République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderahmane Mira de Béjaia
Faculté des Sciences Exactes
Département de Recherche Opérationnelle

Mémoire de fin de cycle

En vue de l'obtention du diplôme Master en Recherche Opérationnelle

Option :

Modélisation Mathématique et Évaluation de Performance des Réseaux

Thème

Modèles Mathématiques et algorithmes pour le problème de transport

Présenté par :

M^me HAMMACHI NOUHA

Soutenu le : 19/11/2020

Devant le jury composé de :

Présidente :	*M^me YOUNSI* Leila	MAA	U.A.Mira Béjaia.
Encadreur :	*M^r KABYL* Kamal	MCB	U.A.Mira Béjaia.
Examinateurs :	*M^r M'HAMDI* MOHAMMED SALAH	MCB	U.A.Mira Béjaia.

Année universitaire 2019/2020

Remerciements

Tout d'abord, je remercie **ALLAH** qui m'a donné la volonté, la motivation et le courage de mener à terme ce mémoire. Qui m'a ouvert les portes du savoir.

Je tiens à exprimer ma profonde gratitude et sincères remerciements à dr **kabyl Kamal**, qui m'a encadré, il était très généreux à travers son soutien, sa disponibilité, et ses conseils et orientations qui m'ont permis de mener ce travail à son terme.

Je tiens à remercier, également, Madame **YOUNSI Leila** pour l'honneur d'avoir accepter de présider le jury.

Je tiens à adresser mes remerciements à Monsieur **M'HAMDI Mohammed Salah** pour l'honneur d'avoir accepter d'examiner ce travail et faire partie du jury.

Je tiens à remercie, chaleureusement l'être qu'est la plus chére au monde « *ma mère* », ma sœur Nesrine, mes deux frères Adem et Sami, et mon mari Zinou pour leurs sacrifices et leurs conseils, sans

eux, je ne serai jamais arrivée à ce niveau. que dieu les garde.

Par ailleurs, mes remerciements s'adressent, également, à tous mes amis et mes collègues pour les moments et les souvenirs inoubliables qu'on a passé ensemble.

Enfin, je ne saurai oublier de remercier tous mes enseignants du département de recherche opérationnel, qui m'ont accompagnés et aidés à m'améliorer durant mon cursus de formation.

Dédicaces

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.

Mes deux chers frères Sami et Adem et ma chère sœur Nesrine et a mes nièces El hachemi et Anes.

A tous les membres de ma famille, petits et grands.

A mon mari Zino.

A tous ceux qui m'aiment et que j'aime.

TABLE DES MATIÈRES

TABLE DES FIGURES

INTRODUCTION GÉNÉRAL

Depuis les débuts de l'histoire, le transport s'avère important pour l'humanité. En effet, les moyens et les systèmes de transport facilitent le déplacement des personnes et des biens d'un endroit à l'autre. C'est

en 1941 que **Frank L.Hitchcock** a formulé pour la première fois le problème de transport, traité et étudié par **Koopmans** en 1947, **L.V Kantrovitch** et **M.K.Savorine** en 1949 et puis **G.B.Dantzig** en 1951. Le problème de transport à deux indices largement étudiés dans la littérature, est un modèle générique pour les problèmes d'affectation et peut être formulé comme un programme linéaire avec une structure spéciale des contraintes. Dans sa forme classique, le problème de transport consiste à minimiser le coût total de transport des marchandises disponibles en m origines pour n destinations[1].

Dans ce chapitre, nous allons présenter le problème de transport. Nous commençons par définir le problème de transport et les notions préliminaires de basse qu'on va utiliser dans le problème de transport. Par la suite, nous allons présenter la modélisation du problème

de transport sous forme d'un graphe et sous forme d'un programme linéaire pour le cas équilibré.

CHAPITRE 1

QUELQUES NOTIONS ET NOTATIONS DE BASE POUR L'OPTIMISATION

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**(nœuds), 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 v_1, v_2 (v_1, v_2), Dans le cas d'un arc $e = (v_1, v_2)$, $v1$ est appelé **extrémité initiale** de e et $v2$ est appelé **extrémité terminale** de e[2].

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

$$|\mathbf{V}(\mathbf{G})|$$

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

Si $e = (x, y)$ est une 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.[3]

Le graphe $\mathbf{H} = (\mathbf{V}, \mathbf{E}')$ est un graphe partiel de **G**, si $\mathbf{E}' \subseteq \mathbf{E}$. Autrement dit on obtient **H** en enlevant une ou plusieurs arêtes au graphe **G**.[1]

Un sous graphe engendré par $\mathbf{A} \subseteq \mathbf{V}$ est un graphe dont les sommets sont ceux de A et les arêtes sont celles ayant les deux extrémités dans **A**. on le note **GA**.[8]

La figure ci-dessous présente un graphe à 4 sommets et 4 arcs

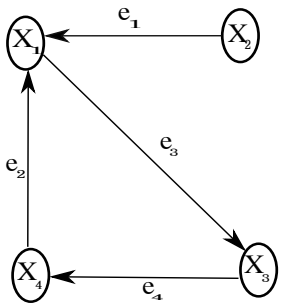


FIGURE 1.1 – Graphe à 4 sommet et 4 arcs

Quelques définitions des graphes simples et multiples

- **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.
- **Sous-graphe :**Si G est un graphe dont les sommets sont l'ensemble S et les arêtes sont l'ensemble A, et si S' est une partie de S, on appelle sous-graphe de S formé à partir de S' le graphe dont les sommets sont les éléments de S' et les arêtes sont les éléments de A reliant deux sommets de S'
- **Graphe partiel :**Soit $G = (S, A)$ un graphe. Le graphe $G' = (S, A')$ est un graphe partiel de G, si A' est inclus dans A.

1.2 Chaîne, 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, $\forall i \in \{1, 2, ..., k\}$, v_i et relié 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.

Chaîne simple

est une chaîne ne passant pas deux fois par une même arête, c'est-à-dire dont toutes les arêtes sont distinctes.

Chaîne élémentaire

est une chaîne ne passant pas deux fois par un même sommet, c'est-à-dire dont tous les sommets sont distincts.

