République Algérienne Démocratique et Populaire
Ministère De l'Enseignement Supérieur Et De La Recherche Scientifique
Université Mouloud Mammeri De TIZI-OUZOU
Faculté des Sciences
Département des Mathématiques

Mémoire de fin de cycle

En vue de l'obtention du Diplôme de Master en mathématiques appliqué
Option : Recherche Operationnelle

Thème

Modélisation, Résolution et Implémentation du Problème de Transport

Réalisé par :
KACIMI Imilia

Encadré par :
Mr KASDI Kamel

Devant le jury

Président : Mr. BELHADJ Abdelaziz
Examinateur : Mr. MERAKEB Abdelkader

2023-2024

Dédicaces

Avec un cœur déterminé et un esprit ouvert, les études deviennent le chemin vers la réussite et l'épanouissement.

Je dédie ce modeste travail :
À ***mes parents****, qui m'ont donné la vie et ont été la source de mes efforts.*
À ma mère, symbole de tendresse, qui s'est sacrifiée pour mon bonheur, et à mon père, qui m'a inculqué le désir d'apprendre, la discipline, les valeurs de la réussite et le respect d'autrui.
Merci pour votre amour inconditionnel et votre soutien indéfectible.
À ***ma sœur*** *à laquelle je souhaite un brillant avenir et une vie pleine de bonheur.*
À tous mes petits cousins.
À toute ma famille.
À toutes mes amies.
Avec toute ma reconnaissance et mon amour,

Imilia

Remerciement

Louange à Dieu, le Miséricordieux. Sans Sa grâce, rien de tout cela n'aurait été possible.
Je tient à remercier ***Mr.KASDI*** *pour l'honneur qu'il m'a fait en acceptant de m'encadrer.*
Mes remerciements s'adressent également aux honorables memebres de jury qui ont accepté d'évaluer ce travail.
Je tient également à exprimer ma profonde gratitude à l'ensemble des enseignants du département mathématiques qui ont contribué à ma formation.
Et je tient à remercier toutes les personnes qui m'ont encourager et m'ont soutenu de prés ou de loin durant toutes ces années d'étude.
J'espère que ce mémoire servira de guide pour les promotions à venir.
Merci à tous.

Résumé

Le problème de transport est un problème d'optimisation linéaire visant à déterminer le moyen le plus économique de transporter des marchandises d'un ensemble de fournisseurs à un ensemble de clients.
Ce travail présente les étapes pour trouver une solution à ce problème. Pour obtenir une solution initiale, on utilise des méthodes telles que le Coin Nord-Ouest, qui commence par le coin supérieur gauche de la matrice des coûts, le Coût Minimum, qui privilégie les coûts les plus bas, et la méthode de Vogel, qui se base sur les pénalités de coûts.
Pour optimiser cette solution initiale, on applique des méthodes comme MODI, qui utilise des multiplicateurs pour ajuster les coûts réduits, et la méthode de Stepping-stone, qui identifie les chemins de cycle pour réduire les coûts de transport.

Abstract

The transportation problem is a linear optimization problem aimed at determining the most economical way to transport goods from a set of suppliers to a set of customers. This work presents the steps to find a solution to this problem. To obtain an initial solution, methods such as the North-West Corner, which starts at the top-left corner of the cost matrix, the Least Cost Method, which prioritizes the lowest costs, and Vogel's Approximation Method, which is based on cost penalties, are used.
To optimize this initial solution, methods like MODI, which uses multipliers to adjust reduced costs, and the Stepping Stone Method, which identifies cycle paths to reduce transportation costs, are applied.

Table des matières

Table des figures

Liste des tableaux

Introduction

Le développement technologique, soumet l'homme aux contraintes d'un système de relation économique de plus en plus difficile. On remarque qu'il devient de plus en plus nécessaire de prendre en compte un nombre croissant de nouveaux éléments lors des prises de décision concernant une action donnée telle que l'organisation d'une production, un réseau de transport,..., etc.
Ces prises de décision deviennent l'objet d'études approfondies qui requièrent l'assistance d'outils mathématiques pour être menées à bien. C'est ainsi que s'est développé un domaine des mathématiques appelé ***Recherche Opérationnelles*** en abrégé **RO**.

La recherche opérationnelle est une discipline interdisciplinaire qui combine des concepts et des techniques provenant des mathématiques (programmation linéaire, théorie des graphes, programmation dynamique,...), de l'informatique, de l'ingénierie et de l'économie visant à optimiser les processus décisionnels dans des environnements complexes.

Si la recherche opérationnelle continue aujourd'hui de jouer un rôle crucial dans la prise de décision et la gestion des opérations dans de nombreux domaines, elle trouve ses origines dans les besoins stratégiques et tactiques des services militaires, notamment lors de la Seconde Guerre mondiale.
À cette époque, l'ampleur des opérations militaires nécessitait une allocation efficace des ressources limitées disponibles. Les militaires britanniques, puis américains, ont donc mobilisé un grand nombre de scientifiques pour répondre à ces défis logistiques et tactiques. Les principes et techniques de la RO ont été utilisées pour optimiser le déploiement des troupes, la gestion des approvisionnements, la planification des itinéraires, et bien d'autres aspects cruciaux des opérations militaires.

Parmi les problèmes les plus courants en recherche opérationnelle, on trouve le problème de transport, qui constitue le sujet de notre projet de fin d'étude.
Ce problème est un modèle essentiel de programmation linéaire qui apparaît dans de nombreux contextes différents. Il a suscité une attention particulière en recherche opérationnelle en raison de sa fréquence d'apparition dans les applications pratiques et de la simplicité des méthodes développées pour le résoudre.
En effet, le problème de transport est souvent considéré comme l'un des problèmes de programmation linéaire les plus importants en raison de sa récurrence et de l'efficacité des procédures mises au point pour sa solution.

Ce document est organisé en quatre chapitres précédés d'une introductin générale.
Le premier chapitre est la base de la suite, on présente dedans quelques notions de base sur la théorie des graphes et les points essentiels de la programmation linéaire.

Ensuite dans le deuxième chapitre, on présente le problème de transport et sa modélisation mathématique en tant qu'un programme linéaire.

Dans le troisième chapitre on s'intéresse à la résolution du problème de transport avec la présentation des différentes méthodes de détermination d'un solution de base initiale puis les algorithmes d'optimisation de cette solution afin d'avoir une solution optimale.

Le quatrième chapitre est consacré à l'application des méthodes citées dans le troisième chapitre et la programmation en langage C++ sur un exemple illustratif.

On termine le mémoire par une conclusion générale et les perspectives de recherche.

Chapitre 1

Notions de base sur la théorie des graphes et la programmation linéaire

Le but de ce chapitre est de récapituler quelques concepts fondamentaux de la théorie des graphes et des principes généraux de la programmation linéaire, nécessaire à une bonne compréhension dans la suite.

1.1 Eléments de la théorie des graphes

La théorie des graphes est une branche des mathématiques qui étudie les relations entre des objets distincts, appelés « nœuds » ou « sommets », à travers des connexions appelées « arêtes » ou « arcs ». Elle est largement utilisée dans la résolution de problèmes pratiques tels que la planification des réseaux de transport, l'ordonnancement des tâches, la modélisation des réseaux sociaux, la conception de circuits électroniques, et bien plus encore.

1.1.1 Qu'est-ce-que un graphe ?

Un graphe est une représentation géométrique qui consiste en un ensemble de points appelés sommets ou nœuds, qui sont reliés les uns aux autres par des lignes ou des flèches appelées arêtes ou arcs. Chaque arête relie deux points, qui peuvent éventuellement être les mêmes. Les graphes peuvent servir à représenter un grand nombre de situations courantes comme les liens routiers, les réseaux de communication, les circuits électriques, les liens entre diverses personnes ou entités administratives ..., etc. [11]

- ◇ **Une arête :** Une arête est une liaison non ordonnée entre deux sommets distincts u et v dans un graphe, établissant une connexion directe entre eux. On note e=uv ; on a uv=vu, la notation uv et vu désigne la même arête e. [11]

- ◇ **Un arc :** Un arc dans un graphe orienté est une connexion unidirectionnelle entre deux sommets distincts, représentée par une flèche ou une courbe indiquant la direction de la relation entre ces sommets. [11]

- ◇ **Une boucle :** On appelle une boucle dans un graphe une arête ou un arc qui relie un sommet à lui-même ; autrement dit l'extrémité initiale est confondue avec l'ex-

trémité terminale. [11]

◇ **Extrémité initiale et terminale (successeur et prédécesseur) :**
Dans un graphe, soit un arc (i, j), on dit que i est une extrémité initaiale de l'arc (i, j) et j est l'extrémité terminale. Aussi j est dit successeur de i, et i le prédecesseur de j. [8]

Définition mathématique d'un graphe

Un graphe est un couple d'ensemble G = (V, E) ou G = (V, U) ; tels que V = $\{v_1, v_2, \ldots, v_n\}$ est l'ensemble des sommets du graphe G et E = $\{e_1, e_2, \ldots, e_n\}$ est l'ensemble de ses arêtes où U = $\{u_1, u_2, \ldots, u_n\}$ est l'ensemble de ses arcs si c'est un graphe orienté. [1]

Exemple :

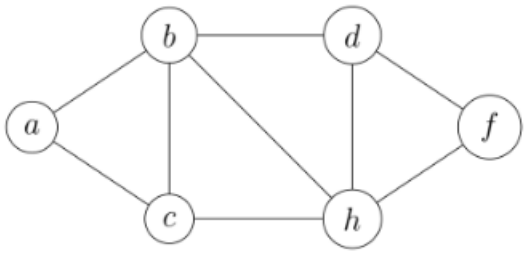


FIGURE 1.1 – Graphe de la relation d'amitié

Dans la figure 1.1 ; on a une représentation graphique d'une relation d'amitié entre six personnes. Chaque personne est représentée par un sommet désigné, par exemple, par la première lettre de son prénom ; si deux personnes sont amies leur relation d'amitié est représentée par une arête reliant les sommets qui leur correspondent.

1.1.2 Types de graphes

a) **Un graphe orienté :**
Un graphe orienté est un système formé d'un ensemble fini de sommets V = $\{v_1, v_2, \ldots, v_n\}$ et d'un ensemble fini d'arcs U = $\{u_1, u_2, \ldots, u_n\}$ relient dans un ordre bien définit ces sommets.

- Autrement dit :
  Un graphe orienté est définit par le quadruplet : G = (V, U, I, T) où
  o I est l'application extrémité initiale d'un arc définie par :

$$I : U \to V$$
$$(x, y) \to I(x, y) = x$$

  o T est l'application extrémité terminale d'un arc défini par :

$$T : U \to V$$
$$(x, y) \to T(x, y) = y$$

[11]
Exemple : la figure ci-dessous représente un graphe orienté.
Dans ce graphe on a l'ensemble des sommets V = $\{x_1, x_2, x_3, x_4\}$
L'ensemble des arcs U = $\{e_1, e_2, e_3, e_4, e_5\}$
On prend un arc du graphe G par exemple l'arc e_2 ; le sommet $x_(1)$est l'extrémité initiale de l'arc e_2 notée : I $(e_2) = x_1$; et le sommet x_4 est l'extrémité terminale de l'arc e_2 notée : T $(e_2) = x_4$.

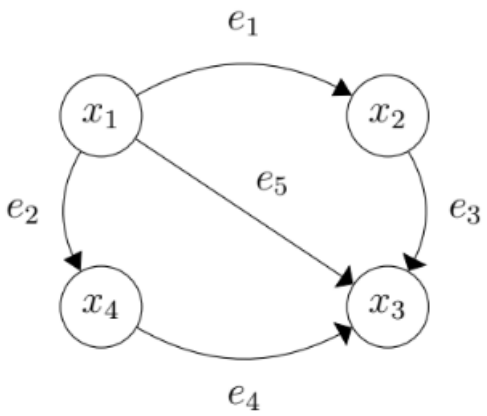


FIGURE 1.2 – Un graphe orienté

b) **Un graphe non orienté :**
Si nous définissons une relation sur un ensemble où l'ordre n'a pas d'importance, nous symbolisons la connexion entre deux sommets par une ligne sans direction, appelée arête. Ainsi, nous obtenons un graphe non orienté G = (V, E), où V représente les sommets et E représente les arêtes. [11]
Exemple :
La figure suivante est une illustration d'un graphe non orienté.
Dans le graphe ci-après on a : L'ensemble des sommets V = $\{v_1, v_2, v_3, v_4, v_5, v_6\}$
L'ensemble des arêtes E = $\{v_1v_2, v_1v_3, v_2v_3, v_2v_5, v_2v_6, v_5v_4, v_5v_6\}$

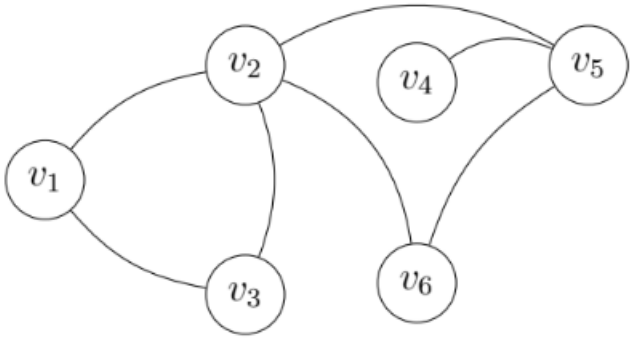


FIGURE 1.3 – Un graphe non orienté

c) **Un graphe pondéré :**
Un graphe pondéré aussi appelé graphe valué est un type de graphe où chaque arête est associée à une valeur numérique appelée poids. Ce poids peut représenter différentes mesures selon le contexte, comme la distance, le coût, le temps, la capacité, etc. Ainsi, les arêtes d'un graphe pondéré ont des valeurs qui reflètent une certaine caractéristique ou contrainte de la relation entre les sommets qu'elles relient. [11]
Exemple :

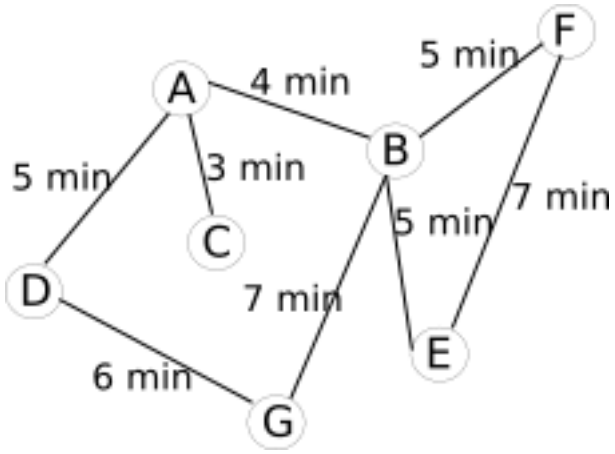


FIGURE 1.4 – un graphe pondéré (valué)

la figure 1.4 représente un graphe valué tel que sur chaque arête il y a un poids qui désigne ici le temps pour passer d'un sommet à un autre.

1.1.3 Ordre et multiplicité d'un graphe

a) **Ordre d'un graphe :**
Un graphe G est dit d'ordre n, s'il contient n sommets. C'est-à-dire l'ordre de G est le cardinale de V, on note |V|. Le graphe G est dit fini s'il est d'ordre fini. [11]
Exemple :
Dans le graphe représenté dans la figure 1.1, on dit que le graphe est d'ordre 6 car il a 6 sommets.

b) **Multiplicité d'un graphe :**

◇ **Un graphe simple et graphe multiple :**
Un graphe est dit graphe simple si c'est un graphe sans boucles ni arc (arête) multiples. Dans le cas contraire, c'est-à-dire, si des boucles ou des arcs (arêtes) multiples sont autorisés, on dira alors que le graphe est multiple. [11]
Exemple :
Dans la figure au-dessous on a deux graphes multiples ; un avec des arcs multiples et l'autre avec une boucle et des arêtes multiples.
Tandis que dans la figure 1.3 on a un graphe simple car il n'a ni boucle ni arêtes multiples.

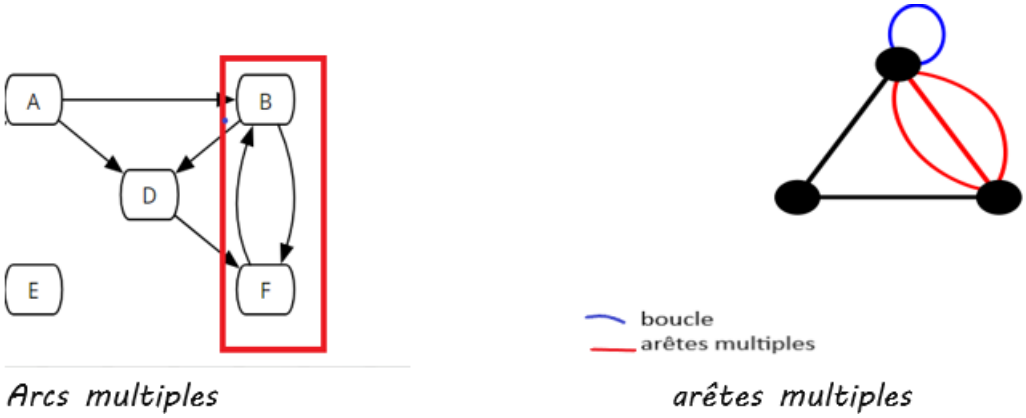


FIGURE 1.5 – Graphes multiples

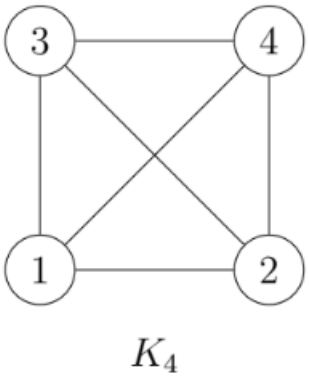
◇ **La multiplicité d'un graphe :**

On définit ainsi la multiplicité d'un graphe orienté multiple par le nombre maximum d'arcs ayant la même extrémité initiale et la même extrémité terminale. on appele ce nombre P, on dit alors que G est un *P-graphe*.

$$P = \max\{u \in U \mid I(u) = x \text{ et } T(u) = y\}$$

1.1.4 Classifications des graphes

a) **Un graphe complet (Clique) :**
On dit qu'un graphe est complet ou une clique si tous les sommets de ce graphe sont deux à deux adjacents. Autrement dit, il y a une arête ou un arc entre chaque deux sommets. On note K_n une clique d'ordre n. [11]
Exemple :
La figure suivante illustre une clique d'ordre 4 (K_4).



