signe * indique que la solution est optimale, l'optimalité ayant été vérifiée par la méthode de Stepping-Stone via notre programme.

Nous observons que pour les problèmes de petite taille la méthode VOGEL et CM semble mieux se comporter que la méthode CNO . En effet nous observons qu'elles donnes une meilleure solution pour 15 des 17 problèmes de petite taille pour la méthode de VOGEL et 9 des 17 pour la méthode CM .
De plus elles donne une solution optimale pour 10 problèmes contre 8 pour la méthode de CM et 1 pour la méthode du coin du Nord-Ouest .
Nous en déduisons ainsi que de ce point de vue la méthode VOGEL est supérieure aux deux autres méthodes et que dans beaucoup de cas elle permet d'obtenir une solution optimale.

Pour les problèmes de plus grande taille, nous remarquons que aussi la méthode VOGEL semble être à efficace .

Nous en déduisons que de ce point de vue la méthode VOGEL est meilleure que la méthode de CNO et est compétitive avec la méthode de CM sur des problèmes de petit taille et est meilleur que les deux méthodes sur des problèmes grand taille .

pour obtenir la solution optimale à partir de la solution initiale fournie par les trois méthodes d'initialisation. Nous ne donnons pas les temps pour les 17 premiers problèmes car pour ces problèmes de petite taille le temps d'exécution est approximativement le même et est négligeable pour les trois méthodes.

Pour les problèmes de taille plus grande, nous observons que la méthode VOGEL est plus sensible à la taille du problème que les deux autres méthodes. En ce qui concerne le nombre d'itérations fait par notre programme .

Le nombre d'itérations pour les trois méthodes augmente avec la taille du problème.

## 5.4 Modélisation d'un problème d'affectation et résolution

On modélisé le problème précédent comme un problème d'affectation ,donc la demande et l'offre sont égale à 1 ,d'où le problème ira comme suit

| é \ m | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | 7 | 12 | 1 | 5 | 9 |
| 2 | 15 | 3 | 12 | 6 | 14 |
| 3 | 8 | 16 | 10 | 12 | 7 |
| 4 | 18 | 8 | 17 | 11 | 16 |