Chemin

Un **chemin** dans un graphe orienté est un suite de sommets $\{v_0, v_1, ..., v_k\}$ tel que $\forall i \in \{1, 2, ..., k-1\}$, $\exists$ une arête de v_{i-1} vers v_i et autre arête de v_i vers v_{i+1}

• Un chemin p est **simple** si chaque arc du chemin a empruntée une seule fois.

Le graphe de la figure 1.2(b) présente un chemin de longueur 4 dans le graphe de la figure 1.2(a)

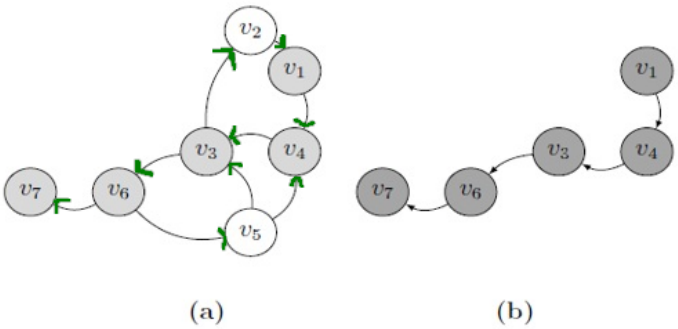


FIGURE 1.2 – Chemin

Cycle

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

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.

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.

Arbre

Un **arbre** est un graphe simple connexe acyclique (sans cycle). Également un arbre comporte exactement $|\mathbf{V}| - 1$ arêtes.

1.2.2 Composantes connexes

On appelle **composantes connexes** un ensemble de sommets, qui ont deux à deux la relation de connexité.

• 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[1.2].

Principe : Soit $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ un graphe orienté(digraphe). L'idée de cet algorithme est la suivante : pour un sommet quelconque $v \in \mathbf{G}$, 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.

Énoncé : $*$ Données : un graphe $\mathbf{G} = (\mathbf{V}, \mathbf{E})$

1. Initialisation $k \longleftarrow 0, A \longleftarrow \mathbf{V}$.
2. 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.

Poser $k = k + 1$ et C_k l'ensemble des sommets marqués.

Retirer de A les sommets de C_k et poser $A = A - C_k$.

Tester si $A = \emptyset$

— Si oui terminer ; aller à (3).

— Si non aller à (2) ;

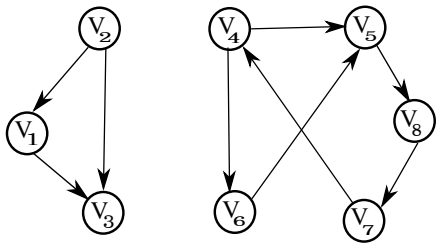
3. Le nombre de composantes connexes de $(\mathbf{G})$ est k.

Chaque ensemble C_i, $i = 1, .., k$ correspond aux sommets d'une composante connexe de $(\mathbf{G})$.

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

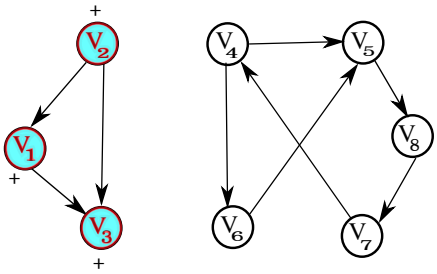
Le graphe de la figure suivant n'est pas connexe car il n'existe pas de chaîne reliant les sommets v_1 et v_6.

Soit le graphe $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ suivant :

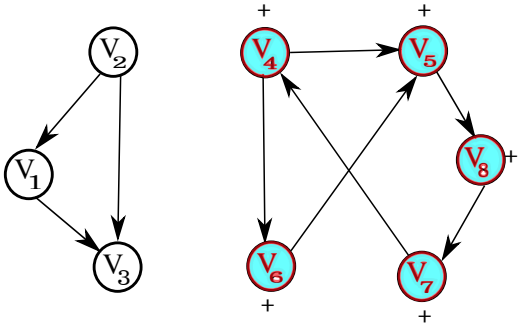


- On peut le vérifier par l'algorithme de marquage précédent pour déterminer les composantes connexes :

- Initialisation : $k = 0$. $A = \mathbf{V} = v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8$.
- **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.

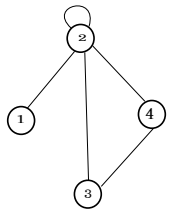


Soit $C_2 = \{v_4, v_5, v_6, v_7, v_8\}$ l'ensemble des sommets marqués. On retirons de A les sommets de C_2, on obtient : $A = \emptyset$ **terminer**.

1.3 Représentations matricielle des graphes

1.3.1 Matrice d'adjacence

Soit $\mathbf{G}(V, E)$ un graphe non-orienté qui possède n sommets numérotés de 1 à n. On appelle **matrice adjacence** du graphe la matrice $A = (a_{ij})$ où a_{ij} est le nombre d'arêtes joignant le sommet i au sommet j. Le graphe de la figure 1.3a est la matrice d'adjacence associée au graphe de la figure 1.3*b*



(a)

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

(b)

FIGURE 1.3 – Graphe à 4 sommets et 5 arêtes et la matrice adjacence associée

Pour les matrices d'adjacence d'un graphe non-orienté. Le résultat principal est le théorème suivant :

Théorème. Soit $\mathbf{G}(V, E)$ un graphe non-orienté de matrice d'adjacence A. Le nombre de chaînes de longueur n joignant le sommet i au sommet j est donné par le terme d'indice i, j de la matrice[11].

• Dans le cas d'un graphe $\mathbf{G}$ non orienté, la matrice d'adjacence A est symétrique[11].