K_4

FIGURE 1.6 – Un graphe complet

b) **Un stable :**
Un stable dans un graphe G est un ensemble de sommets deux à deux non adjacents, c'est-à-dire un sous-graphe sans arête. [11]
Exemple :
Dans la figure 1.3 on a l'ensemble des sommets V $=\{v_1, v_4, v_6\}$ forme un stable.

c) **Un graphe biparti :**
Un graphe G = (V, E) est dit biparti si V peut être partitionné en deux ensemble X et Y, tels que deux sommets du même ensemble ne peuvent pas être adjacents ; autrement dit une arête e de E a une extrémité dans X et l'autre extrémité dans Y. On écrit alors G = (X, Y, E).
De plus si pour tout $x \in X$ et pour tout $y \in Y$, $xy \in E$, le graphe G est un biparti complet ou biclique.
La notation $K_{p,q}$ désigne une biclique avec $|X| = p$ et $|Y| = q$. [11]
Exemple :

La figure 1.7 montre un graphe biparti tel que l'ensemble des sommets est partitionné en deux X = $\{1, 2, 5\}$ et Y = $\{3, 4\}$, et il est complet puisque chaque sommet de X est adjacent à tous les sommets de Y.

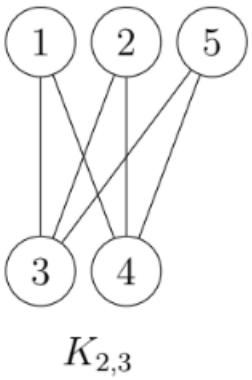


FIGURE 1.7 – Un graphe biparti complet (biclique)

d) **Sous-graphe (partiel, induit) :**

- ⋆ Un graphe partiel (couvrant) d'un graphe G = (V, E) est un sous-graphe H = (V, E') tel que $E' \subset E$, c'est-à-dire h est obtenu à partir de G par suppressions des arêtes sans toucher aux sommets.
  Si A est l'ensemble des arêtes supprimées, on écrit $H = G \setminus A = (V, E - A)$ pour désigner le graphe partiel qui en résulte. [8]
  Exemple :
  Dans la figure qui suit on a un graphe G = (X, E) à gauche et puis à droite on voit le graphe partiel qui en résulte par la suppression de quelques arêtes de G.
- ⋆ Un sous-graphe induit par un sous-ensemble de sommets S de V_G est le sous-graphe noté G_S dont l'ensemble des sommets est S et l'ensemble d'arêtes est constitué des toutes les arêtes de G. le sous-graphe induit G_S est le sous-graphe obtenu par la suppression consécutive des sommets de $V_G \setminus S$. [8]

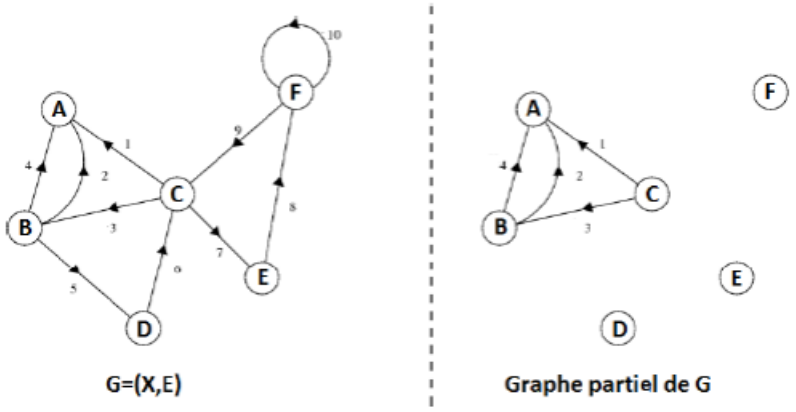


FIGURE 1.8 – Illustration d'un graphe partiel

Exemple :
La figure illustre un graphe G et le sous-graphe obtenu par la suppression d'un sommet de G.

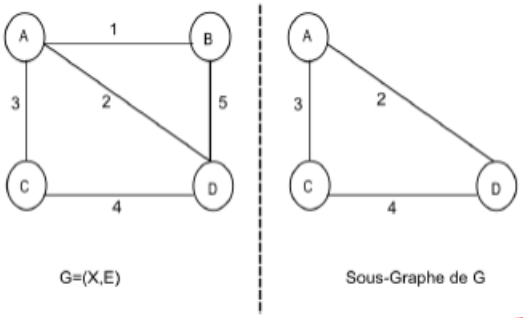


FIGURE 1.9 – Illustration d'un sous-graphe induit

1.1.5 La représentation matricielle

Soit G un graphe orienté (ou non) contenant n sommets et m arcs (arêtes) ; la représentation des graphes en ordinateur nécessite d'associer à G différents types de matrices, matrice d'adjacence, matrice d'incidence.

a) **Matrice d'adjacence :**
La matrice d'adjacence d'un graphe G d'ordre n est une matrice n×n ; ses éléments prennent deux valeurs 1 ou 0. Chaque ligne et chaque colonne correspond à un sommet du graphe. Ainsi chaque élément de la matrice indique la relation qui existe entre deux sommets.
a_{ij}=1 S'il existe une arête (arc) entre les sommets i et j, et a_{ij}=0 Sinon.[11]

$$a_{ij} = \begin{cases} 1 & \text{si } (i,j) \in E \text{ (resp. } (i,j)) \in E') \\ 0 & \text{sinon} \end{cases}$$

▷ **Remarque 1 :**
Une matrice d'adjacence dépend de la numérotation des sommets du graphe

quelle représente, donc si nous changeons la numérotation des sommets, nous trouverons une autre matrice d'adjacence.

▷ **Remarque 2 :**
La matrice d'adjacence d'un graphe orienté G est la même matrice du graphe non orienté G. Notons aussi que la matrice d'adjacence d'un graphe simple est symétrique et sa diagonale est nulle.

Exemple :
On prend le graphe déjà illustré dans la figure1.2. On trouve la matrice d'adjacence suivante :

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

b) **Matrice d'incidence :**
La matrice d'incidence d'un graphe simple est la matrice B = (n×m), une matrice ayant n lignes et m colonnes ; chaque ligne de la matrice est associée à un sommet et chaque colonne est associée à une arête (arc), ses éléments prennent les valeurs 1, 0, -1 tel que :

- Pour un graphe orienté $b_{ij} = 1$ si le sommet i est l'extrémité initiale de l'arc j, b_{ij} = -1 si i est l'extrémité terminale de j, $b_{ij} = 0$ sinon.
- Pour un graphe non orienté $b_{ij} = 1$ si le sommet i est une extrémité de l'arête j,
  $b_{ij} = 0$ sinon.
  Pour un graphe orienté, la matrice d'incidence est définie :

$$m_{ij} = \begin{cases} 1 & \text{si } x_i = I(u_i) \\ -1 & \text{si } x_i = T(u_i) \\ 0 & \text{sinon} \end{cases}$$

Pour un graphe non orienté, la matrice d'incidence est définie :

$$m_{ij} = \begin{cases} b_{ij} = 1 & \text{si } x_i \text{ est une extrémité de } e_i \\ b_{ij} = 0 & \text{sinon} \end{cases}$$

[11]
Exemple :
Soit la figure suivante qui montre un graphe non orienté G, et le graphe H obtenu par l'orientation de G.

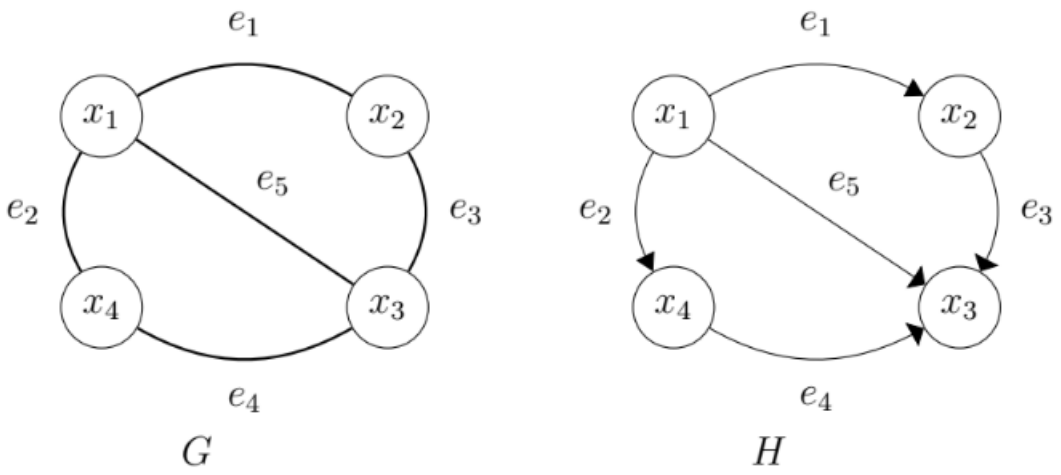


FIGURE 1.10 – Un graphe non orienté G et un graphe orienté H

La matrice d'incidence du graphe orienté H dans la figure 1.10 est la suivante :

$$\begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \begin{pmatrix} e_1 & e_2 & e_3 & e_4 & e_5 \\ 1 & 1 & 0 & 0 & 1 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 \\ 0 & -1 & 0 & 1 & 1 \end{pmatrix}$$

La matrice d'incidence du graphe G non orienté dans la figure 1.10 est la suivante :

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

1.1.6 La connexité dans un graphe

a) **Cheminement dans un graphe :**

◇ **La chaine :** Une chaine dans un graphe est une séquence de sommets reliés les uns aux autres par des arêtes. Formellement, une chaine dans un graphe est une suite de sommets $(v_1, v_2, \ldots, v_k)$ où chaque paire de sommets consécutifs $(v_i, v_{(i+1)})$ est relié par une arête.
On dit que v_1 et v_k sont les extrémités de la chaine.
La longueur d'une chaine est le nombre des arêtes qu'elle contient. Une chaine peut être orienté ou non orienté en fonction du type de graphe.
Une chaine est dite simple si on passe une seule fois par ses arcs (arêtes).[11]

◇ **Le chemin :** Un chemin dans un graphe G, est une suite de sommets reliés successivement par des arcs orientés dans le même sens. On le note : $(v_1, v_2, v_3, \ldots, v_k)$.
Deux sommets successifs d'un chemin sont respectivement extrémité initiale et terminale du même arc. [11]

Exemple :
Dans le graphe qui suit on une représentation d'une chaine en rouge et un chemin en vert.

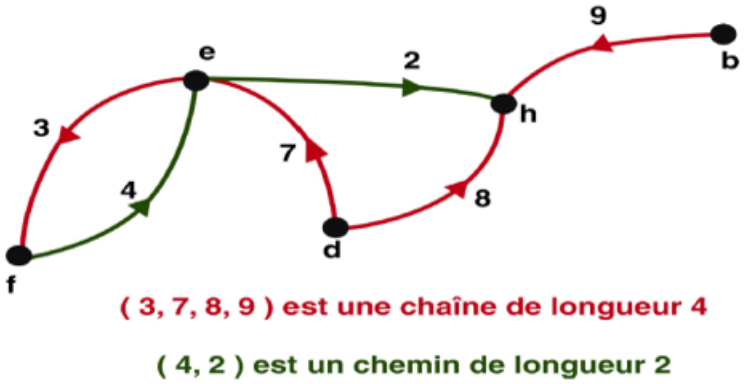


FIGURE 1.11 – Illustration d'une chaine et un chemin dans un graphe

◇ **Le cycle :** Un cycle est une chaine simple dont les deux extrémités coïncident (x_0 coïncide avec x_k). On le note : $(x_0, x_1, x_2, x_3, \ldots, x_k = x_0)$. [11]

◇ **Le circuit :** Un circuit est une série de sommets et d'arêtes dans un graphe qui forme une boucle, où le point de départ et le point d'arrivée sont les mêmes. C'est un type spécifique de cycle où tous les sommets visités sont uniques, sauf pour le point de départ et le point d'arrivée, qui se chevauchent. [11]

b) **La connexité :**
Un graphe G est dit connexe si entre deux sommets quelconques, il existe une chaine allant de l'un vers l'autre. Un graphe non connexe est constitué de plusieurs composantes connexes.
On appelle composante connexe un ensemble de sommets, qui ont 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.
Un graphe est connexe s'il contient une seule composante connexe. [11]

Exemple :
Dans la figure suivante on trouve un graphe G non connexe composé de trois composantes connexes comme suit : $C_1 = \{a, b, c, d\}$, $C_2 = \{g, e, f\}$ et $C_3 = \{h, i\}$.

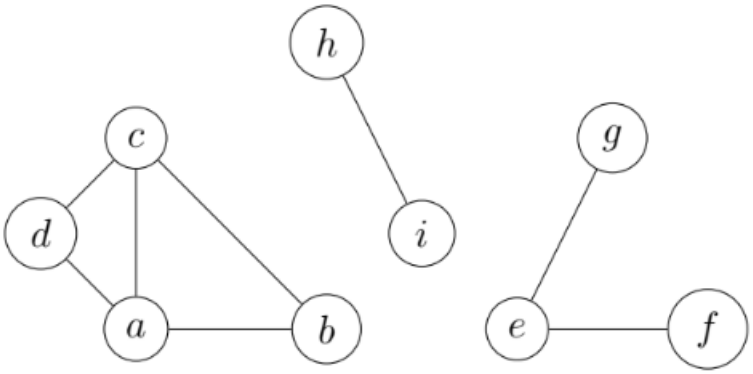


FIGURE 1.12 – Un graphe non connexe

◇ **Forte connexité :**
Un graphe orienté G = (V, U) est fortement connexe si pour tout couple de deux sommets x et y, il existe un chemin allant de x à y et un autre chemin allant de y à x, c'est-à-dire il existe un circuit passant par les deux sommets.[11]

1.1.7 Un arbre

Les arbres (les arborescences dans le cas d'un graphe orienté) représentent une classe de graphes largement utilisée en analyse de données et la résolution des problèmes. Ils permettent de représenter des relations de hiérarchie et s'apprêtent bien aux traitements récursifs et les traitements efficaces.[5]

Définition :

Un arbre est un graphe connexe, acyclique (sans cycle).

Un arbre a les propriétés suivantes :
– G est connexe et sans cycle.
– G est sans cycle et possède n-1 arêtes.
– G est connexe et admet n-1 arêtes.
– G est sans cycle, et en ajoutant une arête, on crée un et un seul cycle élémentaire.
– G est connexe, et en supprimant une arête quelconque, il n'est plus connexe.
– il existe une chaîne et une seule entre 2 sommets quelconque de G.

Exemple :

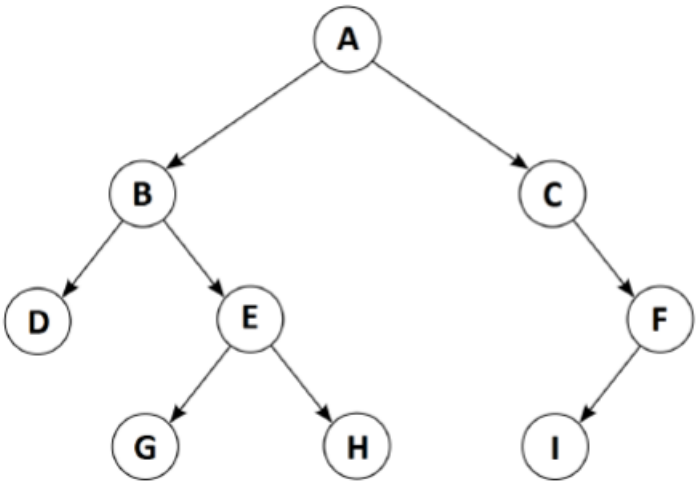


FIGURE 1.13 – Illustration d'un arbre

1.2 Programmation linéaire

La programmation linéaire est l'une des parties essentielle de la recherche opérationnelle. C'est une méthode mathématique utilisée pour trouver la meilleure solution à un problème d'optimisation (maximisation ou minimisation) d'une fonction linéaire soumise à des contraintes linéaires. Elle est largement utilisée dans divers domaines tels que l'économie, la gestion, l'ingénierie, la logistique et bien d'autres.
Le terme programmation linéaire a été introduit, en même temps que la méthode du simplexe (l'une des méthodes les plus connues pour résoudre des programmes linéaires en nombre réel).
La programmation linéaire a été introduite par le russe Kantorovitch, et la première résolution a été faite par l'américain G.B.Dantzig en 1951.

1.2.1 Modélisation

La modélisation d'un programme linéaire (PL) implique de définir une fonction objectif linéaire à maximiser ou à minimiser, ainsi que des contraintes linéaires qui décrivent les limitations du problème.