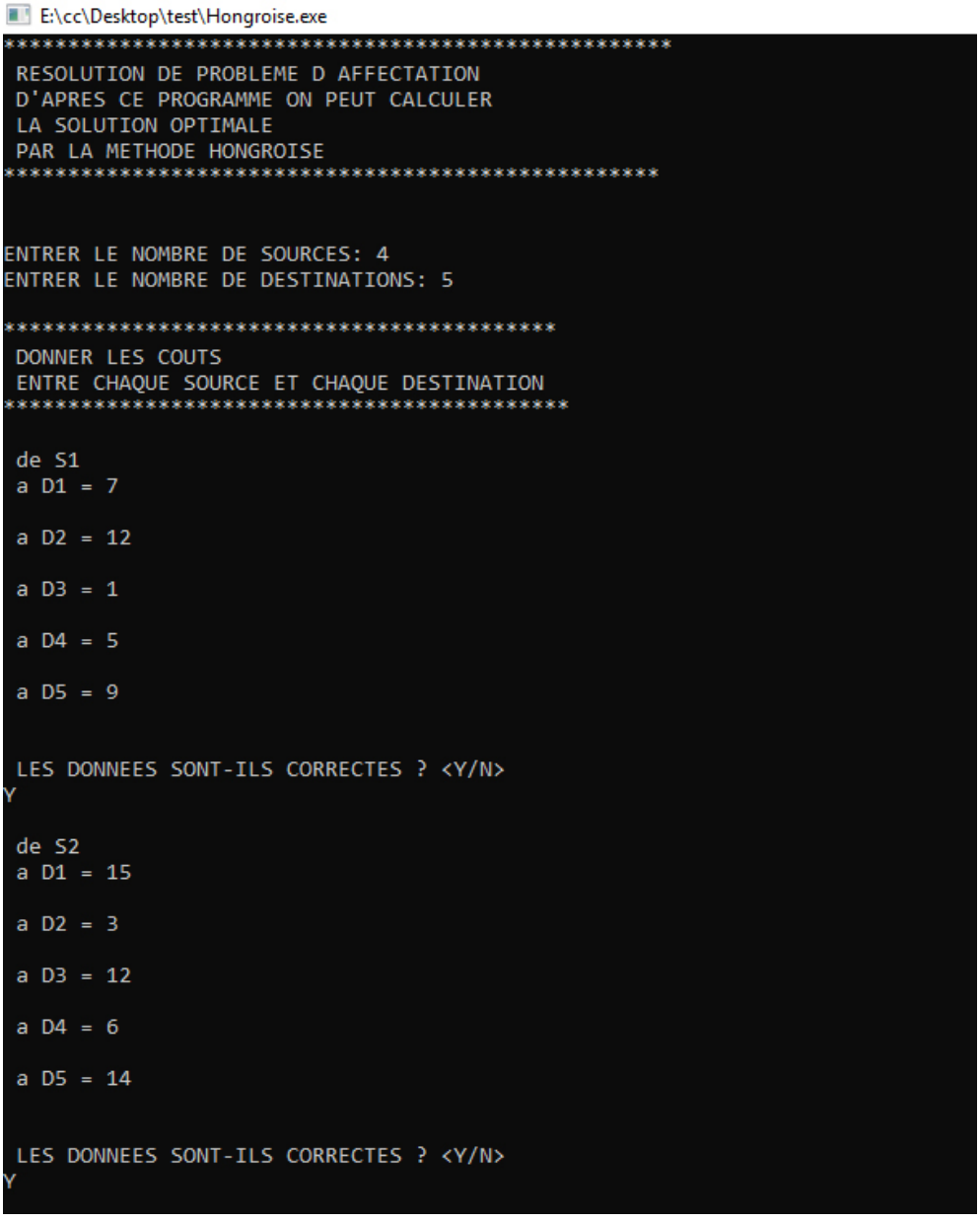
TABLE 5.1 – Tableau d'affectation

### 5.4.1 Formulation mathématique

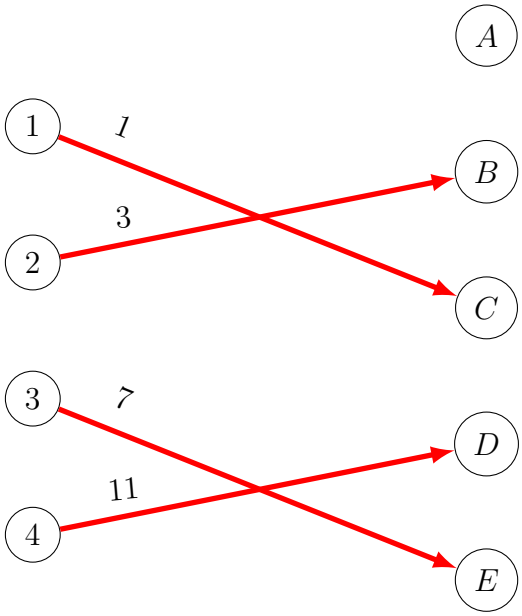
Le modèle de d'affectation deviendra alors :

$$
\begin{cases}
\min z = & 7x_{11} + 12x_{12} + x_{13} + 5x_{14} + 9x_{15} + 15x_{21} + 3x_{22} + 12x_{23} + 6x_{24} + 14x_{25} \\
& +8x_{31} + 16x_{32} + 10x_{33} + 12x_{34} + 7x_{35} + 18x_{41} + 8x_{42} + 17x_{43} + 11x_{44} + 16x_{45} \\
\\
& x_{11} + x_{12} + x_{13} + x_{14} + x_{15} = 1 \\
& x_{21} + x_{22} + x_{23} + x_{24} + x_{25} = 1 \\
& x_{31} + x_{32} + x_{33} + x_{34} + x_{35} = 1 \\
& x_{41} + x_{42} + x_{43} + x_{44} + x_{45} = 1 \\
\\
& x_{11} + x_{21} + x_{31} + x_{41} = 1 \\
& x_{12} + x_{22} + x_{32} + x_{42} = 1 \\
& x_{13} + x_{23} + x_{33} + x_{43} = 1 \\
& x_{14} + x_{24} + x_{34} + x_{44} = 1 \\
& x_{15} + x_{25} + x_{35} + x_{45} = 1 \\
\\
& \forall x_{ij} \geq 0
\end{cases}
$$