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 $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ 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 :

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

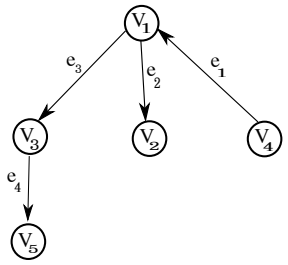
- La matrice d'incidence d'un graphe **G** non orienté sans boucle définie par :

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

- 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 1.4 est donnée par :



(a)

$$M = \begin{array}{c} \begin{matrix} e_1 & e_2 & e_3 & e_4 \end{matrix} \\ \begin{pmatrix} -1 & 1 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \end{array}$$

(b)

FIGURE 1.4 – Graphe à 4 sommets et 5 arcs et la matrice adjacence associée

1.3.3 Flux et Flots

Flux

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

Flot

Un flot φ est déterminé par la donnée du flux pour tout arc du réseau de transport. La valeur d'un flot $\mathbf{V}(\varphi)$ est par définition, la somme des flux partant de la source $x_(1)$ ($\mathbf{V}(\phi)$ est aussi égale à la somme des flux des arcs arrivant sur le puits x_p)

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

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

Où le cocycle

$\omega^+(i)$ est l'ensemble des arcs sortant en i.

$\omega^-(i)$ est l'ensemble des arcs entrant en i.

1.3.4 Couplage

Étant donné un graphe non orienté, un couplage, ou "matching" en anglais, est un sous-ensemble d'arêtes disjointes 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[14].

1.3.5 Chaîne améliorante

μ est une chaîne améliorante (ou augmentant) de s à t pour un flot ϕ admissible donné si :

— $\phi_{ij} < c_{ij}$ pour tout arc (i, j) de μ dans le bon sens (de s vers t).

— $\phi_{ij} > 0$ pour tout arc (i, j) de μ dans le mauvais sens (de s vers t)[5].

Quelques définitions des graphes particuliers

On distingue plusieurs types de graphes :

- **Graphe complet :** Un graphe $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ est dit complet si tous les sommets de G sont deux à deux adjacents, le graphe complet à n sommet est notéK_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 $\mathbf{G}(v) = (\mathbf{V}, \mathbf{E}(v))$. Soit (i, j) un arc de $\mathbf{G}$ 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 $\mathbf{G}(x)$ va donc proposer deux arcs mettant en avant ces deux possibilités :

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

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

- **Graphe biparti :** Un graphe $\mathbf{G}$ est dit **biparti** si l'ensemble de ses sommets peut-être partitionné 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 $\mathbf{G} = (V_1, V_2, \mathbf{E})$.[8]
  La figure 1.5 présente un graphe biparti

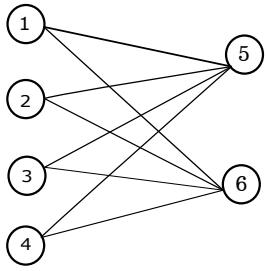


FIGURE 1.5 – Graphe biparti complet $k(n)$

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 nombre réels.

Une des méthodes les plus connues pour résoudre des programmes linéaires en nombre réels est la méthode du Simplex.

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

1.4.1 Forme générale d'un programme linéaire

La formulation mathématique d'un problème linéaire est donnée comme suit :

$$\begin{cases} z = \sum_{j=1}^{n} c_j x_j. & (1.1) \\ \forall i = 1, ..., m : \sum_{j=1}^{n} a_{ij} x_j \leq, = \text{ou} \geq b_i. & (1.2) \\ \forall j = 1, ..., n : x_{ij} \geq 0. & (1.3) \end{cases}$$

(1.1) : fonction objective à minimiser où à maximiser.
(1.2) : m contraintes linéaires.
(1.3) : Contraintes de positivité.

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 $a_{ij} avec \quad 1 \leq i \leq m \quad 1 \leq j \leq n$. Comme forme matricielles, nous avons :

- **Forme canonique :** la forme canonique d'un programme linéaire est donner par :

$$\begin{cases} maxz = \mathrm{C}^T \ \mathrm{x} \\ Ax \leq b \\ x \geq 0 \end{cases}$$

- **Forme standard :** la forme standard d'un programme linéaire est donner par :

$$\begin{cases} maxz = \mathrm{C}^T \ \mathrm{x} \\ Ax = b \\ x \geq 0 \end{cases}$$

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 dans le pire des cas. 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(.)[12].

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 [13].

Les deux précisions sur le caractère de cette mesure importante :

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

L'objectif de ce chapitre est 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 $\{X_1, ..., X_m\}$ et n destinations $\{Y_1, ..., Y_n\}$. 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 quantite 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.[9]

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 (i) (disponibilité) et b_j désigne le besoin d'une destination (j)(demande).

Les contraintes sont :

— La disponibilité à chaque source doit être épuisée :

$$\sum_{j}^{n} x_{ij} = a_i, i \in \{1, ..., m\}.$$

— La demande à chaque destination doit être satisfaite :

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

— La non négativité des quantités :

$$x_{ij} \geq 0 \forall i \in \{1, ..., m\}; \forall j \in \{1, ..., n\}.$$

2.2.4 Formulation mathématique

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

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 vers la $i^{ème}$ source.
b_j : Demande vers la $j^{ème}$ destination.
c_{ij} : Coût unitaire de transport de la $i^{ème}$ source vers la $j^{ème}$ destination.
x_{ij} : Quantité transportée de la $i^{ème}$ source vers la $j^{ème}$ destination[11].

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

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

[11].

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]

le tableau 2.1 présente un tableau de problème de transport :

i \ j	D_1	D_2	D_j	D_n	disponibilité
O_1	X_{11}	X_{12}	X_{13}	X_{1n}	a_1
O_2	X_{21}	X_{22}	X_{23}	X_{2n}	a_2
O_i	X_{i1}	X_{i2}	X_{i3}	X_{in}	a_i
O_m	X_{m1}	X_{m2}	X_{m3}	X_{mn}	a_m
demande	b_1	b_2	b_i	b_n	

TABLE 2.1 – tableau 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 flèches 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" Ceci est représenté dans la figure 2.1

Dans la figure 2.1 , il y a $x_1 ... x_m$ sources et $y_1 ... y_n$ destinations. Les arcs montrent des flux de transport de source vers destination. Chaque destination est liée à chaque source par un arc, le nombre $(c_{11} ... c_{mn})$ 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 représenter des populations, des clients,...etc.