- Les variables de décision : les variables de décision dans un programme linéaire sont les inconnus qu'on essaye de résoudre afin d'optimiser la fonction objectif tout en tenant compte des contraintes du problème.
  On note x_j Les variables de décision avec j=1, ..., n.
- La fonction objectif : La fonction objectif est l'expression mathématique qui représente la quantité à maximiser ou minimiser dans un problème d'optimisation, en fonction des variables du problème.

$$Z = \max \text{ ou } \min \sum_{j=1}^{n} c_j x_j = c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$$

  Où x_j sont les variables de décision ; c_j c'est le coefficient de contribution de la variable x_j dans la fonction objectif.

- Les contraintes : Les contraintes sont des équations ou des inégalités qui décrivent les limitations du problème. Elles sont généralement exprimées sous forme d'équations linéaires ou d'inégalités linéaires.

$$\forall i = 1, \ldots, m : \sum_{j=1}^{n} a_{ij} x_j \leq, =, \text{ ou } \geq b_i$$

Les nombres a_{ij} et b_i des constantes réelles et m le nombre de contrainte.[10]

1.2.2 Forme générale d'un programme linéaire

$$\begin{cases} (1) \quad Z = \max \text{ ou } \min \sum_{j=1}^{n} c_j x_j \\ (2) \quad \forall i = 1, \ldots, m : \sum_{j=1}^{n} a_{ij} x_j \leq, =, \text{ ou } \geq b_i \\ (3) \quad \forall j = 1, \ldots, n : x_j \geq 0 \end{cases}$$

- La fonction Z (1) est appelée fonction objectif, fonction de but ou critère de qualité.
- Les contraintes (2) sont appelées contraintes linéaires, principales, ou essentielles.
- Les contraintes (3) sont dites directes ou de positivité.

1.2.3 Forme standard et forme canonique d'un programme linéaire

Les problèmes de programmation linéaire peuvent naturellement se présenter sous une forme différente. 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.

Définition 1.2.1 *(forme canonique)*
Un programme linéaire est sous forme canonique lorsque toutes ses contraintes sont des inégalités et toutes ses variables sont positives. [10]

$$\begin{cases} max \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \leq b_i \\ x_j \geq 0 \end{cases}$$

Définition 1.2.2 *(forme standard)*
Un programme linéaire est sous forme standard lorsque toutes ses contraintes sont des

égalités et toutes ses variables non-négatives. [10]

$$\begin{cases} max \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j = b_i \\ x_j \geq 0 \end{cases}$$

1.2.4 Formes matricielles classiques et conventions

Soit x = $(x_1, x_2, x_3, \ldots, x_n)^T$ le vecteur des variables, b = $(b_1, b_2, \ldots, b_m)^T$ le second membre des contraintes, c = $(c_1, c_2, \ldots, c_n)^T$le vecteur coût ou profit associé aux variables et A la matrice n×m des a_{ij}.

$$\begin{cases} \text{forme canonique} \\ \max z = cx \\ Ax \leq b \\ x \geq 0 \end{cases} \quad \text{ou} \quad \begin{cases} \max z = cx \\ Ax \geq b \\ x \geq 0 \end{cases} \qquad \begin{cases} \text{forme standard} \\ \max z = cx \\ Ax = b \\ x \geq 0 \end{cases}$$

◇ **Exemple d'un programme linéaire :**
Une unité de production de parpaings fabrique quatre types de produit : Les parpaings de dimensions respectivement 10 cm (noté P_1), 15 cm (noté P_2), 20 cm (noté P_3) et l'ourdi (noté P_4).
Pour la fabrication de ces produits, on utilise quatre matières premières, la sable (M_1), le gravier (M_2), le ciment (M_3) et l'eau (M_4), disponibles en quantité respectivement illimitée.
Le plan de production de l'unité est donné dans le tableau suivant :

Matières premières	P_1	P_2	P_3	P_4	Quantités matières premières disponibles
M_1	2	3	5	6	5000
M_2	1	2	3	3	3000
M_3	0.8	1	2	3	2000
M_4	1	1	2	2	/

TABLE 1.1 – Tableau des matières premières et produits

Le tableau signifie que pour fabriquer un parpaing de 10 cm, il faut 2 unités de M_1, 1 unité de M_2, 0.8 unité de M_3 et 1 unité de M_4, et de la même manière pour P_2,P_3 et P_4. Les parpaings sont vendus respectivement à raison de 6, 7, 9 et 10 DA l'unité. Le problème pour la direction de l'unité est de trouver le nombre maximal de produit P_1, P_2, P_3 et P_4 à fabriquer pour avoir un bénéfice maximal, tout en respectant les contraintes de l'unité.[10]
Résolution :
Soit x_1, x_2, x_3 et x_4 les quantités de produits P_1, P_2, P_3 et P_4.
Ces quantités doivent vérifier les conditions suivantes :

- Les quantités utilisées en matières premières ne doivent pas dépasser les quantités disponibles :

$$\begin{cases} 2x_1 + 3x_2 + 5x_3 + 6x_4 \leq 5000 \\ x_1 + 2x_2 + 3x_3 + 3x_4 \leq 3000 \\ 0.8x_1 + x_2 + 2x_3 + 3x_4 \leq 2000 \end{cases}$$

- Les quantités à produire sont toutes positives ou nulles :

$$x_1 \geq 0, \quad x_2 \geq 0, \quad x_3 \geq 0, \quad x_4 \geq 0$$

Comme l'eau est disponible en quantité illimitée, donc on n'a aucune contrainte sur la matière première M_4.
Le chef de production de l'unité choisira le programme réalisable qui donnera le maximum de la fonction bénéfice Z :

$$Z(x_1, x_2, x_3, x_4) = 6x_1 + 7x_2 + 9x_3 + 10x_4 \rightarrow \max$$

Z représente le bénéfice que va réaliser l'unité.
Le chef de production aura pour objectif de trouver la solution optimale du problème de programmation linéaire suivant :

$$\begin{cases} Z(x_1, x_2, x_3, x_4) = 6x_1 + 7x_2 + 9x_3 + 10x_4 \rightarrow \max \\ 2x_1 + 3x_2 + 5x_3 + 6x_4 \leq 5000 \\ x_1 + 2x_2 + 3x_3 + 3x_4 \leq 3000 \\ 0.8x_1 + x_2 + 2x_3 + 3x_4 \leq 2000 \\ x_j \geq 0, \quad j = 1, 2, 3, 4 \end{cases}$$

1.2.5 Interprétation écnomique

Un programme linéaire a une interprétation économique très large :
- Un acteur économique qui exerce n activités avec des niveaux x_j à déterminer.
- Ces activités utilisent m ressources.
- La quantité a_{ij} de ressources i nécessaires pour exercer l'activité j avec un niveau de 1.
- On connait le profit (danx le cas de maximisation) et le coût (en cas de minimisation).
- c_j Correspond à un niveau de 1 pour l'activité j. [1]

1.2.6 Solution d'un problème

Définition 1.2.3 *(région réalisable)*
Ensemble de points qui satisfont aux contraintes du problème.[6]

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

Définition 1.2.4 *(solution admissible ou réalisable)*
Une solution est réalisable si les valeurs numériques $x_1, \ldots, x_n$ satisfont l'ensemble de contraintes du problème. $x \in X$ [6]

Définition 1.2.5 *(solution optimale)*
Une solution réalisable x^* est optimale si la valeur qu'elle donne à la fonction objectif est $\leq$ aux valeurs données par les autres solutions réalisables. On note $cx^* = maxcx$. [6]

Définition 1.2.6 *(variable de base)*
Les variables dans une solution de base dont les valeurs sont obtenues en tant que solution simultanée du système d'équations qui constituent les contraintes fonctionnelles. [6]

1.2.7 Les méthodes de résolution d'un programme linéaire

La méthode graphique

La méthode graphique en programmation linéaire est une technique utilisée pour résoudre des problèmes d'optimisation linéaire avec deux variables de décision. Elle est particulièrement utile pour visualiser les contraintes et la fonction objectif d'un problème, ce qui peut aider à comprendre et résoudre le problème de manière intuitive.
La résolution par la méthode graphique en programmation linéaire se fait en plusieurs étapes.

- Formulation du problème :
  Premièrement on commence par exprimer le problème posé sous forme d'un programme linéaire c'est-à-dire une fonction linéaire à maximiser ou à minimiser avec des contraintes linéaires.
- Representation graphique des contraintes :
  On dessine sur un graphique cartésien les droites correspondantes à chaque contrainte écrite sous forme d'inégalité on remplace $\leq$ par $=$. Chaque contrainte est symbolisée par une équation linéaire, séparant ainsi l'espace en deux régions distinctes une région admissible et une région inadmissible.
- Détermination de la région réalisable :
  On identifie la région commune à toutes les contraintes autrement dit l'intersection des demi-plans où toutes les contraintes sont toutes satisfaites simultanément. Cette région doit être convexe, et forme un polygone.
  o Un polygone est une forme géométrique plate formée par une série de segments de droite qui se connectent pour créer une figure fermée.
- Représentation de la fonction objectif :
  Tracer la droite correspondante à la fonction objectif sur le même graphique. La direction que prenne cette droite dépend de la nature du problème (à maximiser ou à minimiser) et est guidée par le vecteur gradient de la fonction objectif.
- Le point optimal :
  Le point optimal est est toujours un sommet du polygone trouvé, on le touve en deplaçant la droite de la fonction objectif jusqu'à ce qu'elle atteinge la valeur maximale ou minimale désirée.
- Une fois la solution optimale est trouvée on utilise ses coordonées pour déterminer la valeur optimale de la fonction objectif ainsi que les valeurs optimales des variables de décision.

La méthode du simplexe

La méthode du simplexe est un algorithme permettant de calculer une chaine de solutions de base admissible qui améliorent graduellement la valeur de la fonction objectif c'est-à-dire passer d'un sommet à un sommet voisin meilleur jusqu'à arriver à un sommet qui est la solution optimale.
La méthode du simplexe a été développée par George Dantzig dans les années 1940 et reste une méthode efficace et l'une des techniques les plus couramment utilisée pour résoudre des problèmes de programmation linéaire (avec un grand nombre de variables et

de contraintes).

◇ **Résolution d'un problème PL par la méthode du simplexe :**

Pour appliquer la méthode du simplexe sur un problème linéaire sous la forme générale, il faut d'abord le transformer sous la forme standard de la manière suivante : Un PL sous forme générale c'est-à-dire la fonction objectif est à minimiser et les contraintes sont ($\geq$).
Pour passer à la forme standard, il faut d'abord transformer le PL en forme canonique en multipliant la fonction objectif Z $\times$(-1) et les contraintes du problème des deux côtés $\times$(-1) ; autrement dit min Z = max (-Z) et les contraintes ($\geq$) deviennent ($\leq$).
Après la forme canonique on peut passer à la forme standard en ajoutant d'autres variables positives appelées variables d'écarts à la fonction objectif et aux contraintes et les contraintes d'inégalité deviennent des contraintes d'égalité (=).
Après avoir la forme standard on applique l'algorithme du simplexe.

◇ **Algorithme du simplexe :**
- Après écriture du PL sous la forme standard ; on construit le premier tableau correspondant à la forme standard.
- On choisit une variable hors base ayant le coût réduit (négatif) le plus petit alors cette variable va rentrer en base on la note x_r.
- Pour trouver la variable qui va sortir de la base on fait le rapport composante par composante du second membre par la colonne de x_r dans la matrice et on ne considère que les rapports positifs, et on choisit le plus petit rapport.

$$\min_{i=1,\ldots,m} \left\{ \frac{b_i}{a_{ir}}, a_{ir} > 0 \right\} = \frac{b_i}{a_{sr}}$$

La variable qui va sortir de la base est x_s.
- a_{sr} est appelé pivot si a_{sr}= 1 on le garde tel qu'il est, si $a_{sr} \neq 1$ on divise la ligne du pivot par a_{sr} pour avoir un pivot égal à 1.
- On calcule les valeurs des autres lignes de la façon suivante :

$$a_{ij} = a_{ij} - \left(\frac{a_{ir}}{a_{sr}} \right) \times a_{sj}$$

- On vérifie si les coûts réduits (les coefficients de la fonction objectif) sont-ils tous nuls ou négatifs, si oui on est à l'optimum, sinon on effectue un nouveau passage.
Toutes ces étapes on peut les résumer dans la figure suivante :

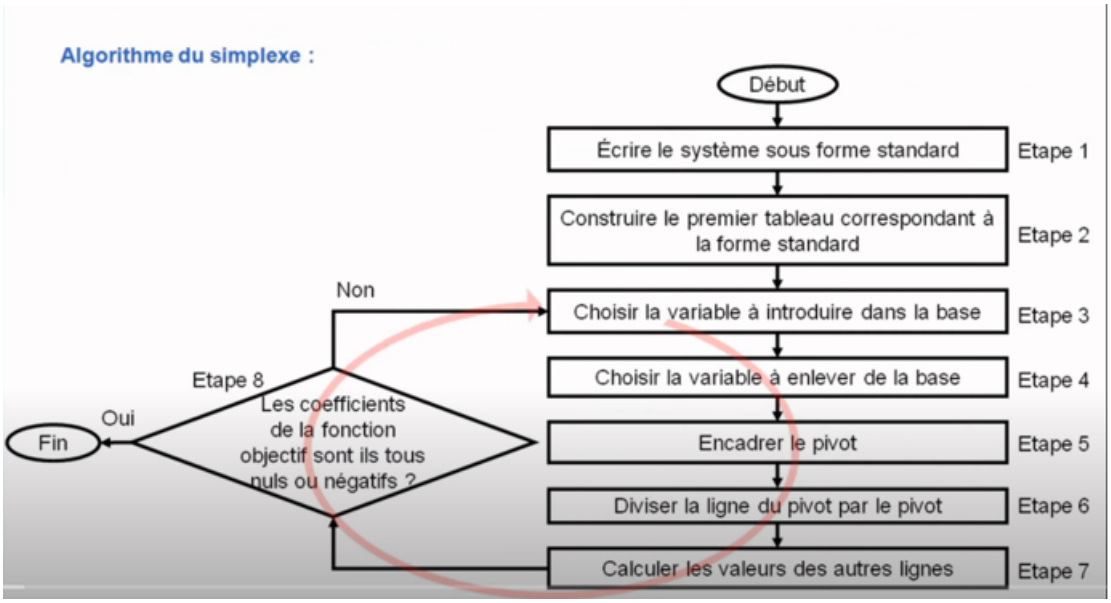


FIGURE 1.14 – Etape de l'algorithme du simplexe

Chapitre 2

La complexité

Dans ce chapitre on vise à introduire le concept de la complexité, qui évalue les ressources nécessaires pour résoudre un problème, ainsi que la classification des problèmes et la complexité algorithmique. On va aborder aussi la complexité temporelle et spatiale, qui mesurent respectivement le temps et la mémoire requis en fonction de la taille de l'entrée.

2.1 Théorie de la complexité

La théorie de la complexité s'intéresse à l'étude formelle de la difficulté des problèmes en informatique, distinctement de la théorie de la calculabilité, qui se concentre sur la possibilité de résoudre un problème par un ordinateur. En se focalisant sur les problèmes solubles, la théorie de la complexité évalue si ces problèmes peuvent être résolus de manière efficace, en estimant théoriquement les temps de calcul et les besoins en mémoire nécessaires.[13]

- 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.[8]

2.2 La complexité d'un problème

La complexité d'un problème en informatique repose sur celle des algorithmes qui le résolvent. Elle est définie comme la complexité minimale des algorithmes capables de répondre à ce problème. En général, la complexité d'un problème évalue les ressources nécessaires pour le résoudre, notamment en termes de temps (complexité temporelle) et de mémoire (complexité spatiale).[14]

2.2.1 Classe de complexité

La classe de complexité d'un problème est définie par la difficulté inhérente à le résoudre. Les classes de complexité couramment étudiées en théorie de la complexité incluent notamment :

- P (Polynomial Time) : Les problèmes qui peuvent être résolus par un algorithme en temps polynomial, c'est-à-dire en temps qui augmente au plus comme une puissance d'un entier naturel.
- NP (Nondeterministic Polynomial Time) : Les problèmes dont les solutions peuvent être vérifiées par un algorithme en temps polynomial, mais qui peuvent ne pas être résolus par un algorithme déterministe en temps polynomial.
- NP-complets : Les problèmes les plus difficiles de la classe NP, qui sont au moins aussi difficiles que tous les autres problèmes NP. Si un algorithme peut résoudre un problème NP-complet, il peut résoudre n'importe quel autre problème NP.

Ces classes de complexité sont utilisées pour évaluer la difficulté de résoudre un problème et pour examiner les relations entre différents problèmes algorithmiques. [2]

2.3 Complexité algorithmique

La complexité algorithmique mesure la quantité de ressources nécessaires pour décrire ou générer une suite de données ou résoudre un problème. Elle est souvent définie comme la taille du plus court programme, généralement en bits, capable de produire une suite donnée lorsqu'il est exécuté sur une machine de Turing universelle. Développée par Andreï Kolmogorov et Gregory Chaitin, cette théorie formalise l'idée que la complexité réside dans ce qui ne peut pas être simplement exprimé, soulignant ainsi la relation entre complexité et aléatoire.[7]

La complexité d'un algorithme peut être évaluée en termes de temps et d'espace :
- Complexité temporelle : mesure du temps d'exécution de l'algorithme.
- Complexité spatiale : mesure de l'espace mémoire utilisé par l'algorithme lors de son exécution.

Remarque :
On se concentre principalement sur la complexité temporelle, car de nos jours, la mémoire est moins coûteuse et plus abondante qu'auparavant.