### 5.4.2 Résolution


E:\cc\Desktop\test\Hongroise.exe
**********
RESOLUTION DE PROBLEME D AFFECTATION
D'APRES CE PROGRAMME ON PEUT CALCULER
LA SOLUTION OPTIMALE
PAR LA METHODE HONGROISE
**********
ENTRER LE NOMBRE DE SOURCES: 4
ENTRER LE NOMBRE DE DESTINATIONS: 5
**********
DONNER LES COUTS
ENTRE CHAQUE SOURCE ET CHAQUE DESTINATION
**********
de S1
a D1 = 7
a D2 = 12
a D3 = 1
a D4 = 5
a D5 = 9
LES DONNEES SONT-ILS CORRECTES ? <Y/N>
Y
de S2
a D1 = 15
a D2 = 3
a D3 = 12
a D4 = 6
a D5 = 14
LES DONNEES SONT-ILS CORRECTES ? <Y/N>
Y

```
 de S3
 a D1 = 8

 a D2 = 16

 a D3 = 10

 a D4 = 12

 a D5 = 7


 CETTE DONNEE EST CORRECTE ? <Y/N>
Y

 de S4
 a D1 = 18

 a D2 = 8

 a D3 = 17

 a D4 = 11

 a D5 = 16


 CETTE DONNEE EST CORRECTE ? <Y/N>
Y


*** LA SOLUTION OPTIMALE DE HONGROISE ***

     0.000000     0.000000     1.000000     0.000000     0.000000
     0.000000     3.000000     0.000000     0.000000     0.000000
     0.000000     0.000000     0.000000     0.000000     7.000000
     0.000000     0.000000     0.000000    11.000000     0.000000
     0.000000     0.000000     0.000000     0.000000     0.000000

 LE COUT TOTALE D AFFECTATION:       22.0


 le nombre d iteration est : 2



**************************************************
le temp d execution est : 39.000000
```




### 5.4.3 Complexité de l'algorithme

Cet algorithme est de complexité polynomiale ,complexité qu'on peut évaluer dans un premier temps en O($n^4$) où $n$ est le cardinal commun des ensemble $X$ et $Y$

## 5.5 Modélisation d'un problème de flot maximum à coût minimal

En applique la méthode de simplexe réseau implémenté sure le C++ pour déterminer le flot maximum à cout minimum du problème de transport précédent , après que en le modélisé comme un problème de flot .