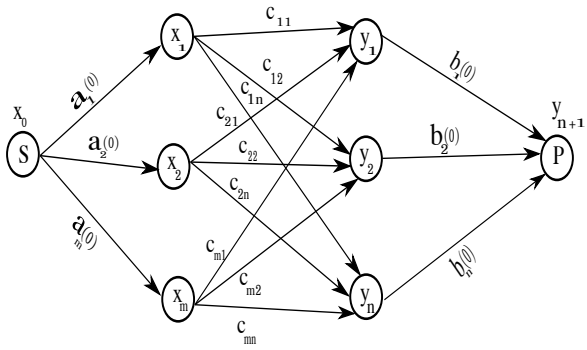


FIGURE 2.1 – Réseau de transport

• **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} $(i = 1, ..., m)$ 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 $x_{m+1}(i = 1, ..., m)$ dont disponibilité :

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

CHAPITRE 3

LES MÉTHODES DE RÉSOLUTION D'UN 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 les méthodes de résolution. Nous commençons par les méthodes qui déterminent une solution de base réalisable. Par la suite nous allons faire une comparaison entre ces méthodes, et nous terminons par les méthodes d'amélioration tel que Stripping Stone et multiplicateurs.

3.1 Structure de la résolution de problème de transport

Considérons un problème de transport impliquant m origines et n destinations. Étant 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

Pour un problème de transport sous forme équilibré, une solution de base réalisable aura au plus $m + n - 1$ variables de base positives.

3.1.2 Solution de base dégénérée

On dit qu'on a une solution de base dégénérée lorsqu'une ou plusieurs variables de base sont nulles, c'est-à-dire lorsque toutes les contraintes sont satisfaites et qu'il y a moins de $(m + n - 1)$ cellules positives (> 0) dans la solution considérée.

3.1.3 Méthode pour déterminer une solution de base réalisable

Pour déterminer une solution de base réalisable d'un problème de transport, on utilise l'une des méthodes suivantes : la méthode Coin Nord-Ouest, la méthode des Coût Minimum, la méthode de Ballas

Hammer (la méthode de Vogel) ...etc.

3.1.3.1 La méthode des Coûts Minimum

[7] Cette méthode elle a l'avantage de fournir rapidement et aisément une solution de base, son principe est le suivant :

1. Sélectionner la cellule de coût minimum ;
2. Allouer le plus possible à la cellule courante et ajuster l'offre et la demande ;
3. On élimine du tableau la ligne ou la colonne saturée ;
4. Sélectionner la cellule de coût minimum ayant une demande et une offre non nul ;
5. Répéter jusqu'au moment où toute l'offre est allouée.

Soit le problème de transport donné par le tableau suivant :

Origines \ Destinations	D_1	D_2	D_3	Offre
O_1	25	17	16	35
O_2	24	18	14	550
Demande	300	300	300	

TABLE 3.1 – tableau de donné

La méthode de coût minimum nous a donné la solution suivant :

	D_1	D_2	D_3	Offre
O_1	25 Min{50,50}=50	17 Min{350,300}=300	16 0	350-300=50
O_2	24 Min{250,300}=250	18 0	14 Min{550,300}=300	550- 300=250
Demande	300 300-250=50 50-50=0	300 300-300=0	300 300-300=0	

TABLE 3.2 – tableau représente la solution réalisable initial

La solution réalisable de base initiale est :

$$2 + 3 - 1 = 4.$$

Le coût de transport total est :

$$Z = 25(50) + 17(300) + 24(250) + 14(300) = 16550.$$

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

3.1.3.2 Méthode du COIN NORD-OUEST

Cette méthode permet d'obtenir une solution de base réalisable, très simple quant à son implémentation, son principe est le suivant :[7]

Étape 1 : On attribue la quantité minimale entre la disponibilité et la demande á la variable représentant la cellule se situant au coin nord-ouest du tableau "Disponibilité-Demande". Aller à l'étape2.

Étape 2 : :On élimine du tableau la ligne ou la colonne saturée,on diminue de $\mathcal{X}_{11}$ la ligne ou colonne non saturée. Mettre à jour

les demandes et les disponibilités. Aller à l'etape 3.

Étape 3 : On répète l'étape1 et l'étape2 sur le reste du tableau jusqu'á épuisement de la disponibilité et de la demande.

Remarque

Dans le cas où une augmentation sature la ligne et la colonne en même temps, on choisit d'éliminer seulement soit la ligne, soit la colonne.

La dernière case sature á la fois sa ligne et sa colonne.

La solution de base réalisable obtenue par la méthode de Vogel et donné par le tableau suivant :

	D_1	D_2	D_3	Offre
O_1	25 Min{50,50}=50	17 Min{350,300}=300	16 0	350-300=50 50-50=0
O_2	24 Min{250,300}=250	18 0	14 Min{550,300}=300	550 550-250=300 250-250=0
Demande	300 300-250=50 50-50=0	300 300-300=0	300 300-300=0	900 900

TABLE 3.3

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

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

On voit tout de suite que la solution trouvée n'est pas optimale. En effet, le coût total de ce plan de transport est plus élevé que celui

trouvé à l'aide de la méthode de la matrice minimale, qui était de 16550.

Faiblesse de la méthode de Coin Nord-Ouest

La méthode du CNO donne bien une solution de base réalisable, mais elle peut être très loin de l'optimal.
La méthode du CNO a tendance á donner des solutions de base réalisables dégénérées (avec des variables de base à zéro).
Elle ne tient pas compte du tout du coût.

3.1.3.3 La méthode de Vogel

Appelée encore méthode des regrets, ou la différence maximale, ou de Balas-Hammer, elle Procède comme suite :

1. On choisit en ligne et en colonne les deux chiffres les plus bas et on calcule leur différence,
2. On prend la différence la plus élevée. Si on a deux ou plus des différences égales alors on prend une au hasard,
3. Sur la ligne ou la colonne à laquelle appartient cette différence, on prend le coût le plus bas,
4. Dans la case à laquelle appartient, allouer le plus possible de quantité à la cellule courante tout en respectant les contraintes de l'offre et la demande. On barre la ligne ou (et) la colonne saturée et on recommence le processus de 1 à 4 jusqu'a, le tableau soit barré

	1	2	3	Offre
1	25 Min{50,300}=50	17 Min{350,300}=300	16 0	350 350-300=50 50-50=0
2	24 Min{250,250}=250	18 0	14 Min{550,300}=300	550 550-300=250 250-250=0
Demande	300 300-50=250 250-250=0	300 300-300=0 250-250=0	300 300-300=0 300-300=0	900

TABLE 3.4

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

$$Z = 25(50) + 17(300) + 24(250) + 14(300) = 16550.$$

On obtient par hasard la même solution réalisable de base initiale qu'avec la première méthode proposée.