La complexité tomporelle :

La complexité temporelle d'un algorithme correspond au nombre d'opérations élémentaires (affectations, comparaisons, opérations arithmétiques) effectuées par cet algorithme. Ce nombre s'exprime en fonction de la taille n des données.[8]

On dit que la complexité de l'algorithme est $O(f(n))$ où f est d'habitude une combinaison de polynômes, de logarithmes ou d'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.[8]

La notation $O(\cdot)$:
Comme on l'a vu dans la section précédente, les calculs nécessaires pour évaluer le temps d'exécution d'un algorithme peuvent parfois être longs et fastidieux. De plus, le degré de précision requis est souvent superflu. Par conséquent, nous utilisons une approximation de ce temps de calcul, représentée par la notation $O(\cdot)$.[8]

Définition 2.3.1 *Une fonction $f(n)$ est en $O(g(n))$ si :* $\exists n_0 \in \mathbb{N}, \exists c \in \mathbb{R}, \forall n \geq n_0 : |f(n)| \leq c|g(n)|$.

En d'autres termes, $f(n)$ est en $O(g(n))$ s'il existe un seuil à partir duquel la fonction $f(n)$ est toujours inférieure à $g(n)$ avec une constante multiplicative fixée. Bien que la définition implique l'utilisation de valeurs absolues, en pratique, on ne se préoccupe jamais. Les fonctions que nous utilisons pour mesurer le temps d'exécution d'un algorithme seront toujours positives.[8]

2.3.1 Calcul de la complexité algorithmique

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 $O(\cdot)$. 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. [8]

Chapitre 3

Problème de transport

Ce chapitre vise à introduire et à représenter le problème de transport à travers ses diverses formulations.
Le Problème de Transport est un problème fondamental en recherche opérationnelle qui consiste à trouver le moyen le plus efficace de transporter des biens d'un ensemble de sources à un ensemble de destinations tout en minimisant les coûts de transport ou en maximisant le profit, sous certaines contraintes.

3.1 Le problème de transport

Le problème de transport est un problème particulier de la programmation linéaire. Le plus connu de ces problèmes est le problème du transport de marchandises des usines (unités de production) aux dépôts (magasins).
Il peut être énoncé comme suit : « comment transporter des quantités de marchandises de m origines $X_i = \{x_1, x_2, ..., x_m\}$ vers n destinations $Y_j = \{y_1, ..., y_n\}$ de manière à minimiser les coûts, en respectant les disponibilités a_i aux origines et les demandes b_j aux destinations, tout en tenant compte des coûts de transport c_{ij} entre chaque origine et chaque destination».
Le problème de transport est identique au problème de l'offre et de la demande.

◇ **Caractéristique du problème de transport :**

- Le problème de transport est un problème d'optimisation combinatoire inscrit dans la lignée d'un programme linéaire.
- Le but du problème de transport est de transporter aux moindres coûts des marchandises de m sources à n destinations. [1]

3.2 Modèle mathématique du problème de transport

Considérons une entreprise équipée de m entrepôts et n points de vente. Son objectif est de transporter un seul produit des entrepôts vers les points de vente. Chaque entrepôt, étant une source, dispose d'un niveau d'approvisionnement déterminé (disponibilité), tandis que chaque point de vente, en tant que destination, exprime un niveau de demande fixe.
Les coûts associés au transport entre chaque paire d'entrepôt et de point de vente sont également fournis, et ces coûts sont supposés être linéaires.

- **Données :**

  Soit m points de productions (usines, sources) S_i, i=1, ..., m et n points de consommations (dépôts, destinations) D_j, $\forall j = 1, \ldots, n$.
  Une quantité a_i $\forall i = 1, \ldots, m$, est produite par chaque usine, cette quantité doit être acheminée vers le dépôt D_j, $\forall j = 1, \ldots, n$, dont la capacité est b_j, $\forall j = 1, \ldots, n$.
  Notre problème de transport consiste à transporter les quantités produites vers les dépôts, avec un cout de transport minimum.
  x_{ij} $\forall i = 1, \ldots, m$ et $\forall j = 1, \ldots, n$ la quantité de produits à transporter de S_i vers D_j. x_{ij} est une variable de décision dans le programme linéaire (PL) du problème de transport. c_{ij} Le coût de transport d'une unité de produit de S_i vers D_j.

◇ **Fonction objectif :**

  La fonction objectif contient des coûts associés à chacune des variables. C'est une minimisation du problème de transport.
  Si une quantité x_{ij} est transportée de S_i vers D_j, et que c_{ij} est le coût de son transport, alors le coût de transport de cette quantité est égal à $c_{ij}x_{ij}$, puisque la fonction de coût totale est supposée linéaire.
  De là, le coût global de transport est égal à

$$Z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

  Z est appelé la fonction objectif.

◇ **Les contraintes :**

  Les contraintes sont les conditions qui obligent à satisfaire la demande et épuiser la disponibilité.
  Les quantités à transporter doivent satisfaire les contraintes suivantes :
  i. Toute la marchandise produite doit être acheminée :

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

  ii. Toutes les demandes doivent être satisfaites :

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

  iii. Toutes les quantités transportées x_{ij} doivent être positives ou nulles :

$$x_{ij} \geq 0, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n$$

3.2.1 Formulation mathématique

Le modèle mathématique du problème de transport prend la forme du programme linéaire suivant :

$$\begin{cases} Z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \to \min \quad (1) \\ \sum_{j=1}^{n} x_{ij} = a_i, & i = 1,\ldots,m \quad (2) \\ \sum_{i=1}^{m} x_{ij} = b_j, & j = 1,\ldots,n \quad (3) \\ x_{ij} \geq 0, & i = 1,\ldots,m, \quad j = 1,\ldots,n \quad (4) \end{cases}$$

L'ensemble x = $\{x_{ij},\ \forall i = 1,\ldots,m$ et $\forall j = 1,\ldots,n\}$ est dit plan de transport du problème s'il satisfait les contraintes (2), (3) et (4). Il est dit plan optimal de transport si de plus il réalise le minimum de la fonction objectif (1). [10]

3.3 Condition d'existence d'un plan optimal de transport

Pour que le problème de transport admette une solution réalisable il faut que les quantités produites soient égales aux quantités demandées, c'est-à-dire :

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j \qquad (*)$$

La condition (*) est dite condition de balance. [10]

Théorème :

Le problème de transport possède un plan optimal de transport si et seulement si la condition de balance (*) est vérifiée. [10]

Preuve :

Soit $x^* = \{x^*_{ij},\ i = 1,\ldots,m, j = 1,\ldots,n\}$ un plan optimal du problème (1)-(4), vérifiant les contraintes (2) et (3).

En sommant la contrainte (2) par rapport à i et (3) par rapport à j, on obtient la condition de balance :

$$\sum_{i=1}^{m} a_i = \sum_{i=1}^{m}\left(\sum_{j=1}^{n} x^*_{ij}\right) = \sum_{j=1}^{n}\left(\sum_{i=1}^{m} x^*_{ij}\right) = \sum_{j=1}^{n} b_j$$

Inversement, en supposant que la condition (*) est vérifiée, on peut alors construire un plan de transport x du problème (1)-(4) de la manière suivante :

Posons $x_{ij} = \frac{a_i b_j}{K}, \quad i = 1,\ldots,m, \quad j = 1,\ldots,n.$

En effet, le vecteur $\mathbf{x} = \{x_{ij} = \frac{a_i b_j}{K}, i = 1,\ldots,m,\ j = 1,\ldots,n\}$ vérifie les contraintes (2),

(3) et (4).

$$\sum_{j=1}^{n} x_{ij} = \sum_{j=1}^{n} \frac{a_i b_j}{K} = \left(\frac{a_i}{K}\right) \sum_{j=1}^{n} b_j = \frac{a_i}{K} \times K = a_i, \quad i = 1, \ldots, m;$$
$$\sum_{i=1}^{m} x_{ij} = b_j, \quad j = 1, \ldots, m.$$

et

$$x_{ij} = \frac{a_i b_j}{K} \geq 0, \quad 1 \leq i \leq m, \quad 1 \leq j \leq n$$

Ainsi l'ensemble des solutions réalisables du problème (1)-(4) n'est pas vide.
En outre il est borné puisqu'on a pour tout plan de transport x les doubles inégalités suivantes :

$$0 \leq x_{ij} \leq \sum_{j=1}^{n} x_{ij} = a_i \leq \sum_{i=1}^{m} a_i = K$$

Il en résulte que la fonction continue (1) atteint son minimum sur cet ensemble. Par conséquent le problème (1)-(4) admet une solution optimale. [12]

3.4 Problème de transport non équilibré

Un problème de transport est dit non équilibré si on a :

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

Dans ce cas (problème de transport déséquilibré) on a deux cas :

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

ou :

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

On se ramène à un problème de transport équilibré de la manière suivante :
a. Si $\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j$, il suffit d'introduire une destination fictive $y_{(n+1)}$ de coût de transport égal à zéro entre x_i et $y_{(n+1)}$ $\forall i = 1, \ldots, m$ dont la demande est :

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

b. Si $\sum_{i=1}^{m} a_i < \sum_{j=1}^{n} b_j$, il suffit d'introduire une source fictive $x_{(m+1)}$ de coût de transport égale à zéro entre y_j et $x_{(m+1)}$ $\forall j = 1, \ldots, n$ dont la disponibilité :[1]

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

3.5 La matrice des coefficients des contraintes principales

Le problème de transport s'écrit de manière matricielle :

$$\begin{cases} \min \quad z = c^T x \\ \qquad\quad Ax = d \\ \qquad\quad x \geq 0 \end{cases}$$

Où x = $\{x_{11}, x_{1n}, x_{21}, x_{2n}, x_{mn}\}$. C'est-à-dire que l'on déroule la matrice x_{ij} suivant les lignes. On fait de même pour c = $(c_{11}, c_{1n}, c_{21}, c_{mn})$. Il y a m×n variables et m+n contraintes. Le vecteur d correspond à d = $(a_1, a_2, a_m, b_1, b_2, b_n)$.
Illustrons la matrice A pour m = 3 et n = 4.

$$\begin{bmatrix} 1111 & & & & & & & & & & & \\ & & & & 1111 & & & & & & & \\ & & & & & & & & & 1111 & & \\ 1 & & & 1 & & & & 1 & & & & \\ & 1 & & & & 1 & & & & 1 & & \\ & & 1 & & & & 1 & & & & 1 & \\ & & & 1 & & & & 1 & & & & 1 \end{bmatrix}$$

La somme des n premières lignes donne L_1+L_2+...+L_n = (1,1,...,1).
Aussi, la somme des lignes n+1 à n+m donne $L_{(n+1)}$+$L_{(n+2)}$+...+$L_{(n+m)}$=(1,1,...,1).
Si on combine les deux résultats, on obtient : L_1+L_2+...+L_n-$L_{(n+1)}$-$L_{(n+2)}$-...-$L_{(n+m)}$=0
Ceci implique que $\text{rg}(A) < m + n$.

◇ On a les propriétés suivantes pour la matrice A.
- Chaque colonne contient exactement deux entrées non nulles et qui sont égales à 1.
- Le rang de la matrice A est égal à $m + n - 1$: $\text{rg}\, A = m + n - 1$.
- Il y a toujours une ligne de trop que l'on peut éliminer.
- Il y a exactement $m + n - 1$ variables de base réalisables.[12]

3.6 Le problème dual du problème de transport

Le dual du problème de transport constitue un outil précieux pour analyser et optimiser les systèmes de transport. En saisissant les interprétations et les applications de ce dual, les décideurs sont en mesure de faire des choix éclairés qui optimisent l'utilisation des ressources et minimisent les coûts de transport. Un problème de transport est de la forme :

$$\min Z = \sum_{i,j} c_{ij} x_{ij} = c^T x$$

Sous les contraintes :

$$\forall i = 1, 2, \ldots, m \quad \sum_{j=1}^{n} x_{ij} = a_i \quad \Leftrightarrow \quad A_1 x = a$$

$$\forall j = 1, 2, \ldots, n \quad \sum_{i=1}^{m} x_{ij} = b_i \quad \Leftrightarrow \quad A_2 x = b$$

$$x_{ij} \geq 0 \quad \Leftrightarrow \quad x \geq 0$$

Sous forme compacte ceci s'écrit :

$$\min Z = c^T x$$

$$\begin{bmatrix} A_1 \\ -A_1 \\ A_2 \\ -A_2 \end{bmatrix} x \geq \begin{bmatrix} a \\ -a \\ b \\ -b \end{bmatrix}$$

Le dual est :

$$\max Z = a^t u_+ - a^t u_- + b^t v_+ - b^t v_-$$

$$\begin{bmatrix} A_1^t & -A_1^t & A_2^t & -A_2^t \end{bmatrix} \begin{bmatrix} u_+ \\ u_- \\ v_+ \\ v_- \end{bmatrix} \leq c$$

$$u_+, u_-, v_+, v_- \geq 0$$

C'est-à-dire avec $u = u_+ - u_-$ et $v = v_+ - v_-$, on obtient :

$$\max Z = a^t u + b^t v$$

$$A_1^t u + A_2^t v \leq c$$

u, v libres

Or $A_1^t u + A_2^t v \leq c \quad \Leftrightarrow \quad u_i + v_j \leq c_{ij}$.
Supposons que x soit la solution optimale du problème. Selon les conditions KKT, on a :

$$x^t (c - A_1^t u - A_2^t v) = 0$$

On a les relations :

$$\forall x_{ij} > 0 \quad u_i + v_j = c_{ij}$$

Si x est solution de base non dégénérée, on a bien la décomposition $u_i + v_j = c_{ij}$ pour les indices des variables de base.[12]
Remarque :
Le dual du dual est le primal. [12]

3.7 Le tableau de transport

Le problème de transport peut être formulé à l'aide d'un modèle de programmation linéaire et est souvent présenté sous forme de tableau de transport. Ce tableau compact représente le modèle avec tous les paramètres pertinents, permettant une visualisation claire et concise du problème.
Le tableau de transport à m lignes et à n colonnes, défini de la manière suivante :[10]

- Chaque ligne i ($i \in I$) correspond au point de production S_i et chaque colonne j ($j \in J$) au point de distribution D_j.
- La case (i, j) représente une voie, et contient les quantités associées à l'arc (i, j) :
  - Le coût unitaire c_{ij} de transport de S_i à D_j, indiqué en haut et à droite de la case.
  - La valeur de la variable x_{ij} écrite au centre de la case.
- En outre, le tableau est bordé à droite par une colonne indiquant les disponibilités a_i et par une ligne située en bas et indiquant les demandes b_j.

Le tableau de transport présente alors la forme représentée dans la figure au-dessous :

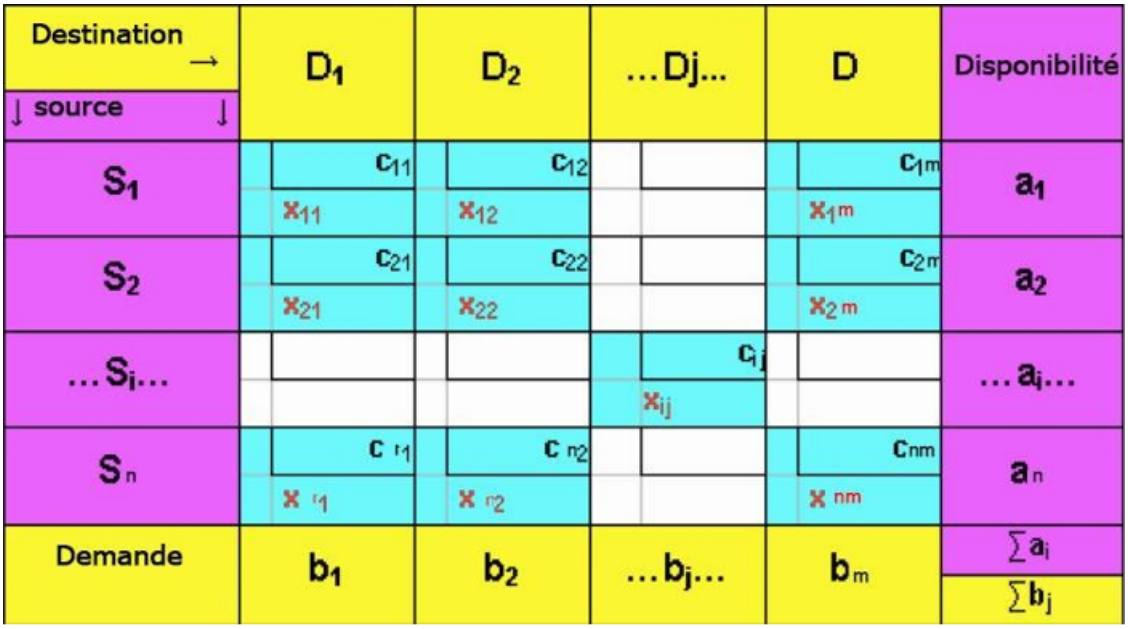


FIGURE 3.1 – Représentation d'un tableau de transport

◇ Un circuit dans un tableau de transport équilibré est une séquence de cases (cellules) tel qu'il se termine à la même case d'où il a commencé, et chaque cellule de la séquence est connectée à celle qui suit soit par une ligne ou une colonne dans le tableau.[1]

◇ Une allocation : c'est le nombre d'unités d'articles transporté d'une source à une destination enregistré dans une cellule du tableau de transport.[1]

3.8 Réseau de transport

Graphiquement, le problème du transport est souvent visualisé comme un réseau avec m nœuds sources, n nœuds destinations et un ensemble de m×n « arcs orientés ».
Dans la figure suivante on a une représentation d'un réseau de transport, où il y a $S_1,\ldots,S_m$ sources et $D_1,\ldots,D_n$ destinations. Les arcs montrent des flux de transport de source vers destination.
Chaque destination est liée à chaque source par un arc, le nombre $(c_{11},\ldots,c_{mn})$ sur chaque arc représente le coût de transport sur cette route.