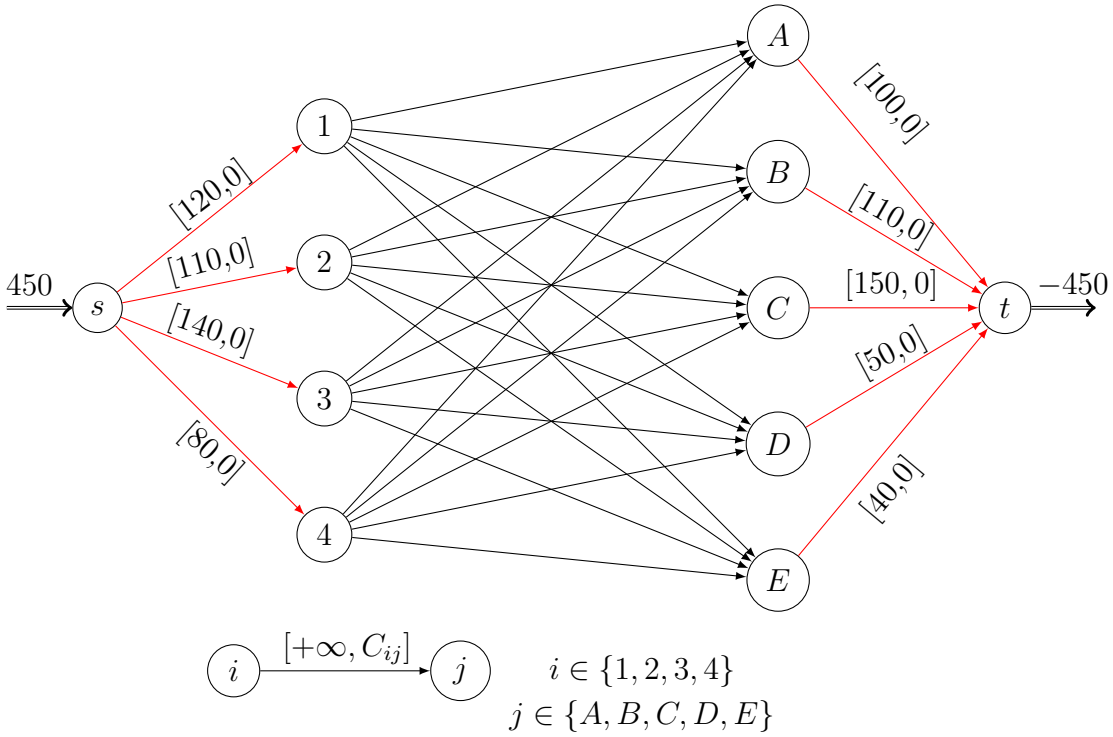
### 5.5.1 Modélisation

Le problème de flot comporte un source unique $s$ et une seul puit $t$ , le problème de transport précédent comporte plusieurs sources $X = \{1, 2, 3, 4\}$ et plusieurs puits $Y = \{A, B, C, D, E\}$ , on se ramène au reseau et on ajoutant à l'ensemble des sources $X$ un sommet $s$ et à $E(s, X_i)$ $\forall i = 1, 2, 3, 4$ , avec un capacité $\mu(s, X_i)$ égale à l'offre de chaque source $i$ et le coût $C(s, X_i) = 0$ .

On ajoute aussi à l'ensemble des puits un sommet $t$ et à $E(Y_j, t)$ $\forall j = 1, 2, 3, 4, 5$ , avec un capacité $\mu(Y_j, t)$ égale à la demande de chaque puits $j$ et le coût $C(Y_j, t) = 0$ .

On met la capacité de transporté un quantité entre les sources et les puits illimité $\mu(X, Y) = +\infty$.

D'ou , on obtient le problème de flot suivant :