3.1.3.4 La comparaison entre les trois méthodes

On remarque que les trois méthodes exposées ci-dessus sont équivalents. Cependant, les résultats obtenus ne le sont pas. La méthode du Coin-Nord-Ouest est celle qui demande le moins de temps pour trouver la solution réalisable de base initiale, car son critère de choix à la première étape est très simple. En revanche, elle ne donne pas en général une bonne solution de départ, le coût total du plan de transport généré étant assez éloigné de l'optimum.

3.2 Méthode de Stepping-Stone

Avant de démarrer l'illustration de cette méthode nous allons donner quelques notions :

3.2.1 Dégénérescence

La méthode Stepping-Stone traite le cas dégénéré et le cas non dégénérée. La dégénérescence peut apparaitre soit dans la première solution de base, soit au cours du processus des itérations menant à la solution optimale.
Pour résoudre le problème de dégénérescence, il faut introduire une perturbation infiniment très petite ($\varepsilon > 0$) dans la case associée à une variable hors base.

3.2.2 Notion de boucle

Une boucle est une séquence de 4 cellules au moins, telle que :

1. Deux cellules consécutives sont dans la même ligne ou même colonne,
2. Toute suite de trois cellules consécutives n'est jamais dans la même ligne ou colonne,
3. La dernière cellule dans la séquence a une ligne ou une colonne en commun avec la première.

Théorème

Soit un problème de transport avec m producteurs et n consommateurs. Les cellules qui correspondent à un ensemble de $m + n - 1$ variables ne contiennent aucune boucle si et seulement si les $m + n - 1$ variables forment une solution de base autrement dite si la solution de base n'est pas dégénérée.[5]

3.2.3 Le coût réduit de la fonction objectif[3]

Soit :

Z_1 : la valeur de la fonction objectif pour la solution de base initial.

Z_2 : la valeur de la fonction objectif de la nouvelle solution de base.

Alors

$Z_2 = Z_1 + \theta\delta_{ij}$, où δ_{ij} est la rédaction ou l'augmentation des coûts pour une unité de θ sur le parcours tracé.Puisque $\theta > 0$,il faut que $\delta_{ij} < 0$ pour diminuer la valeur de fonction objectif.

3.2.4 Algoritme de Stepping-Stone

Cet algorithme est essentiellement est une adaptation de la méthode du simplexe, qui utilise la notion de boucle pour effectuer des pivots directement sur le tableau de transport [7].

Elle consiste à trouver un plan faisable puis à construire une suite de programmes de base améliorant constamment la fonction économique et donc conduisant à l'optimum, d'où le nom de la méthode. Pour la détermination d'une solution de départ. On peut utiliser l'une des trois méthodes suivantes :

- Méthode de Coin Nord Ouest ;
- Méthode de Coût Minimum ;
- Méthode de Vogel.

a. Critères de l'algorithme de Danzing

- Critère d'entrée d'une variable :la variable à entrer la base est celle qui possède le minimum des δ_{ij} où $\delta_{ij} < 0$.
- Critère de sortie :la variable qui sort de la base est celle correspond à $\theta = Min\mathcal{X}_{ij}$ sur le parcours dont on a soustrait θ.

— Critère de l'optimalité :la solution est optimal lorsque tous les $\delta_{ij} > 0$. Si la solution optimale à un ou plusieurs $\delta_{ij} = 0$, alors il existe plusieurs solutions optimales.La solution est unique si $\delta_{ij} > 0$.

b. Enocé de l'algorithme

L'algorithme se résume en deux étapes essentielles

Étape1 : trouver une solution de base avec l'un des méthodes vus précédemment ;

Étape2 : amélioration de la solution de base.

1. Calculer les coûts marginaux notés δ_{ij} pour chaque liaison non affectée ;
2. Si tous les δ_{ij} sont positifs ou nuls $\Longrightarrow$ Fin.La solution obtenue est une solution de base.Si non,prendre le cycle de substitution μ associé au δ_{ij} le plus petit ;
3. Retour en 1.

CHAPITRE 4

RÉSOLUTION DE QUELQUES PROBLÈME DE TRANSPORT

4.1 Réseau de flot

En théorie des graphes, un **réseau de flot** (aussi appelé **réseau de transport**) est un graphe orienté 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. Un graphe connexe orienté est souvent appelé **réseau** en recherche opérationnelle. 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.

4.2 Algorithme de Ford Fulkerson

Étape 1 : *Initialisation :*

A partir d'un flot réalisable quelconque ($f = 0$) marquer la source d'un signe $(., +\infty)$, tous les autres sommets sont non marques et non examinés.

Étape 2 : *Éxamination d'un sommet :*

pour tout sommet j tel que :

$a = (i, j) \in A$ avec $\{f(a) < c(a)\}$ avec i marqué et j non marqué. faire marquer j d'un signe $\{+i, min[\sigma, c(a) - f(a)]\}$.

Pour tout sommet j tel que :

$a = (j, i) \in A$ avec $f(a) > 0$, i marqué et j non marqué faire marqué j d'un signe $\{-i, min[\sigma, f(a)]\}$, le sommet j maintenant est examiné.

Étape 3 : *Test d'optimalité du flot courant :*

si le puit p est marqué aller 'a l'étape 4,

sinon voir s'il $\exists$ d'autres sommets marqués non examinés, retourner à l'étape 2, sinon il $\exists$ aucun sommet marqué non examiner alors le flot courant est maximal.

Étape 4 : *Actualisation des flots :*

1. Identifier la chaîne augmentent joignant s à p.
2. Actualiser le flot courant le long de la chaîne augmentent.
3. Efface toute marque et retourner en étape 1.

Exemple 1

Une entreprise s'intéresse à acheminer un type de produit de ses usines (lieux de production) vers ses clients.