Les problèmes avec la structure ci-dessous se posent dans de nombreuses applications. Par exemple, les sources pourraient représenter des entrepôts, des puits, ... etc. les destinations pourraient représenter des populations, des clients, ... etc.

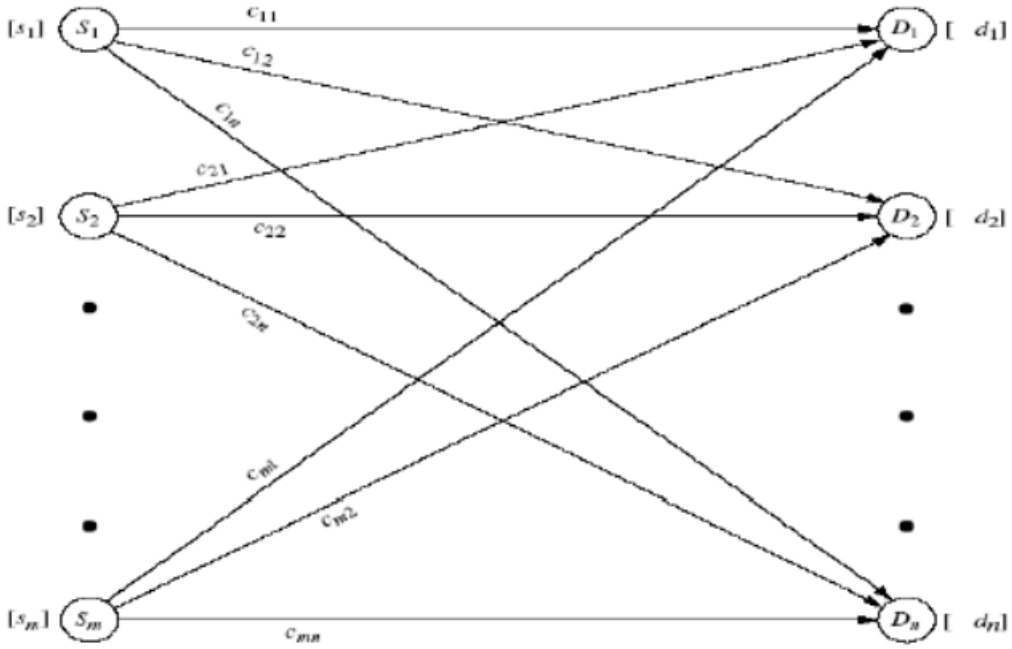


FIGURE 3.2 – Illustration d'un réseau de transport

3.9 Dégénérescence en problème de transport

La dégénérescence survient dans un problème de transport lorsque le nombre de cases occupées est inférieur à la somme du nombre de lignes et du nombre de colonnes, moins un (m + n - 1).
La dégénérescence peut être observée soit lors de l'attribution initiale, lorsque la première entrée dans une ligne ou une colonne satisfait à la fois aux exigences de cette ligne et de cette colonne, soit lors de l'application d'une méthode de résolution du problème de transport, lorsque les valeurs ajoutées et soustraites sont identiques.
Le transport avec m-origines et n-destinations peut avoir (m + n – 1) variables de base positives, sinon la solution de base dégénèrera, Donc chaque fois que le nombre de cases 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.
[9] **Exemple :**
Soit le tableau au-dessous qui représente un tableau de transport avec la solution initiale généré par une des méthode de solution .

Les variables de base sont :

	D_1	D_2	D_3	D_4	Offre
S_1		450			450
S_2			450		450
S_3	400		100	250	750
Demande	400	450	550	250	1650

TABLE 3.1 – Tableau de transport

$$x_{12}, x_{23}, x_{31}, x_{33}, x_{34}$$

On a $a_i < b_j$ tel que m=3 et n=4.
On a aussi 5 variables de base or que , m+n-1=7-1=6 et $5 < 6$ d'où la solution de base est dégénérée.

Chapitre 4

Résolution du problème de transport

Le problème de transport se présente sous forme d'un programme linéaire, pouvant être résolu efficacement par des méthodes du simplexe. Néanmoins, il existe des approches plus spécifiques, basées sur le même principe que le simplexe mais mieux adaptées à ce type de problème. Comme pour toute utilisation du simplexe dans ce contexte, il est nécessaire d'avoir une solution de base initiale. Ce chapitre se concentre donc sur la détermination et l'amélioration de cette solution de base initiale.

4.1 Structure de résolution du problème de transport

Dans un problème de transport impliquant m origines et n destinations, il est notable que la somme des offres (disponibilités) des origines est toujours égale à la somme des demandes des destinations. Ainsi, une solution réalisable est toujours possible. La contrainte (m+n) nième devient alors redondante et peut être supprimée.
En conséquence, une solution de base réalisable pour un problème de transport peut comporter au maximum (m+n–1) variables de base strictement positives. Autrement, la solution risque de devenir dégénérée.
Il est toujours envisageable d'attribuer une solution réalisable initiale à un problème de transport, garantissant ainsi que les exigences des destinations soient satisfaites. Cette attribution peut être effectuée soit par une inspection directe, soit en suivant des règles simples. Nous partons de l'hypothèse qu'au départ, la table de transport est vide, c'est-à-dire que toutes les valeurs x_{ij} sont égales à zéro. Les méthodes les plus élémentaires pour cette allocation initiale seront examinées dans la section suivante. [1]

4.1.1 Plan basique de transport et ensemble basique des cases

Définition 4.1.1 *Un plan basique de transport doit avoir (m+n-1) variables basiques et* $m \times n - (m + n - 1) = (m - 1)(n - 1)$ *variables hors bases nulles. [10]*

Définition 4.1.2 *Un ensemble de cases* $U_B \subset U$ *est dit complet si* $|U_B| = m + n - 1$ *et si aucun autre cycle ne peut être constitué à partir des éléments de* U_B*. [10]*

Définition 4.1.3 *Un plan de transport* $x = \{x_{ij} \,|\, 1 \leq i \leq m, 1 \leq j \leq n\}$ *est appelé basique si (m+n-1) variables forment un ensemble de cases* U_B *complet, et* $x_{ij} = 0$*,* $(i, j) \in U_H, \quad U_H = U \setminus U_B$*.*
L'ensemble U_B *est alors dit ensemble basique de cases, correspondant au plan basique de transport. [10]*

4.1.2 Solution de base réalisable

Définition 4.1.4 *Une solution de base réalisable d'un problème de transport est une solution admissible comportant exactement (m+n-1) flux non nuls, et toutes les autres variables de flux sont égales à 0.*

◇ **Interprétation en théorie de graphe de la notion de solution de base :**

Une solution de base réalisable peut être interprétée comme un ensemble de chemins qui forment un graphe connexe sans cycle (c'est-à-dire un arbre), où chaque nœud est soit une source, soit une destination, et où chaque arête représente un chemin le long duquel la ressource est transportée.
Cette solution satisfait toutes les contraintes du problème et représente donc un point de départ valide pour la recherche de la solution optimale.

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 $\varphi_{ij} = \min(a'_i, b'_j)$ avec $a'_i \neq 0$, $b'_j \neq 0$, et $\varphi_{ij} \in \mathbb{N}$.
Ici : $a'_i =$ disponibilité restante dans x_i.
$b'_j =$ 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 (φ_{ij}) non nuls et nous obtenons une solution de base :
$(m \times n - (m + n - 1) = (m - 1) \times (n - 1))$ flux nuls. [1]

4.1.3 Solution optimale

Une solution réalisable, même si elle n'est pas une solution de base, est jugée optimale si elle réduit au minimum le coût total du transport. [1]

4.1.4 Organigramme de résolution d'un problème de transport

La résolution du problème de transport peut se résumer sous forme de l'organigramme suivant :

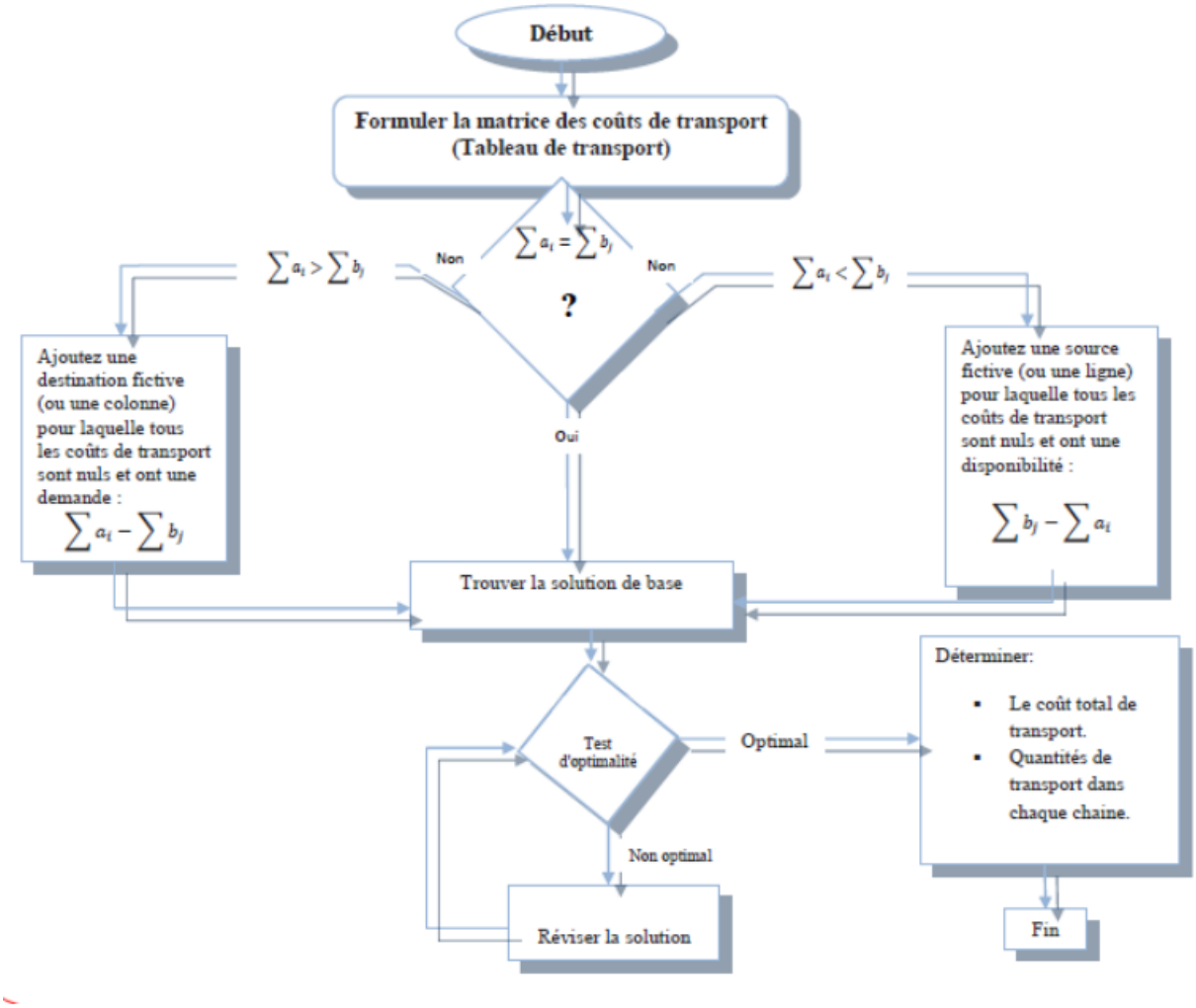


FIGURE 4.1 – Organigramme de résolution du problème de transport

- **Déscription d'organigramme de résolution :**

1. Premièrement, le problème doit être formulé comme un tableau de transport.
2. Ensuite, on vérifie si le modèle de transport est équilibré.
3. Sinon, on ajoute une source ou une destination fictive pour l'équilibrer.
4. On trouve la solution de base initiale du problème de transport.
5. On vérifie si la solution est optimale, si elle l'est on calcule le coût total de transport. Sinon revenir à 4.

4.1.5 Algorithme générale de résolution du problème de transport

Les modèles de transport ne démarrent pas avec toutes les valeurs de décision égales à zéro. Ils nécessitent plutôt une solution de base réalisable initiale.
L'algorithme de résolution d'un problème de transport se résume en ces étapes suivantes :

Étape 1 : Formuler et configurer le problème sous forme matricielle :
La formulation du problème de transport est analogue à celle du problème de programmation linéaire. La fonction objectif consiste à minimiser le coût total du transport, tandis que les contraintes représentent les quantités d'offre disponibles à chaque source et les quantités de demande à chaque destination.

Étape 2 : Obtenir une première solution de base réalisable :
Diverses méthodes peuvent être utilisées pour trouver la solution de base. Parmis ces méthodes on trouve :
- La méthode du Coin Nord-Ouest.
- La méthode du Coût Minimum.
- La méthode d'Approximation de Vogel.

La solution générée par l'une des méthodes mentionnées doit respecter les exigences suivantes :
1. La solution doit être réalisable, c'est-à-dire qu'elle doit respecter toutes les contraintes d'offre et de demande.
2. Le nombre de variables de base doit être égale à m+n-1.

La solution qui satisfait les conditions montionnées ci-dessus est dite solution de base non dégénérée.

Étape 3 : Tester la solution de base initiale pour l'optimalité :
Pour vérifier si la solution de base initiale obtenue si elle est optimale, il faut recourir à l'une des méthodes mentionnées au-dessous :
- Méthode de Stepping Stone.
- Méthode de distribution modifiée.

Si la solution est optimale on arrête, sinon on détermine une nouvelle solution améliorée.

Étape 4 : Mise à jour de la solution
on répéte l'étape 3 jusqu'à atteindre la solution optimale. [1]

4.2 Méthodes de détermination de la solution de base initiale

4.2.1 La méthode du Coin Nord-Ouest

Définition 4.2.1 *C'est la méthode heuristique la plus simple à mettre en œuvre pour obtenir une solution de base réalisable dans un problème de transport. Elle consiste à sélectionner les variables de base à partir du coin nord-ouest (c'est-à-dire le coin supérieur gauche du tableau de transport). [3]*

◇ **Principe de la methode du Coin Nord-Ouest :**

Le principe de la méthode du coin nord-ouest est de commencer à allouer les ressources à partir du coin nord-ouest (coin supérieur gauche) de la matrice de coûts. Ensuite, on remplit progressivement les cases de la première ligne ou de la première colonne jusqu'à ce que l'offre ou la demande de l'une des sources ou destinations soit épuisée, puis on passe à la source ou à la destination suivante, continuant ainsi

jusqu'à ce que toute offre et toute demande soient satisfaites.

◇ **Avantages de la méthode du coin nord-ouest :**

- Simplicité : La méthode du coin nord-ouest est simple à comprendre et à mettre en œuvre, ce qui en fait une option attirante pour des problèmes de taille modérée.
- Rapidité : Elle est généralement plus rapide à exécuter que d'autres méthodes plus sophistiquées, ce qui en fait un choix efficace pour des problèmes simples.
- Aucun calcul supplémentaire : Cette méthode ne nécessite pas de calculs supplémentaires tels que des comparaisons de coûts ou de potentiels réduits, ce qui réduit la complexité. [9]

◇ **Inconvénients de la méthode du coin nord-ouest :**

- Pas toujours optimal : Bien que la méthode du coin nord-ouest fournisse une solution réalisable, elle ne garantit pas toujours la solution optimale. Elle peut conduire à des solutions sous-optimales dans certains cas.
- Dépendance aux données initiales : La qualité de la solution dépend en grande partie de la disposition des coûts initiaux dans la matrice, ce qui peut limiter sa robustesse pour des problèmes complexes. [9]

◇ **Les étapes de la méthode du coin nord-ouest :**

1. Commencer la méthode en choisissant la case se trouvant dans le coin supérieur gauche du tableau de transport.
2. Allouer le plus grand nombre d'unités possible à la case courante.
3. Aller à une case adjacente à la case courante, en se déplaçant soit vers la droite, soit vers le bas.

Revenir à l'étape 2 jusqu'à ce que toutes les exigences du tableau soient respectées.

◇ **Algorithme du coin nord-ouest :**

1) i=1, j=1.
2) $\phi_{ij} = \min(a_i, b_j)$ si $\phi_{ij} = a_i$ passer à (3) sinon passer à (4).
3) Poser $b_j = b_j - a_i$ et $i = i + 1$, si i $\leq$ n passer à (2) sinon FIN.
4) Poser $a_i = a_i - b_j$ et $j = j + 1$, si j $\leq$ n passer à (2) sinon FIN.

De cette manière après (m+n-1) opérations, on trouve (m+n-1) quantités positives x_{ij} affectées à (m+n-1) cases, et les cases restantes auront des quantités nulles $x_{ij} = 0$, on obtiendra ainsi un plan basique de transport.[1]

◇ La méthode du coin nord-ouest est généralement efficace pour trouver une solution initiale à un problème de transport, sa complexité est exprimée en terme de $O(m \times n)$.