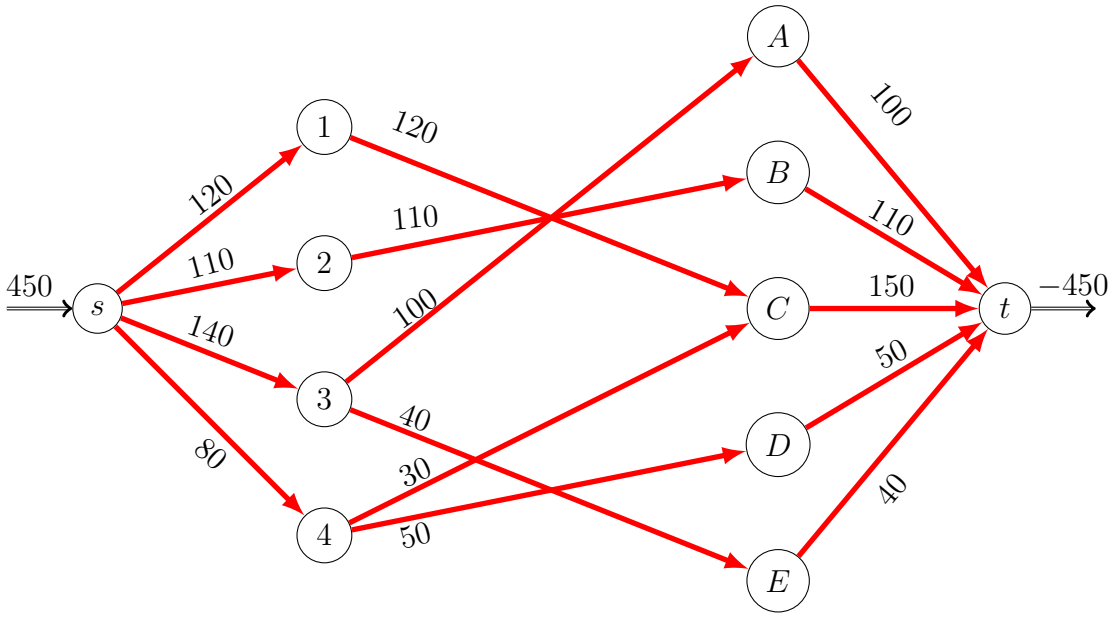



### 5.5.2 Résolution

C:\Users\MOHAMED\Desktop\simplexe_reseau.exe

```
**********************************************************
Graphe initial :
de sommet 1    vers le sommet 2    le flux est : 0    , et le cout egale : 0
de sommet 1    vers le sommet 3    le flux est : 0    , et le cout egale : 0
de sommet 1    vers le sommet 4    le flux est : 0    , et le cout egale : 0
de sommet 1    vers le sommet 5    le flux est : 0    , et le cout egale : 0
de sommet 2    vers le sommet 6    le flux est : 0    , et le cout egale : 7
de sommet 2    vers le sommet 7    le flux est : 0    , et le cout egale : 12
de sommet 2    vers le sommet 8    le flux est : 0    , et le cout egale : 1
de sommet 2    vers le sommet 9    le flux est : 0    , et le cout egale : 5
de sommet 2    vers le sommet 10   le flux est : 0    , et le cout egale : 9
de sommet 3    vers le sommet 6    le flux est : 0    , et le cout egale : 15
de sommet 3    vers le sommet 7    le flux est : 0    , et le cout egale : 3
de sommet 3    vers le sommet 8    le flux est : 0    , et le cout egale : 12
de sommet 3    vers le sommet 9    le flux est : 0    , et le cout egale : 6
de sommet 3    vers le sommet 10   le flux est : 0    , et le cout egale : 14
de sommet 4    vers le sommet 6    le flux est : 0    , et le cout egale : 8
de sommet 4    vers le sommet 7    le flux est : 0    , et le cout egale : 16
de sommet 4    vers le sommet 8    le flux est : 0    , et le cout egale : 10
de sommet 4    vers le sommet 9    le flux est : 0    , et le cout egale : 12
de sommet 4    vers le sommet 10   le flux est : 0    , et le cout egale : 7
de sommet 5    vers le sommet 6    le flux est : 0    , et le cout egale : 18
de sommet 5    vers le sommet 7    le flux est : 0    , et le cout egale : 8
de sommet 5    vers le sommet 8    le flux est : 0    , et le cout egale : 17
de sommet 5    vers le sommet 9    le flux est : 0    , et le cout egale : 11
de sommet 5    vers le sommet 10   le flux est : 0    , et le cout egale : 16
de sommet 6    vers le sommet 11   le flux est : 0    , et le cout egale : 0
de sommet 7    vers le sommet 11   le flux est : 0    , et le cout egale : 0
de sommet 8    vers le sommet 11   le flux est : 0    , et le cout egale : 0
de sommet 9    vers le sommet 11   le flux est : 0    , et le cout egale : 0
de sommet 10   vers le sommet 11   le flux est : 0    , et le cout egale : 0
```

```
**********************************************************

Graphe final :
de sommet 1    vers le sommet 2    le flux est : 120  , et le cout egale : 0
de sommet 1    vers le sommet 3    le flux est : 110  , et le cout egale : 0
de sommet 1    vers le sommet 4    le flux est : 140  , et le cout egale : 0
de sommet 1    vers le sommet 5    le flux est : 80   , et le cout egale : 0
de sommet 2    vers le sommet 6    le flux est : 0    , et le cout egale : 7
de sommet 2    vers le sommet 7    le flux est : 0    , et le cout egale : 12
de sommet 2    vers le sommet 8    le flux est : 120  , et le cout egale : 1
de sommet 2    vers le sommet 9    le flux est : 0    , et le cout egale : 5
de sommet 2    vers le sommet 10   le flux est : 0    , et le cout egale : 9
de sommet 3    vers le sommet 6    le flux est : 0    , et le cout egale : 15
de sommet 3    vers le sommet 7    le flux est : 110  , et le cout egale : 3
de sommet 3    vers le sommet 8    le flux est : 0    , et le cout egale : 12
de sommet 3    vers le sommet 9    le flux est : 0    , et le cout egale : 6
de sommet 3    vers le sommet 10   le flux est : 0    , et le cout egale : 14
de sommet 4    vers le sommet 6    le flux est : 100  , et le cout egale : 8
de sommet 4    vers le sommet 7    le flux est : 0    , et le cout egale : 16
de sommet 4    vers le sommet 8    le flux est : 0    , et le cout egale : 10
de sommet 4    vers le sommet 9    le flux est : 0    , et le cout egale : 12
de sommet 4    vers le sommet 10   le flux est : 40   , et le cout egale : 7
de sommet 5    vers le sommet 6    le flux est : 0    , et le cout egale : 18
de sommet 5    vers le sommet 7    le flux est : 0    , et le cout egale : 8
de sommet 5    vers le sommet 8    le flux est : 30   , et le cout egale : 17
de sommet 5    vers le sommet 9    le flux est : 50   , et le cout egale : 11
de sommet 5    vers le sommet 10   le flux est : 0    , et le cout egale : 16
de sommet 6    vers le sommet 11   le flux est : 100  , et le cout egale : 0
de sommet 7    vers le sommet 11   le flux est : 110  , et le cout egale : 0
de sommet 8    vers le sommet 11   le flux est : 150  , et le cout egale : 0
de sommet 9    vers le sommet 11   le flux est : 50   , et le cout egale : 0
de sommet 10   vers le sommet 11   le flux est : 40   , et le cout egale : 0

Le cout minimum pour le flot est : 2590

--------------------------------
Process exited after 0.2607 seconds with return value 0
Appuyez sur une touche pour continuer...
```