Elle doit se plier aux contraintes de capacité du système de transport. On désigne les sommets de réseau par des nombres, par chaque arc (i, j) on désignera par $C_{i,j}$ sa capacité (quantité maximal pouvant

être acheminer (traverser l'arc)), et par $x_{i,j}$ la quantité de produit transportée via l'arc (i, j). Considérons le réseau suivants :

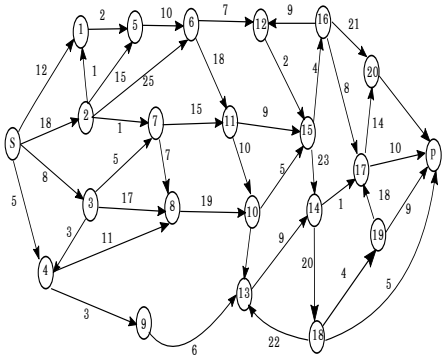


FIGURE 4.1 – le premier exemple d'un flot

Itération 1 :

✓ $C_1 = (\{S, 1\}\{1, 5\}\{5, 6\}\{6, 12\}\{12, 15\}\{15, 16\}\{16, 20\}\{20, P\})$

✓ $\sigma(C_1) = 2$

✓ $f(a_r) = 2$

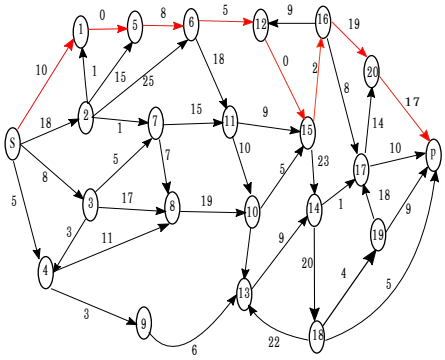


FIGURE 4.2 – 1 ere itération

Itération 2 :

✓ $C_2 = (\{S, 2\}\{2, 5\}\{5, 6\}\{6, 11\}\{11, 15\}\{15, 16\}\{16, 20\}\{20, P\})$

✓ $\sigma(C_2) = 2$

✓ $f(a_r) = 4$

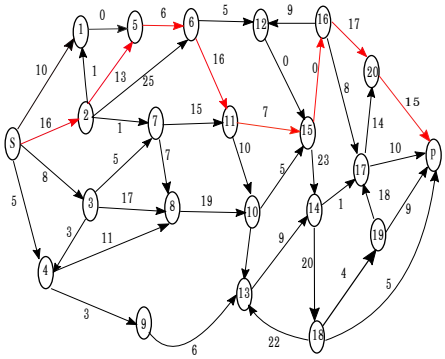


FIGURE 4.3 – 2 éme itération

Itération 3 :

✓ $C_3 = (\{S, 2\}\{2, 7\}\{7, 11\}\{11, 15\}\{15, 14\}\{14, 17\}\{17, P\})$

✓ $\sigma(C_3) = 2$

✓ $f(a_r) = 2$

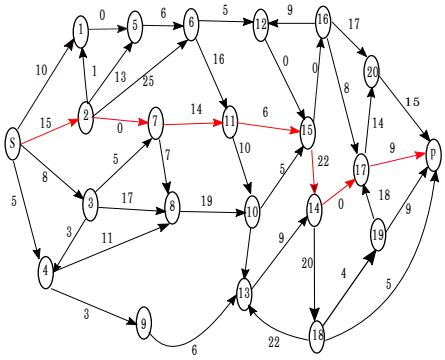


FIGURE 4.4 – 3 éme itération

Itération 4 :

✓ $C_4 = (\{S, 2\}\{2, 6\}\{6, 11\}\{11, 15\}\{15, 14\}\{14, 18\}\{18, P\})$

✓ $\sigma(C_4) = 4$

✓ $f(a_r) = 9$

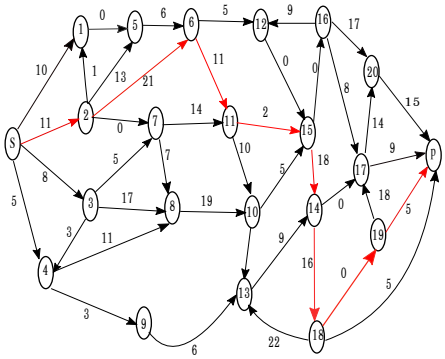


FIGURE 4.5 – 4 éme itération

Itération 5 :

✓ $C_5 = (\{S, 3\}\{3, 8\}\{8, 10\}\{10, 15\}\{15, 14\}\{14, 18\}\{18, P\})$

✓ $\sigma(C_5) = 5$

✓ $f(a_r) = 14$

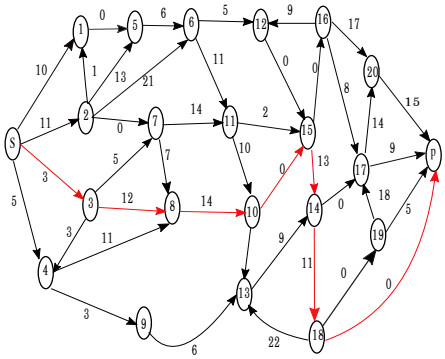


FIGURE 4.6 – 5 éme itération

4.3 Problème

— Une firme automobile a trois usines à Los Angeles, Detroit et New Orleans, et deux centres de distribution à Denver et Miami.

— Les capacités des trois usines sont de 1000, 1500 et 1200 respectivement, et les demandes aux centres de distribution sont de 2300 et 1400 voitures.

— Coûts :

	Denver	Miami
Los Angeles	80	215
Detroit	100	108
New Orleans	102	68

Formulation de problème le modèle mathématique correspondant au problème ci-dessous est donné comme suit :

$$minz = 80x_{11} + 215x_{12} + 100x_{21} + 108x_{22} + 108x_{31} + 68x_{32}$$

s.c.

$$\begin{aligned}
x_{11} \quad + x_{12} &= 1000 \\
x_{21} \quad + x_{22} &= 1500 \\
x_{31} \quad + x_{32} &= 1200 \\
x_{11} \quad + x_{21} &
\end{aligned}$$

Représentation tableau

	Denver	Miami	Offre
Los Angeles	80 1000	215	1000
Detroit	100 1300	108 200	1500
New Orleans	102	68 1200	1200
Demande	2300	1400	

Problèmes non balancés

— Si l'offre n'est pas égale à la demande : modèle non balancé.