4.2.2 La méthode du Coût Minimum

Définition 4.2.2 *La méthode du Coût Minimum est une technique pour obtenir une solution de base réalisable dans un problème de transport en sélectionnant les variables*

de base en fonction du coût unitaire du transport. Cette méthode cherche à améliorer la solution initiale en privilégiant les coûts de transport les moins élevés. [8]

◇ **Pricipe de la méthode du Coût Minimum :**

La méthode du coût minimum débute en attribuant autant que possible à la case avec le coût unitaire de transport le plus bas. Une fois qu'une ligne ou une colonne est complètement satisfaite, les quantités des offres et des demandes sont ajustées en conséquence.
Si une ligne et une colonne sont satisfaites simultanément, une seule est ajustée, comme dans la méthode du Coin Nord-Ouest. Ensuite, trouvez la case non ajustée avec le coût unitaire le plus bas et répétez le processus jusqu'à ce qu'il ne reste plus qu'une seule ligne ou colonne à traiter. [1]

◇ **Les étapes de la méthode du coût minimum :**

1. On identifie la case avec le coût unitaire de transport le plus bas (c_{ij}).
2. S'il y a deux ou plusieurs coûts minimaux, on sélectionne la ligne et la colonne correspondant à la rangée au numéro le plus bas.
3. S'ils apparaissent dans la même rangée, on sélectionne la colonne au numéro le plus bas.
4. On choisi la valeur du x_{ij} correspondante autant que possible en respectant des contraintes de disponibilité et de demande.
5. Si la demande est satisfaite, on supprime la colonne.
6. Si la disponibilité est épuisée, on supprime la ligne.
7. On répéte les étapes 1 à 6 jusqu'à ce que toutes les restrictions soient satisfaites.

◇ De manière générale, la méthode du moindre coût aboutit à une meilleure solution initiale que la méthode du coin nord-ouest.[1]
La complexité de l'algorithme du coût minimum dans le problème de transport est de $O(m \times n \times (m+n))$.

4.2.3 La méthode d'Approximation de Vogel

Définition 4.2.3 *La méthode d'approximation de Vogel (VAM) est une heuristique employée pour résoudre des problèmes d'optimisation liés au transport et à leurs coûts associés, avec pour objectif principal la minimisation de ces coûts. Elle utilise des critères simples pour aborder des problèmes complexes. [4]*

Remarques :
- La méthode d'approximation de Vogel présente un avantage par rapport à d'autres méthodes car, bien qu'elle nécessite plus d'itérations, elle génère généralement une solution de démarrage optimale où proche de celle optimale.
- Si on utilise la solution initiale obtenue par VAM et on procéde à la solution optimale, la quantité de temps nécessaire pour arriver à la solution optimale est considérablement réduite.
- La Méthode d'Approximation de Vogel (VAM) est connue aussi par : l'heuristique de Balas-Hammer /la Méthode de la différence maximale / 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.

◇ **Les étapes de la méthode de VAM :**

1. On calcule les coûts de pénalité qu'on ajoutra au tableau de transport initiale. Ces coûts sont la différence entre les coûts les plus bas de chaque ligne et colonne.
2. On identifie la rangée (ligne ou colonne) avec la plus grande pénalité. Dans cette rangée on choisit la cellule avec le coût le plus bas.
3. Soit la cellule (i, j) avec le coût le plus bas, on lui attribue $\min(a_i, b_j)$, si $\min(a_i, b_j) = a_i$, alors la disponibilité de la i-ème origine est épuisée et la demande à la j- ème destination reste comme $b_j - a_i$ et la i-ème rangée est supprimée de la table.
Encore une fois si $\min(a_i, b_j) = b_j$, la demande de la j- ème destination est remplie et la disponibilité à la i-ème origine reste pour être $a_i - b_j$ et la colonne j est supprimée de la table.
4. On répéte les étapes 1, 2 et 3 avec le tableau restant jusqu'à ce que toutes les origines soient épuisées et toutes les demandes soient satisfaites. [4]

◇ **L'algorithme d'approximation de Vogel :**

Soit δ_l le coût de pénalité sur une ligne, et δ_c le coût de pénalité sur une colonne.
1. Calculer δ_l et δ_c pour chaque ligne et colonne.
2. Sélectionner la ligne ou la colonne avec le δ_l ou δ_c maximum.
3. Choisir dans cette ligne ou colonne la case (i, j) avec le coût le plus faible.
4. Allouer à cette case (i, j) le maximum possible de matière transportable de façon à saturer soit la disponibilité soit la destination.
5. Calculer la quantité résiduelle soit en demande soit en offre.
6. Eliminer la ligne ou la colonne ayant son offre ou sa demande satisfaite.
7. SI le nombre de lignes ou colonnes restant > 2 retour en 2 ;
SINON affecter les quantités restantes aux liaisons. [4]

◇ La méthode d'approximation de Vogel est une méthode intermédiaire trés efficace pour obtenir des solutions initiales dans les problèmes de transport avant d'appliquer des méthodes plus sophistiquées pour atteindre l'optimalité avec une complixité de $O(m \times n \log(m + n))$.

On trouve dans le tableau suivant un résumé sur tous les avantages et les inconvénients des méthodes de détermination des solutions de base initiale.

Méthode	**Avantages**	**Inconvénients**
Coin Nord-Ouest	— Simple à comprendre et à appliquer. — Rapide à mettre en œuvre pour obtenir une solution de base.	— Ne tient pas compte des coûts de transport, donc peut ne pas fournir une solution optimale. — Peut nécessiter des étapes supplémentaires pour trouver la solution optimale. — Peut conduire à une solution initiale loin de l'optimum.
Coût Minimum	— Tient compte des coûts de transport, donc tend à fournir une solution initiale plus proche de l'optimum. — Relativement simple à appliquer.	— Ne garantit pas une solution optimale, peut nécessiter des ajustements supplémentaires. — Peut être plus complexe et prendre plus de temps que la méthode du coin nord-ouest.
Méthode de Vogel	— Prend en compte les différences de coûts, ce qui aide à minimiser les coûts de transport dès le départ. — Souvent fournit une solution initiale plus proche de l'optimale comparée aux autres méthodes. — Équilibre entre simplicité et efficacité en fournissant une bonne approximation de la solution optimale.	— Peut être plus complexe à comprendre et à appliquer que les autres méthodes. — Nécessite plus de calculs pour déterminer les pénalités. — Peut encore nécessiter une optimisation supplémentaire pour atteindre la solution optimale.

TABLE 4.1 – Comparaison des méthodes de détermination de la solution initiale du problème de transport

4.3 Méthodes d'optimisation de solution de base

Pour parvenir à une solution optimale, nous améliorons progressivement la solution de base initiale générée par les méthodes déjà décrites jusqu'à ce qu'aucune réduction supplémentaire du coût de transport ne soit possible. Une solution optimale est atteinte lorsqu'aucun autre ensemble de routes de transport ne peut réduire davantage le coût total du transport.

Ainsi, nous examinons chaque case vide dans le tableau de transport pour voir si elle peut réduire le coût total. Ensuite, nous choisissons la case vide avec le coût d'opportunité

le plus bas pour l'ajouter aux nouvelles routes de transport.

Cette valeur montre combien le coût par unité diminue en augmentant l'allocation de transport dans une case vide. On l'appelle aussi la "case entrante", comme dans la méthode du simplexe. La "case sortante" est la case remplie dans le tableau actuel, dont l'allocation devient nulle quand d'autres unités sont attribuées à la case vide avec la plus grande réduction de coût. Cela signifie que la solution actuelle ne peut plus être améliorée, donc elle est optimale.

⋆ Pour trouver la solution optimale plusieurs méthodes peuvent être utilisées ; et les plus fréquemment utilisées sont :
- La méthode de stepping-stone.
- La méthode de distribution modifiée.

Ces deux méthodes sont utilisées pour résoudre les problèmes de transport dans le domaine de la programmation linéaire, mais elles différent dans leur approche et leur mise en œuvre. Tandis qu'elles génèrent toutes les deux exactement les mêmes résultats « car elles sont des méthodes exactes » et utilisent la même stratégie de test, pour développer la solution améliorée si elle n'est pas optimale.

Le problème de transport peut-être également résolut par la méthode du simplexe, qui est une technique générale utilisée pour résoudre les problèmes d'optimisation linéaire, et que le problème de transport peut-être formulé sous un problème d'optimisation linéaire. Mais il est généralement préférable d'utiliser la méthode de stepping-stone ou la méthode de distribution modifiée en raison de leur efficacité et leur adaptabilité spécifique à ce genre de problème en se concentrant sur l'identification des améliorations potentielles dans le tableau de transport. Tandis que la méthode du simplexe est généralement moins efficace et plus complexe à mettre en œuvre pour ce type de problème.[1]

4.3.1 La méthode de stepping-stone

Définition 4.3.1 *L'algorithme de Stepping-Stone est un processus itératif conçu pour améliorer progressivement une solution de base afin de réduire son coût global. Il peut être appliqué à toute solution de base existante. [8]*

◇ **Déroulement de l'algorithme :**

L'objectif de l'algorithme de stepping-stone est d'améliorer la solution en remplissant les cases vides du tableau de transport, ce qui conduit à une solution optimisée ou améliorée.

◇ **L'algorithme de stepping-stone :**

On appelle :
t_i « u_i » : Potentiel origine.
t_j «v_j » : Potentiel destination.
∂_{ij} : Coût marginal de la liaison $(x_i; x_j)$.

1. Déterminer une solution de base initiale.

2. Calculer les coûts marginaux (coût réduit) pour chaque case vide $\partial_{ij} = c_{ij} - (t_i - t_j)$ avec $t_j - t_i = \theta_{ij}$ la tension de l'arc (i, j), t_i est le potentiel du sommet i de l'arc (i, j).
Ces coûts marginaux représentent le changement de coût total si une unité supplémentaire de flux est allouée à cette case.
Si tous les coûts marginaux sont positifs ou 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 maximale qui peut être déplacé et passer à (4). Alors le gain correspondant est égal au produit de cette quantité par le coût marginale.
4. Retenir la substitution qui réalise la plus grande diminution du coût de transport, l'effectuer et revenir à (2).[1]

◇ **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 ∂_{ij} ?**
Pour chaque case nulle, on calculera ∂_{ij} 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 ∂_{ij} transporté (q) ?**
- On identifie les cases vides où l'ajout d'unités de flux réduit le coût total c'est-à-dire les cases vides dont le ∂_{ij} est négatif, car il ne sert à rien de remplir une case qui fait augmenter le coût parce que l'objectif est de minimiser le coût total.
- Pour remplir une case vide, il faut d'abord vider une case pleine adjacente, ainsi créer un circuit de cases pleines qu'on vide et remplit alternativement.[8]

◇ **Récapitulatif de la méthode de stepping-stone :**
1. Déterminer les chemins d'accès et les variations de coûts pour chaque case vide du tableau.
2.Allouer autant que possible à la case vide présentant la plus forte réduction nette des coûts.
3. Répéter les étapes 1 et 2 jusqu'à ce que toutes les cases vides aient des changements de coûts positifs qui indiquent une solution optimale.[8]

◇ La complexité de l'algorithme de stepping-stone est de $O(m \times n)$.

4.3.2 La méthode de distribution modifiée (MODI)

Définition 4.3.2 *La méthode de distribution modifiée aussi connue par la méthode des pénalités, est une technique utilisée pour trouver la solution optimale d'un problème de transport. Cette méthode est une version améliorante de la méthode de stepping-stone.*

Dans la méthode MODI les équations mathématiques remplacent les chaines de substitution, elle est plus pratique que la méthode de stepping-stone.[3]

◇ **Principe de la méthode MODI :**
Pour appliquer la méthode de distribution modifiée, on commence par la solution de base déjà générer par l'une des méthodes décrite auparavant. Ensuite on calcule les valeurs u_i et v_j pour chaque ligne i et colonne j respectivement dans la table de transport. Et puis on trouve les Δ_{ij}. [6]

◇ **Les étapes de la méthode de distribution modifiée :**
1. Afin de calculer les valeurs u_i et v_j pour chaque ligne et chaque colonne, on définie les équations :

$$u_i + v_j = c_{ij}$$

2. Une fois qu'on a écrit toutes les équations, on assigne u_1 à zéro ou n'importe quel u_i de départ, et on calcule les autres multiplicateurs par la résolution du système d'équations pour toutes les valeurs u_i et v_j.
3. On calcule l'indice d'amélioration Δ_{ij} pour chaque cellule vide (hors base) par l'amélioration de la formule :

$$\Delta_{ij} = c_{ij} - u_i - v_j$$

4. Si on trouve un Δ_{ij} négatif on cherche un cycle fermé impliquant cette cellule et on lui attribue la plus grande quantité possible et on ajuste les quantités transportées pour satisfaire la demande et la disponibilité de chaque rangé.
5. On répète les étapes 2 à 4 jusqu'à ce qu'il n'y ait pas de Δ_{ij} négatif.
6. On calcule le coût total en multipliant chaque allocation (x_{ij}) par son spécifique coût (c_{ij}).[6]

◇ **Résumé de la méthode MODI :**
1. Développer une solution initiale.
2. Calculez les valeurs u_i et v_j pour chaque ligne et chaque colonne.
3. Calculer l'indice d'amélioration Δ_{ij} , pour chaque case vide.
4. Affecter autant que possible à la case vide qui entraînera la plus forte diminution du coût (Δ_{ij} le plus négatif).
Répétez les étapes 2 à 4 jusqu'à ce que toutes les valeurs Δ_{ij}, soient positives où nulles.[8]

◇ **Avantage de la méthode MODI :**
- La méthode MODI est plus rapide et plus efficace que la méthode de stepping-stone pour trouver la solution optimale.
- La simplicité des calculs car elle utilise des équations linéaire simple pour évaluer les coûts de déviation et les potentiels.

◇ La complexité de l'algorithme de MODI est de $O(m \times n)$.

Chapitre 5

Application

Le modèle de transport est un cas particulier du programme linéaire qui peut être efficacement résolu à l'aide de l'algorithme du simplexe. Grâce à la structure spécifique de ce modèle, il est possible de simplifier considérablement l'algorithme, rendant le processus de résolution plus efficace. Dans cette présentation, nous allons démontrer comment simplifier l'algorithme du simplexe en utilisant des méthodes spécifiques pour obtenir et optimiser une solution de base pour un problème de transport. Nous appliquerons ces méthodes à un exemple illustratif afin de clarifier leur fonctionnement.

De plus, nous décrirons la mise en œuvre de ces méthodes sous forme d'un programme informatique, développé en langage C++, capable de résoudre un problème de transport équilibré de manière optimale.

5.1 Exemple illustratif

Une entreprise de distribution alimentaire doit transporter des produits frais de ses centres de distribution à plusieurs supermarchés dans une région.
Les centres de distribution ont des stocks limités, et chaque supermarché a une demande spécifique en produit frais. Le coût de transport varie en fonction de la distance entre les centres de distribution et les supermarchés.

FIGURE 5.1 – Transport de produits frais

a. Cette entreprise dispose de deux centres de distribution dont les capacités sont comme suit :
- Centre de distribution 1 (CD1) : 100 tonnes.
- Centre de distribution 2 (CD2) : 150 tonnes.
b. Les demandes des supermarchés sont :
- Supermarché A (SMA) : 80 tonnes.
- Supermarché B (SMB) : 70 tonnes.
- Supermarché C (SMC) : 50 tonnes.
- Supermarché D (SMD) : 50 tonnes.
c. Les coûts unitaires de transport c_{ij} (en euros par tonne) de chaque centre de distribution CD_i vers chaque supermarché SM_j, sont donnés dans le tableau suivant :

	SMA	SMB	SMC	SMD
CD1	20	24	11	16
CD2	22	28	14	20

TABLE 5.1 – Tableau des coûts unitaires de transport

- **Objectif :**
  Minimiser le coût total de transport tout en satisfaisant la demande des supermarchés et en respectant la capacité des centres de distribution.

5.2 Modélisation mathématique

1. **Variable de décision :**
   x_{ij} : quantité de produit transportée du centre de distribution i au supermarché j.

2. **Fonction objectif :**
   Minimiser Z = 20 x_{11} +24 x_{12} +11 x_{13} +16 x_{14} +22 x_{21} +28 x_{22} +14 x_{23} +20 x_{24}.

3. **Contraintes de capacité des centres de distribution :**

$$x_{11} + x_{12} + x_{13} + x_{14} \leq 100 \tag{5.1}$$
$$x_{21} + x_{22} + x_{23} + x_{24} \leq 150 \tag{5.2}$$

4. **Contraintes de demande des supermarchés :**

$$x_{11} + x_{21} = 80 \quad \text{(SMA)} \tag{5.3}$$
$$x_{12} + x_{22} = 70 \quad \text{(SMB)} \tag{5.4}$$
$$x_{13} + x_{23} = 50 \quad \text{(SMC)} \tag{5.5}$$
$$x_{14} + x_{24} = 50 \quad \text{(SMD)} \tag{5.6}$$

5. **Contrainte de non-négativité :**
   $x_{ij} \geq 0$ $\forall$ i, j.

On retrouve ci-dessous le tableau de transport de notre problème posé :

	SMA	SMB	SMC	SMD	Offre
CD1	20	24	11	16	100
CD2	22	28	14	20	150
Demande	80	70	50	50	250

TABLE 5.2 – Tableau de transport

5.3 Détermination d'une solution de base initiale

5.3.1 Avec la méthode du Coin Nord-Ouest

Dans la méthode du coin nord-ouest, on alloue la plus grande quantité possible à la case située dans le coin supérieur gauche du tableau, puis on continue les allocations vers les cases adjacentes.
Dans notre cas, on alloue à la case (CD1, SMA) autant que possible autrement dit min (CD1, SMA) = min (100, 80) = 80. Donc on alloue a la case (CD1, SMA) 80 tonnes et il nous reste 20 tonnes (100 - 80 = 20) qu'on attribue à la case adjacente (CD1, SMB) car l'offre CD1 n'est pas épuisée.
Cependant, la demande du supermarché SMB n'est toujours pas satisfaite, du coup on attribue à la case adjacente à la case (CD1, SMB) soit la case (CD2, SMB) ce qu'il en manque pour satisfaire la demande de SMB (c'est-à-dire 70 - 20 = 50) soit 50 tonnes.
La quatrième allocation est de 50 tonnes pour la case (CD2, SMC) et la cinquième allocation soit 50 tonnes est attribuée à la case (CD2, SMD).
Le tableau de la solution obtenue par la méthode du coin Nord-Ouest est le suivant :

	SMA	SMB	SMC	SMD	Offre
CD1	20 80	24 20	11	16	100
CD2	22	28 50	14 50	20 50	150
Demande	80	70	50	50	250

TABLE 5.3 – Tableau de la solution initiale par la méthode coin nord-ouest

On a $(m + n - 1) = (2 + 4 - 1) = 5$ le nombre de variables de base nécessaires, donc la solution obtenue par la méthode du coin nord-ouest est une solution de base réalisable puisque il y a 5 allocation dans la solution générée.
Comme le nombre de cases occupées est 5 et est égal à $(m + n - 1)$, la condition est satisfaite, la solution initiale est complète car toutes les exigences sont satisfaites.
La solution de départ composée de 5variables de base est :
CD1 -> SMA (x_{11}) = 80 tonnes.
CD1 -> SMB (x_{12}) = 20 tonnes.