### 5.5.3 Complexité de l'algorithme

cet algorithme est de complexité théorique polynomial, complexité qu'on peut évaluer dans un premier temps en O($nm^2C\mu$). [19]

# CONCLUSION

Cette étude nous a donné l'opportunité de nous familiariser au domaine de la recherche opérationnelle, ce domaine qui est la discipline des méthodes scientifiques pour aider à mieux décider et traiter les problèmes stratégiques et économiques, Le problème de transport est l'un de ces problèmes classiques les plus connus, Mais la complexité et la variation des contraintes de ce problème dans le domaine économique impliquent la recherche d'autres heuristiques et même des méta-heuristiques plus efficaces pour la résolution. Ce qui rend difficile de tirer une conclusion définitive sur la résolution de ce type des problèmes.

Dans notre travail nous avons essayé de résoudre trois problèmes classique de la recherche opérationnelle . Ainsi nous avons implémenté les méthodes de la recherche d'une solution de base (coin nord ouest, aproximation de VOGEL, coût minimum ) et les méthodes de l'optimisation de la solution de base ( Stripping-Stone, distribution modifiée ) pour le problème de transport, la méthode d'hongroise pour la résolution du problème de l'affectation et les méthodes Ford-Fulkerson et Simplexe réseaux pour la résolution du problème de flot maximum et problème de flot maximum à coût minimum respectivement. Mais grâce aux développements scientifiques, différentes variations des trois problèmes ont été proposées, ainsi que différents méthodes de résolutions exactes et approchées.

Dans un premier lieu nous avons présenter quelques notions de base , puis quelques méthodes d'optimisation que nous avons appliqué à différents types de problèmes et en fin nous avons présente le logiciel Dev-C++ qui nous a permet de résoudre via une application les différentes problèmes de transport. Dans un prochain avenir nous espérons pouvoir continuer à travailler sur les problèmes de la recherche opérationnelle ainsi que leur résolution, car c'est un domaine vaste, riche et très intéressant.