— Introduction d'une source ou destination artificielle.

	Denver	Miami	Offre
Los Angeles	80 1000	215	1000
Detroit	100 1300	108	1300
New Orleans	102	68 1200	1200
Artif.	0	0 200	200
Demande	2300	1400	

4.4 Algorithme pour le problème de transport

Le problème de transport représente par le tableau suivant :

Algorithme pour le problème de transport

1. Détermination d'une solution de base admissible.
2. Détermination de la variable entrant en base.
3. Détermination de la variable sortant de base.

— Si l'offre n'est pas égale à la demande : modèle non balancé.
— Introduction d'une source ou destination artificielle.

	1	2	3	4	Offre
1	10	2	20	11	15
2	12	7	9	20	25
3	4	14	16	18	10
Demande	5	15	15	15	

Détermination d'une solution de base admissible

— Heuristiques "gloutonnes", pas besoin de méthode des deux phases.
— Variantes :

1. Coin Nord-Ouest
2. Méthode des moindres coûts

Coin Nord-Ouest

Partir du coin supérieur gauche du tableau.

1. allouer le plus possible à la cellule courante et ajuster l'offre et la demande ;
2. se déplacer d'une cellule vers la droite (demande nulle) ou le bas (offre nulle) ;
3. répéter jusqu'au moment où toute l'offre est allouée.

	1	2	3	4	Offre
1	10 5	2 10	20	11	15
2	12	7 5	9 15	20 5	25
3	4	14	16	18 10	10
Demande	5	15	15	15	

Coût :520

4.5 Présentation de logiciel Matlab

Le langage Matlab (MATrixe LABoratory) est un logiciel commercial de calcul interactif. Il permet de réaliser des simulations numériques basées sur des algorithmes d'analyse numérique. Il peut donc être utiliser pour la résolution approchés d'équations différentielles, d'équations aux dérivées partielles ou des systèmes linéaires. Logiciel MATLAB est donné par 4.7

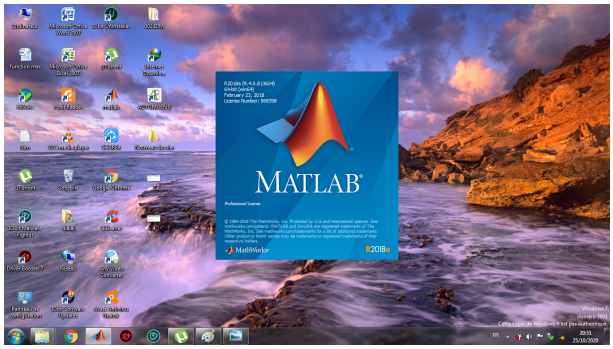


FIGURE 4.7 – logiciel MATLAB

L'interface de logiciel MATLAB est donnée par la figure 4.8

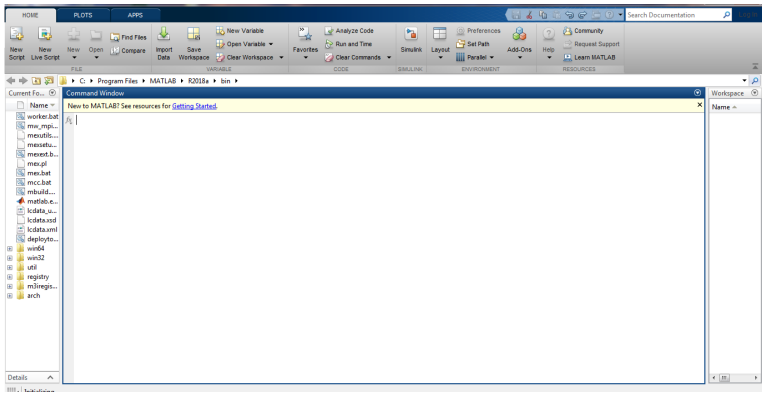


FIGURE 4.8 – Interface

Considérons le réseau de la matrice d'adjacence suivante qui représente la capacité de transport entre les nœuds i et j :

$$m = \begin{pmatrix} 0 & 16 & 13 & 0 & 0 & 0 \\ 0 & 0 & 10 & 12 & 0 & 0 \\ 0 & 4 & 0 & 9 & 14 & 0 \\ 0 & 0 & 0 & 0 & 0 & 20 \\ 0 & 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

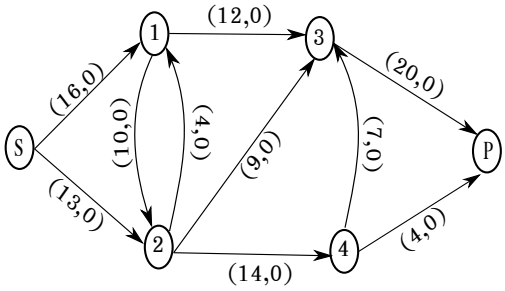


FIGURE 4.9 – Un réseau de flot

On va déterminer la valeur de flot maximal circulant entre les nœuds

S et P et la coupe minimale en utilisant l'algorithme de Ford-Fulkerson sous logiciel MATLAB voici l'implémentation dans la figure ci-dessous :

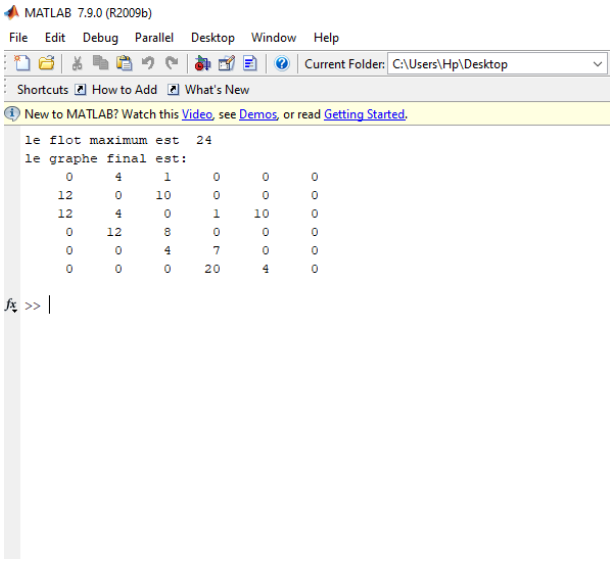


FIGURE 4.10 – implémentation sous logiciel MATLAB de algorithme Ford fulkursan de l'exemple