CD2 -> SMB (x_{22}) = 50 tonnes.
CD2 -> SMC (x_{23}) = 50 tonnes.
CD2 -> SMD (x_{24}) = 50 tonnes.
Le coût total de transport est calculé en évaluant la fonction objectif :

$$\begin{aligned} Z &= 20 \times x_{11} + 24 \times x_{12} + 28 \times x_{22} + 14 \times x_{23} + 20 \times x_{24} \\ &= 20 \times 80 + 24 \times 20 + 28 \times 50 + 14 \times 50 + 20 \times 50 \\ Z &= 5180 \text{ euros} \end{aligned}$$

5.3.2 Avec la méthode du Coût Minimum

Dans la méthode du coût minimum on commence par effectuer autant que possible à la case ayant le coût unitaire de transport le plus petit.
Comme on le voit dans le tableau au-dessous :
Tableau de la 1ére itération.

	SMA	SMB	SMC	SMD	Offre
CD1	20	24	11 50	16	100
CD2	22	28	14	20	150
Demande	80	70	50	50	250

TABLE 5.4 – Tableau de la 1ére itération de la méthode du coût minimum

La case (CD1, SMC) est la case avec le coût unitaire de transport le plus petit.
La demande de SMC est satisfaite, donc on supprime la colonne SMC.
L'offre de CD1 n'est pas épuisée il en reste 50 tonnes.
Le tableau de la deuxième itération :

	SMA	SMB	SMC	SMD	Offre
CD1	20	24	11 50	16 50	100
CD2	22	28	14	20	150
Demande	80	70	50	50	250

TABLE 5.5 – Tableau de la 2éme itération de la méthode du coût minimum

Ici la case avec le coût unitaire le plus petit est (CD1, SMD).
La ligne CD1 est épuisée donc on la supprime ainsi que la colonne SMD.
Le tableau de la 3éme itération.

	SMA	SMB	SMC	SMD	Offre
CD1	[20]	[24]	[11] 50	[16] 50	100
CD2	[22] 80	[28]	[14]	[20]	150
Demande	80	70	50	50	250

TABLE 5.6 – Tableau de la 3éme itération de la méthode du coût minimum

La case avec le coût unitaire de transport le plus petit est (CD2, SMA) on lui attribue une allocation de 80 tonnes car c'est le plus possible et que CD2 offre 150 tonnes.
Le tableau de la 4éme itération :
Dans cette itération, il reste 70 tonne de l'offre de CD2.
La demande de SMB est de 70 tonnes, ce qu'il fait qu'on alloue 70 tonnes à la case (CD2, SMB).

	SMA	SMB	SMC	SMD	Offre
CD1	[20]	[24]	[11] 50	[16] 50	100
CD2	[22] 80	[28] 70	[14]	[20]	150
Demande	80	70	50	50	250

TABLE 5.7 – Tableau de la 4éme itération de la méthode du coût minimum

On obtient la solution initiale ayant 4 variables de base, tandis que dans notre problème de transport on a m+n-1 = 5, ce qui fait que notre solution obtenue est dégénérée.
Et pour remédier à cela on doit ajouter une allocation fictive pour compléter la solution de base étant donné que toutes les demandes sont satisfaites.
On choisit une allocation fictive, la case (CD1, SMB) est une bonne candidate pour une allocation fictive de 0 unités.
Tableau finale avec l'allocation fictive :

Avec cette allocation fictive on obtient une solution initiale avec 5 variables de base ; comme suit :
CD1 -> SMC (x_{13}) : 50 unités
CD1 -> SMD (x_{14}) : 50 unités
CD2 -> SMA (x_{21}) : 80 unités
CD2 -> SMB (x_{22}) : 70 unités
CD1 -> SMB (x_{12}) : 0 unités (allocation fictive)
Le coût total de transport est calculé en évaluant la fonction objectif :

	SMA	SMB	SMC	SMD	Offre
CD1	[20]	[24] 0	[11] 50	[16] 50	100
CD2	[22] 80	[28] 70	[14]	[20]	150
Demande	80	70	50	50	250

TABLE 5.8 – Tableau de la solution initiale

$$\begin{aligned} Z &= 11 \times x_{13} + 16 \times x_{14} + 28 \times x_{22} + 22 \times x_{21} + 24 \times x_{12} \\ &= 11 \times 50 + 16 \times 50 + 28 \times 70 + 22 \times 80 + 24 \times 0 \\ Z &= 5070 \text{ euros} \end{aligned}$$

5.3.3 Avec la méthode d'approximation de Vogel

On commence par calculer les coûts de pénalités de chaque ligne et colonne comme suit :
- La différence entre le coût de la ligne 1 : $\delta_l = 16 - 11 = 4$.
- La différence entre le coût de la ligne 2 : $\delta_l = 20 - 14 = 6$.
- La différence entre le coût de la colonne 1 : $\delta_c = 22 - 20 = 2$.
- La différence entre le coût de la colonne 2 : $\delta_c = 28 - 24 = 4$.
- La différence entre le coût de la colonne 3 : $\delta_c = 14 - 11 = 3$.
- La différence entre le coût de la colonne 4 : $\delta_c = 20 - 16 = 4$.

	SMA	SMB	SMC	SMD	Offre	Pénalité 1
CD1	[20]	[24]	[11]	[16]	100	4
CD2	[22]	[28]	[14]	[20]	150	6
Demande	80	70	50	50	250	
Pénalité 1	2	4	3	4		

TABLE 5.9 – Tableau initiale de VAM (pénalité)

La méthode d'approximation de Vogel alloue autant que possible à la case de coût minimum dans la ligne ou la colonne avec le plus grand coût de pénalité.
On voit que la 2éme ligne a le plus grand coût de pénalité, et dans cette ligne on choisit la cellule avec le coût unitaire le plus bas comme le montre le tableau au-dessous.

	SMA	SMB	SMC	SMD	Offre	Pénalité 1
CD1	[20]	[24]	[11]	[16]	100	4
CD2	[22]	[28]	[14] 50	[20]	150	6
Demande	80	70	50	50	250	
Pénalité 1	2	4	3	4		

TABLE 5.10 – Tableau de la 1ére allocation avec VAM

On élimine la colonne 3 (SMC) puisqu'elle est satisfaite en on calcule les pénalités 2.

	SMA	SMB	SMC	SMD	Offre	Pénalité 2
CD1	[20]	[24]	[11]	[16]	100	4
CD2	[22]	[28]	[14] 50	[20]	150	2
Demande	80	70	50	50	250	
Pénalité 2	2	4		4		

TABLE 5.11 – Tableau de la 2éme pénalité VAM

La plus grande pénalité est dans la colonne 2 (SMB) et la cellule avec le coût unitaire le plus bas est (CD1, SMB) ce qu'il fait qu'on lui attribue le plus grand nombre possible.

	SMA	SMB	SMC	SMD	Offre	Pénalité 2
CD1	[20]	[24] 70	[11]	[16]	100	4
CD2	[22]	[28]	[14] 50	[20]	150	2
Demande	80	70	50	50	250	
Pénalité 2	2	4		4		

TABLE 5.12 – Tableau de la 2éme allocation VAM

La colonne SMB est satisfaite donc on l'élimine, on poursuit le processus avec la ligne

CD1 qui a la plus grande pénalité.
Il reste une offre de 30 tonnes on l'alloue à la cellule avec le coût unitaire le plus bas c'est-à-dire la cellule (CD1, SMD). On aura le tableau suivant :

	SMA	SMB	SMC	SMD	Offre	Pénalité 2
CD1	[20]	[24] 70	[11]	[16] 30	100	4
CD2	[22]	[28]	[14] 50	[20]	150	2
Demande	80	70	50	50	250	
Pénalité 2	2			4		

TABLE 5.13 – Tableau de la 3éme allocation VAM

La ligne CD1 est épuisée donc on l'élimine, il nous reste que la ligne CD2 on ajoute les valeurs qui manquent pour avoir tout satisfait (offre et demande).
La colonne SMD n'est toujours pas satisfaite il lui en faut 20 tonnes, et la demande de SMA n'est toujours pas satisfaite il en reste 80 tonne en CD2 et c'est la demande de SMA.
Ainsi, on obtient le tableau final suivant :

	SMA	SMB	SMC	SMD	Offre
CD1	[20]	[24] 70	[11]	[16] 30	100
CD2	[22] 80	[28]	[14] 50	[20] 20	150
Demande	80	70	50	50	250

TABLE 5.14 – Tableau de la solution initiale avec VAM

La solution initiale obtenue par la méthode de Vogel est une solution de base réalisable car on 5 variables de base qui est égale au nombre variable de base de notre problème.
La solution obtenue est :
CD2 -> SMC (x_{23}) : 50 unités
CD1 -> SMB (x_{12}) : 70 unités
CD1 -> SMD (x_{14}) : 30 unités
CD2 -> SMD (x_{24}) : 20 unités
CD2 -> SMA (x_{21}) : 80 unités

$$
\begin{aligned}
Z &= 14x_{23} + 24x_{12} + 16x_{14} + 20x_{24} + 22x_{21} \\
&= 14 \times 50 + 24 \times 70 + 16 \times 30 + 20 \times 20 + 22 \times 80 \\
Z &= 5020 \text{ euros}
\end{aligned}
$$

5.4 Optimisation de la solution de base initiale

La solution initiale qu'on va utiliser dans la suite pour trouver la solution optimale est la solution obtenue par la méthode de Vogel car elle a le coût total minimum des trois méthodes utilisées.

5.4.1 La méthode de stepping-stone

La méthode du stepping-stone est une technique de la programmation linéaire qui vise à déterminer si l'utilisation d'une cellule sans allocation pourrait diminuer le coût total du transport.
Dans le contexte de la résolution de base d'un problème de transport, cette méthode examine si l'utilisation potentielle de routes de transport inutilisées actuellement (cellules vides) pourrait entraîner une réduction du coût total du transport.
On utilise la solution de base générée par la méthode du coin nord-ouest.

	SMA	SMB	SMC	SMD	**Offre**
CD1	[20] 80	[24] 20	[11]	[16]	**100**
CD2	[22]	[28] 50	[14] 50	[20] 50	**150**
Demande	**80**	**70**	**50**	**50**	**250**

TABLE 5.15 – Tableau de la solution initiale avec coin nord-ouest

Dans le tableau 5.15 on a 3 cellules vides (CD1, SMC), (CD2, SMA) et (CD2, SMB) qui représentent des itinéraires inutilisées.
Dans la méthode du stepping-stone, notre première action consiste à examiner les cellules vides afin de déterminer si l'utilisation de l'une d'entre elles permettrait de réduire le coût total. En cas de découverte d'une telle possibilité, nous allouons alors autant que possible des ressources correspondantes.
Boucle férmée et coûts marginaux pour chaque cellule vide :

- Cellule vide (CD1, SMC) :
  Boucle (CD1, SMC) -> (CD2-SMC) -> (CD2-SMB) -> (CD1-SMB) -> (CD1-SMC)
  Coût de la boucle : CD1 -> SMC = 11 - 14 + 28 - 24 = 1.
  Coût net est 1 et est positif ce qui fait ce n'est pas une bonne option.

- cellule vide (CD1, SMD) :
  Boucle : (CD1-SMD) -> (CD1-SMB) -> (CD2-SMB) -> (CD2-SMD) -> (CD1-SMD)
  Coût de la boucle : CD1 -> SMD = 16 - 24 + 28 - 20 = 0.
  Coût net est 0 et est non négatif, donc pas bonne option.
- Cellule vide (CD2, SMA) :
  Boucle : CD2-SMA -> CD1-SMA -> CD1-SMB -> CD2-SMB -> CD2-SMA
  Coût de la boucle : CD2 -> SMA = 22 - 20 + 24 - 28 = -2.
  Coût net est -2, il est négatif donc c'est une bonne option.

La cellule vide (CD2, SMA) a le coût net le plus négatif (-2) on va donc ajuster cette cellule.
On ajoute 50 unités à la cellule CD2 -> SMA et on ajuste les autres cellules de la boucle.
On augmente CD2 -> SMA de 50 unités.
Puis on réduit CD1 -> SMA de 50 unités.
Ensuite, on augmenter CD1 -> SMB de 50 unités.
Et enfin, on reduit CD2 -> SMB de 50 unités.
On aura les nouvelles allocations suivantes :
CD1 -> SMA : 30 unités (80 - 50)
CD1 -> SMB : 70 unités (20 + 50)
CD1 -> SMC : 0 unités
CD1 -> SMD : 0 unités
CD2 -> SMA : 50 unités (0 + 50)
CD2 -> SMB : 0 unités (50 - 50)
CD2 -> SMC : 50 unités
CD2 -> SMD : 50 unités
On obtient le tableau suivant :

	SMA	SMB	SMC	SMD	**Offre**
CD1	[20] 30	[24] 70	[11]	[16]	**100**
CD2	[22] 50	[28]	[14] 50	[20] 50	**150**
Demande	**80**	**70**	**50**	**50**	**250**

TABLE 5.16 – Tableau de transport après mise à jour de la solution

Avec cette solution on aura un coût total de transport de :
Z = 5080 euros.

La nouvelle solution après ce premier ajustement est plus optimisée.
On doit vérifier s'il reste des ajustements possibles.
On vérifie les nouvelles cellules vides, qui sont (CD1, SMC), (CD1, SMD) et (CD2, SMB).
Calcul des coûts nets pour les Nouvelles cellules vides :

On recommençe la procédure pour les nouvelles cellules vides jusqu'à ce qu'il n'y ait plus de coûts marginaux négatifs.

- Cellule vide (CD1, SMC)
  Boucle : CD1-SMC -> CD1-SMA -> CD2-SMA -> CD2-SMC -> CD1-SMC
  Coût de la boucle : $11 - 20 + 22 - 14 = -1$
  Coût net est -1 (très bonne option car il est négatif)
- Cellule vide (CD1, SMD)
  Boucle : CD1-SMD -> CD1-SMB -> CD2-SMB -> CD2-SMD -> CD1-SMD
  Coût de la boucle : $16 - 24 + 28 - 20 = 0$
  Coût net : 0 il est non négatif donc ce n'est pas une bonne option.
- Cellule vide (CD2, SMB)
  Boucle : CD2-SMB -> CD1-SMB -> CD1-SMA -> CD2-SMA -> CD2-SMB
  Coût de la boucle : 28 - 24 + 20 - 22 = 2
  Coût net est de 2 ce qui fait ce n'est pas une bonne option car il est positif.

La cellule (CD1, SMC) est un meilleur choix car elle a un coût marginal négatif.
On va donc ajuster cette cellule, et on obtient le résultat dans le tableau de transport suivant :

	SMA	SMB	SMC	SMD	**Offre**
CD1	[20]	[24] 70	[11] 30	[16]	**100**
CD2	[22] 80	[28]	[14] 20	[20] 50	**150**
Demande	**80**	**70**	**50**	**50**	**250**

TABLE 5.17 – Tableau de transport après mise à jour de la solution 2

Avec cette nouvelle solution on obtient un coût total de :
Z = 5050 euros.

On recalcule les coûts marginaux des nouvelles cellules vides.
On a toujours 3 cellules vides : (CD1, SMA), (CD1, SMD) et (CD2, SMB).

- Cellule vide (CD1, SMA)
  Boucle : CD1-SMA -> CD1-SMB -> CD2-SMB -> CD2-SMA -> CD1-SMA
  Coût de la boucle : 20 - 24 + 28 - 22 = 2 Coût net est de 2 positif donc cette option n'est pas bonne.
- Cellule vide (CD2, SMB)
  Boucle : CD2-SMB -> CD1-SMB -> CD1-SMC -> CD2-SMC -> CD2-SMB
  Coût de la boucle : 28 - 24 + 11 - 14 = 1
  Coût net est 1 donc ce n'est pas une bonne option.
- Cellule vide (CD1, SMD)
  Boucle : CD1-SMD -> CD1-SMC -> CD2-SMC -> CD2-SMD -> CD1-SMD

Coût de la boucle : 16 - 11 +14 - 20 = -1
Coût net est négatif (-1) donc c'est bonne option.

La cellule vide (CD1, SMD) est une meilleure option puisque son coût marginal est négatif.
On fait les changement à cette cellule et on obtient le résultat dans le tableau suivant :

	SMA	SMB	SMC	SMD	**Offre**
CD1	[20]	[24] 70	[11]	[16] 30	**100**
CD2	[22] 80	[28]	[14] 50	[20] 20	**150**
Demande	**80**	**70**	**50**	**50**	**250**

TABLE 5.18 – Tableau de transport après mise à jour de la solution

Avec cette solution un coût total de 5020 euros en résulte.
On vérifie les coûts marginaux des nouvelles cellules vides.