# BIBLIOGRAPHIE


[1] A.Chetbani,H.Berdous,B.Benabbou et S.Boudjelda ,La recherche des points d'articulations dans un réseau de télécommunication :Cas réseau Fibre-Optique Algérie Télécom Béjaia , mini projet, Université ABDARRAHMANE MIRA, Bejaia, 2016.

[2] Alcouffe, A. Enjalbert, M. Muratet, G. Méthodes de résolution du problème de transport et de production d'une entreprise à établissements multiples en présence de coûts fixes. Revue française d'automatique, informatique, Recherche Opérationnelle, Tome 9 (1975).

[3] Amzaz Adil,Résolution des problèmes de la recherche opérationnelle :Problème de transport et problème de chargemen,MEMOIRE DE FIN D'ETUDES ,Licence Mathématiques et Applications,UNIVERSITE SIDI MOHAMED BEN ABDELLAH,2016

[4] Ben-Iken Mohamed,Problème de transport :Modélisation et résolution,MEMOIRE DE FIN D'ETUDES ,Licence Mathématiques et Applications,UNIVERSITE SIDI MOHAMED BEN ABDELLAH,2017

[5] B. Brahmi, Cours Master 2, Recherche Opérationnelle , (2017-2018).

[6] Bruno Garcia, cour Recherche Opérationnelle,2000

[7] Dodge Yadolah ,Optimisation appliquée ,Editeur : Springer Livre ,2005

[8] D.Muller .Introduction à la théorie des graphes. (2008).

[9] Frédéric Meunier.INTRODUCTION À LA RECHERCHE OPÉRATIONNELLE.Université Paris Est, CERMICS, Ecole des Ponts Paristech .2016

[10] F. Laburthe. Contraintes et algorithmes en optimisation combinatoire. Thèse de doctorat, Université Paris VII- Denis Diderot, Paris, 1998.

[11] J.cohen .Théorie des graphes et algorithmes. (oct 2006).

[12] Jin Y. Wang ,Operation Research I ,College of Management NCTU ,Fall 2008

[13] L.Wayne, Winston and Munirpallam Venkataramanan.Introduction to Mathematical Programming : Operations Research, Volume 1 4eme édition,2003

[14] L. Ntaimo ,Transportation and Assignment Problems , INEN420 TAMU 2005

[15] M. Gondran & M. Minoux. Graphes et algorithmes. Eyrolles, Paris, 1995.

[16] Rairo. Recherche Opérationnelle . tome 13 n°3.(1979)

[17] S. Skiena, The Algorithm Design Manual, Springer, 2ème édition, 2008.

[18] Yves De Smet , Bernard Fortz,Algorithmique 3 et Recherche Opérationnelle ,2013-2014

[19] Z.Kiraly,P.Kovacs,Efficient implementations of minimum-cost flow algorithms ,Acta Univ.Sapientaie ,Informatica ,2012

**Résume**

l'objective de ce travail de montrer l'importance de la recherche opérationnelle dans la résolution et l'optimisation par ces outil dont la théorie des graphe et la programmation linéaire dans les enjeux économiques , ce mémoire contribue également à montre l'importance des programme linaire et des graphe ( plus particulièrement les graphe orienté sans boucle orienté ). Dans la résolution de certains problèmes de la RO, et cela en cherchant quelques problèmes d'optimisation pour les quelles nous donnons quelque algorithmes de résolution pour chaque problème suivie d'une résolution, en faisant appel à un programme réalisé sous Dev-C++ .

mot clé : recherche opérationnelle, optimisation, programmation linéaire, théorie des graphe, graphe, Dev-C++

**Abstract**

The objective of this work to show the importance of the operational research in the resolution and the optimization by these tool among which the theory of the graph and the linear programming in them Stakes economics, this report also contributes to watch the importance of schedule linear and graph (more particularly graph directed without loop directed). In the resolution of some problems of the RO, and it by looking for some problems of optimization for the which we give about algorithms of resolution for every problem followed by a resolution, by appealing to a program realized under Dev-C ++.

Keyword : operational research, optimization, linear programming, theory of the graph, the graph, Dev-C ++