On utilisant l'algorithme de Ford-Fulkursen dans l'exemple pour déterminer la valeur de flot maximal circulant entre les nœuds S et P et la coupe minimale voici les itérations de l'algorithme ci-dessous :

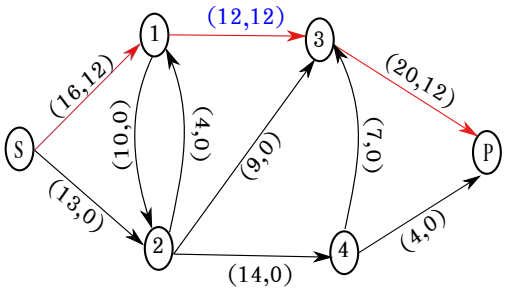


FIGURE 4.11 – premier itération

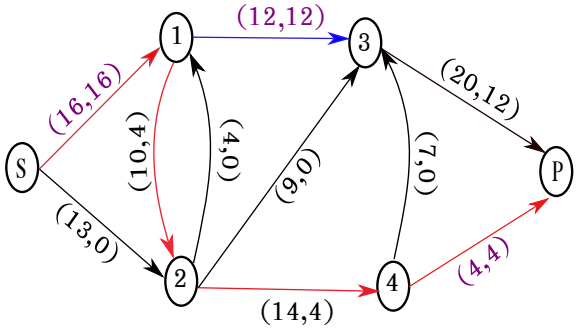


FIGURE 4.12 – deuxième itération

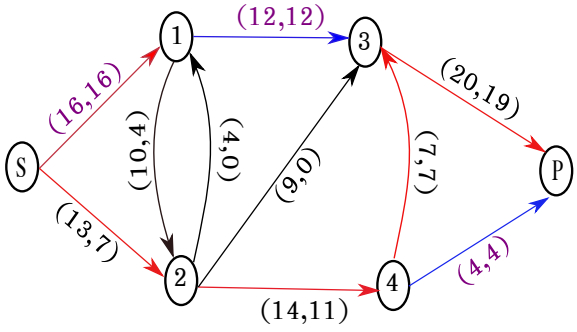


FIGURE 4.13 – troisième itération

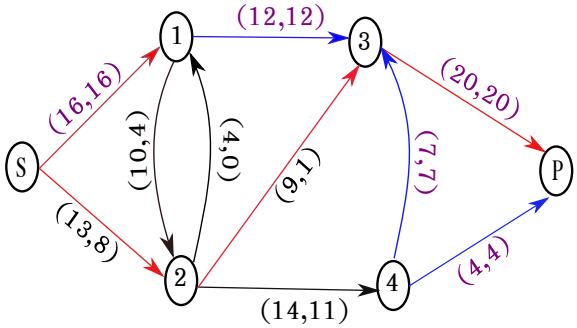


FIGURE 4.14 – quatrième itération

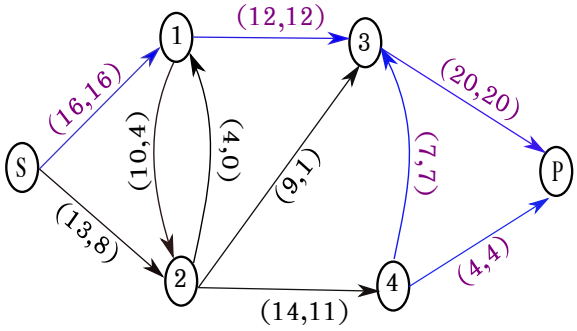


FIGURE 4.15 – cinquième itération et le graphe final

TABLE DES MATIÈRES

CONCLUSION GÉNÉRALE

Cette étude m'a donné l'opportunité de me familiariser au diamine de la recherche opérationnelle, ce domaine qui est la discipline des méthodes scientifiques pour aider à mieux décider et traiter les problèmes stratégique et économiques. Le problème de transport est l'un de ces problèmes classique 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 ce rapport, on s'est intéressé d'avantage à la modélisation et la résolution de problème de transport équilibré par des différentes méthodes qui nous permettons d'obtenir une solution de base réalisable (Nord-ouest, coût minimum, approximation de vogel). Puis nous avons essayer de faire une comparaison entre ces méthodes. Ensuite nous avons essayé d'explique l'optimisation d'un tellement solution de base initiale par la méthode stepping stone. Ainsi nous avons implémenter la méthode de Ford fulkurson pour résolution du problème de flot maximum.

Dans un premier lieu nous avons présenter quelques notion de base. Puis quelques méthodes d'optimisation que nous avons appliqué à la différent types de problèmes et enfin nous avons présenté le logiciel MATLAB qui nous a permet de résoudre via une application les différents problème de transport.

À la fin je peux dire que le travail représente une base de départ pour résoudre les problèmes général de transport.

BIBLIOGRAPHIE

[1] OURBIH,M. Cours sur la programmation linéaire, 2013.

[2] Mamadou,B. "Nouvelle Méthode de Résolution des Problèmes Linéaires à Variables Bornées par Décomposition", thèse de doctorat, mathématiques appliquées, université de Dakar, 2004.

[3] Professeur Yadolah "Optimisation Appliquée", université de Neuchâtel, 2002.

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

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

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

[7] Frédéric Meunier.INTRODUCTION À LA RECHERCHE OPÉRATION- NELLE.Université Paris Est, CERMICS, Ecole des Ponts Paristech .2016

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

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

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

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

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

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

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

Résumé

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 graphes 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 graphes orientés sans boucle orienté). Dans la résolution de certains problèmes de la RO, et cela en cherchant quelques problèmes d'optimisation pour lesquelles nous donnons quelques algorithmes de résolution pour chaque problème suivi d'une résolution, en faisant appel à un programme réalisé sous logiciel MATLAB. Mot clé : recherche opérationnelle, optimisation, programmation linéaire, théorie des graphe, graphe, logiciel MATLAB.

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 MATLAB. Keyword : operational research, optimization, linear programming, theory of the graph, the graph, MATLAB