- Cellule vide CD1 -> SMA
  Boucle : CD1-SMA -> CD2-SMA -> CD2-SMC -> CD1-SMC -> CD1-SMA
  Coût de la boucle : $20 - 22 + 14 - 11 = 1$
  Coût net : 1. Il estnon négatif, donc pas une bonne option.
- Cellule vide CD1 -> SMC
  Boucle : CD1-SMC -> CD1-SMB -> CD2-SMB -> CD2-SMC -> CD1-SMC
  Coût de la boucle :11 - 24 + 28 - 14 = 6
  Coût net : 6 il est non négatif donc ce n'est pas une boone option.
- Cellule vide CD2 -> SMB
  Boucle : CD2-SMB -> CD1-SMB -> CD1-SMA -> CD2-SMA -> CD2-SMB
  Coût de la boucle : 28 - 24 + 20 - 22 = 2
  Coût net : 2 . Il est positif donc ce n'est pas une bonne option.

Après avoir évalué toutes les cellules vides, on constate qu'aucune cellule vide n'a un coût marginal négatif. Cela signifie qu'il n'est pas possible de réduire davantage le coût total du transport avec cette solution.
Autrement dit, la solution actuelle est optimale selon la méthode de stepping-stone.

5.4.2 La méthode de distribution modifiée

On a le tableau de transport avec la solution initiale obtenue par la méthode de Vogel avec les modifications requises par la méthode de distribution modifiée.

		v_1	v_2	v_3	v_4	
		SMA	SMB	SMC	SMD	**Offre**
u_1	CD1	20	24 70	11	16 30	**100**
u_2	CD2	22 80	28	14 50	20 20	**150**
	Demande	**80**	**70**	**50**	**50**	**250**

TABLE 5.19 – Tableau de transport avec la solution initiale obtenue avec VAM

On applique la méthode de distribution modifiée : 1ére itération : 1. On calcule le coût de transport unitaire pour chaque case ij remplie : $C_{ij} = u_i + v_j$
$C_{12} = u_1 + v_2 = 24$
$C_{14} = u_1 + v_4 = 16$
$C_{21} = u_2 + v_1 = 22$
$C_{23} = u_2 + v_3 = 14$
$C_{24} = u_2 + v_4 = 20$
2. On pose $u_2 = 0$ et on trouve : $v_1 = 22$, $v_3 = 14$, $v_4 = 20$.
On a $u_1 + v_4 = 16 \Rightarrow u_1 + 20 = 16 \Rightarrow u_1 =$ -4
$u_1 + v_2 = 24 \Rightarrow$ -4 + $v_2 = 24 \Rightarrow v_2 = 28$
Le tableau de la solution initiale avec les u_i et v_j :

		$v_1 = 22$	$v_2 = 28$	$v_3 = 14$	$v_4 = 20$	
		SMA	SMB	SMC	SMD	**Offre**
$u_1 =$ -4	CD1	20	24 70	11	16 30	**100**
$u_2 = 0$	CD2	22 80	28	14 50	20 20	**150**
	Demande	**80**	**70**	**50**	**50**	**250**

TABLE 5.20 – Tableau de transport avec la solution initiale obtenue avec VAM

3. On utilise la formule de pénalité pour évaluer les cellules vides :

$$C_{ij} - u_i - v_j = \Delta_{ij}$$

Δ_{11}= C_{11} - u_1 - v_1 = 20 - (-4) - 22 = 2
Δ_{13}= C_{13} - u_1 - v_3 = 11 - (-4) - 14 = 1
Δ_{22}= C_{22} - u_2 - v_2 = 28 - 0 – 28 = 0
Tous les Δ_{ij} sont positifs ou nuls donc la solution obtenue par la méthode de Vogel est

optimale.
Donc Z = 5020 euros.

5.5 Implémentation en C++

C++ est un langage de programmation polyvalent et performant, créé par Bjarne Stroustrup en 1983 comme une extension du C pour inclure des fonctionnalités orientées objet.

Il permet une gestion fine des ressources et de la mémoire, et utilise des concepts comme les classes, l'héritage, le polymorphisme, et les templates pour la programmation générique.

Sa compatibilité avec le C facilite la transition pour les développeurs. C++ est utilisé dans divers domaines, notamment les systèmes d'exploitation, les jeux vidéo, les applications financières et scientifiques.

Bien qu'il soit complexe et demande une gestion manuelle de la mémoire, sa flexibilité et son efficacité en font un choix privilégié pour des applications nécessitant des performances élevées. La figure 5.2 montre l'interface du C++ avant de commencer.

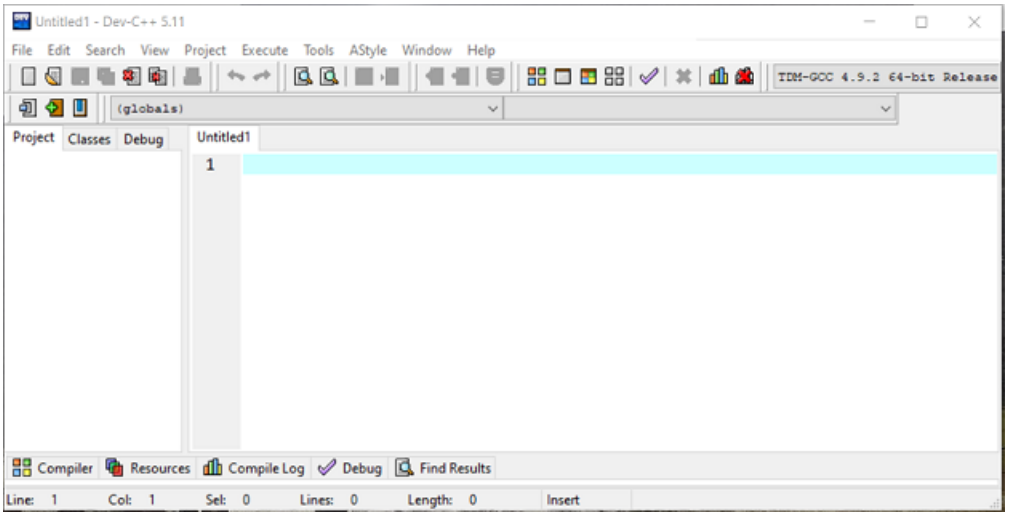


FIGURE 5.2 – L'interface du dev-C++

Pour commencer la programmation sur le C++, si le programme tient dans un seul fichier on'a pas besoin de créer un projet on commence par créer un nouveau fichier en suivant les étapes suivantes :
- On lance le dev-C++ puis on clique sur **file** ;
- Dans **file** on choisit **new** puis **sourc file**.
Où on fait directement ctrl+N.

Pour compiler et executer le programme on clique sur **Execute** et on choisit **Compile and run** pour compiler et executer à la fois.

5.5.1 Execution de la méthode du coin nord-ouest

Pour faire la méthode du coin nord-ouest sur C++ on programme d'abord l'entrée des données du problème de transport et puis on fait l'algorithme du coin nord-ouest pour trouver la solution initiale.

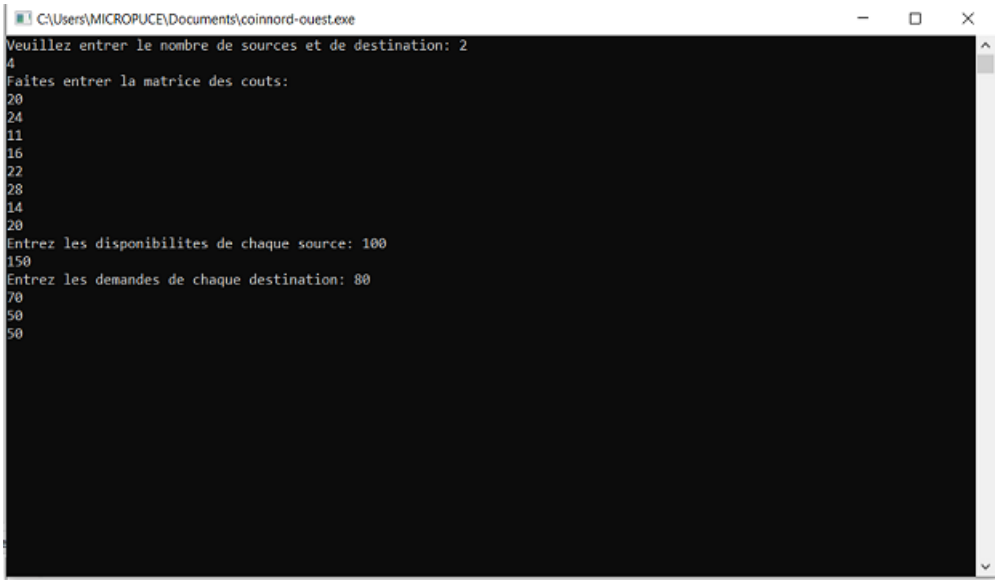


FIGURE 5.3 – Résultat qui apparait après compilation et execution

La figure 5.3 nous motre le résultat après compilation et execution des données du problème qu'on a entré en suivant les instructions.
Et puis qunad on résoud avec l'algorithme du coin nord-ouest on obtient le résultat qui est dans la figure suivante avec la matrice de la solution initiale et le coût de transport :

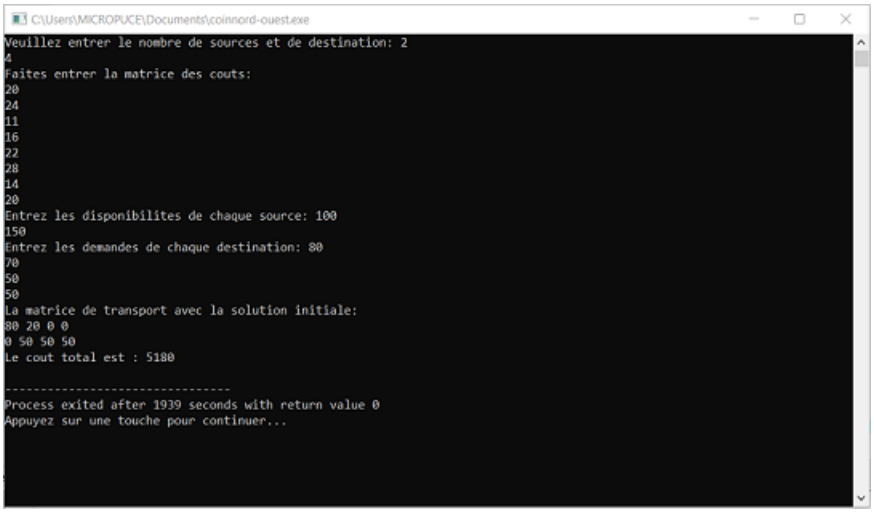


FIGURE 5.4 – Résultat de la méthode du coin nord-ouest sur C++

Certainement, cette méthode est la plus simple à mettre en œuvre et présente une complexité minimale, seulement en O(m.n).

Elle consiste à identifier la variable avec les indices les plus petits où une quantité peut être affectée. Sa simplicité en fait une méthode très populaire. Cependant, la solution obtenue n'est pas nécessairement proche de la solution optimale.

5.5.2 Execution de la méthode du coût minimum

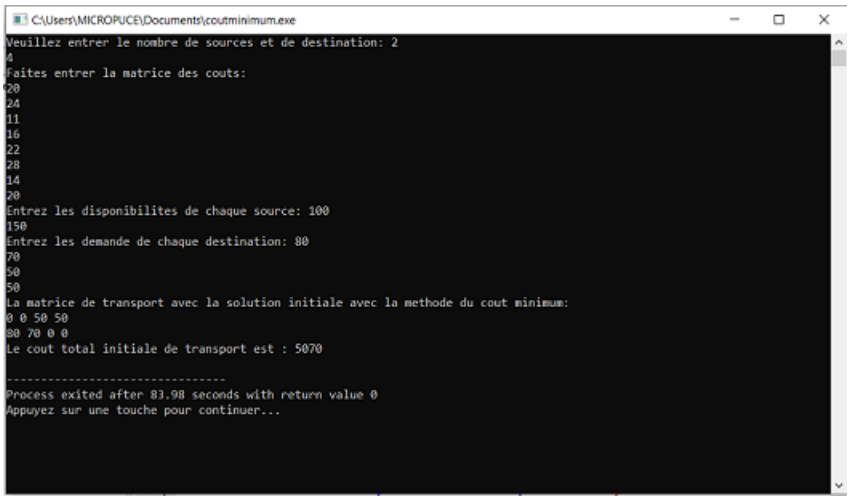


FIGURE 5.5 – Résultat de la méthode du coût minimum sur C++

La figure 5.5 nous donne le résultat obtenu par l'algorithme du coût minimum aprés avoir entrer les données du problème de transport qu'on veut résoudre et son éxecution sur le C++.

La méthode est facile à mettre en œuvre et trouve des solutions de départ proches de la solution optimale. Elle a une complexité de $O((m+n)^2)$. Cette méthode consiste à déterminer le coût le plus bas pour transporter une quantité.

5.6 Comparaison entre les méthodes

Après la résolution de quelques problèmes de transport de différentes tailles par toutes les méthodes de résolution déjà citées dans le chapitre précédent (résolution du problème de transport), on résume les résultats obtenus dans la tableau suivant :

Problème	Taille du problème	MCNO	MCM	VAM	Optimisation de la solution initiale
1	3x3	6600	6460	5920	5920
2	3x4	7750	7650	7350	7225
3	3x5	363	305	290	290
4	3x3	175	148	150	148
5	3x4	670	650	610	610

TABLE 5.21 – Comparaison des méthodes de transport

On a utilisé 3 méthodes de résolution pour trouver la solution de base initiale, la méthode du coin nord-ouest (MCNO), la méthode du coût minimum (MCM) et la méthode

d'approximation de Vogel (VAM) et une méthode pour la solution optimale (la méthode de stepping-stone ou la méthode de MODI).
Ces résultats nous montrent que :
- Quand on utilise la méthode de Vogel la majorité des résultats sont meilleurs (plus optimals) que les deux méthodes MCNO et MCM.
- Une solution de base trouvée par la méthode du coin nord-ouest est généralement loin de la solution optimale par rapport aux autres méthodes.
- La solution obtenue par la méthode du coût minimum est meilleure que celle obtenue par la méthode du coin nord-ouest et elle est plus proche de l'optimale.

Celà signifie que l'utilisation de la méthode de Vogel pour la détermination de la solution de base initiale est meilleure pour réduire les calculs d'optimisation dans un temps optimale et même dans certains problèmes elle donne directement la solution optimale.

Conclusion

Cette étude nous'a permis de se familiariser avec le domaine de la recherche opérationnelle, une discipline qui utilise des méthodes scientifiques pour aider à mieux décider et résoudre les problèmes stratégiques et économiques. Le problème de transport est l'un des problèmes classiques les plus connus dans ce domaine.
Cependant, la complexité et la variation des contraintes économiques associées à ce problème nécessitent la recherche des heuristiques et des métaheuristiques plus efficaces pour sa résolution.
Cela rend difficile de tirer une conclusion définitive sur la meilleure manière de résoudre ce type de problèmes.

Dans ce rapport, nous nous sommes davantage intéressés à la modélisation et à la résolution du problème de transport équilibré à l'aide de différentes méthodes permettant d'obtenir une solution de base réalisable. Nous avons utilisé les méthodes du coin nord-ouest, du coût minimum et de l'approximation de Vogel. Ensuite, nous avons expliqué comment optimiser une telle solution de base initiale en utilisant la méthode de la pierre angulaire (stepping stone) et la méthode de distribution modifiée (MODI). Enfin, nous avons comparé ces méthodes et les avons programmées en langage C++ pour résoudre un problème de transport.

En conclusion, ce travail constitue un point de départ crucial pour aborder les problématiques générales liées au transport.

Bibliographie

[1] BEN-IKEN MOHAMED. ,Problème de transport :Modélisation et résolution, 2017. Autres informations pertinentes.

[2] bibmath. Classes de complexité. https://www.bibmath.net/dico/index.php?action=affiche&quoi=./c/complexite.html, 2024. Date de consultation : 15-06-2024.

[3] DAHOU Mohamed Lamine et CHETBANI Aissa. Méthodes d'optimisation dans les réseaux de transport et applications, 2018. Autres informations pertinentes.

[4] economy pedia. La méthode de vogel. https://economy-pedia.com/11034850-vogel-method, 2024. Date de consultation : 04-05-2024.

[5] F.DROESBEKE, M.HALLIN, CI.LEFERVRE . *LES GRAPHES PAR L'EXEMPLE*. ellipses, Année de publication.

[6] GAANI KHERKHACHE Salsabil. Problème de transport, 2019. Autres informations pertinentes.

[7] Jean-Paul DELAHAYE. La complexité algorithmique. https://www.universalis.fr/encyclopedie/complexite-mathematique/2-la-complexite-algorithmique/, 2009. Date de consultation : 15-06-2024.

[8] KEBBI SOUAD et MESSAOUD SAID . Modélisation et résolution du problème de transport (marchandise), 2020. Autres informations pertinentes.

[9] Université L.AMRANI, cours. problème de transport. Cours. disponible en ligne.

[10] M.AIDEN et B.OUKACHA. *Recherche opérationnelle Programmation Linéaire*. Pages Bleues, Année de publication.

[11] N.BELHARRAT Collectif. *Théorie des Graphes*. Pages Bleues.

[12] Guénette Robert. problème de transport. cours, 2015. disponible en ligne.

[13] techno science. Théorie de la complexité : définition et explication. https://www.techno-science.net/definition/6231.html, 2024. Date de consultation : 15-06-2024.

[14] telecom bretagne. Quelques classes de complexité. http://formations.telecom-bretagne.eu/pyrat/?p=146, 2024. Date de consultation : 15-06-2024.