UNIVERSITE SIDI MOHAMED BEN ABDELLAH
FACULTE DES SCIENCES ET TECHNIQUES
Département des Mathématiques

Licence Sciences et Techniques (LST)

MATHEMATIQUES ET APPLICATIONS

MEMOIRE DE FIN D'ETUDES

Pour l'obtention du Diplôme de Licence Sciences et Techniques

Résolution des problèmes de la recherche opérationnelle :

Problème de transport et problème de chargement

Présenté par :

- **AMZAZ Adil**

Encadré par :

- **Pr ETTAOUIL Mohamed**

Soutenu Le 13 Juin 2016 devant le jury composé de:

- **Pr EL KHOULANI El Idrissi Rachid**
- **Pr CHAKIR Loqman**
- **Pr ETTAOUIL Mohamed**

Année Universitaire 2015 / 2016

FACULTE DES SCIENCES ET TECHNIQUES FES – SAISS
B.P. 2202 – Route d'Imouzzer – FES
212 (0)5 35 61 16 86 – Fax : 212 (0)5 35 60 82 14
Site web : http://www.fst-usmba.ac.ma

Remerciements

Je voudrais commencer par remercier les membres de jury d'avoir accepté et consacré du temps afin de juger ce modeste travail. Je tiens aussi à remercier mon encadrant Monsieur Mohammed ETTOUIL sa disponibilité et ses conseils tout au long de la durée de la préparation du projet.

Mes profonds remerciements vont aussi à Mme Kaoutar ESSENHAJI pour son aide, sa disponibilité et son encouragement.

Je ne passerai pas cette occasion sans exprimer mes sincères remerciements à tous mes professeurs de tout mon cursus pour leurs encouragements et leurs dévouements.

Je tiens à remercier ma famille, mes sœurs et surtout ma sœur Lamiae pour ses soutiens durant tous mes démarches.

Et enfin je ne pourrais finir sans remercié tout spécialement mes parents aux qui les mots ne peuvent exprimer mes sentiments.

Sommaire

Liste des figures

Liste des tableaux

Introduction

La **recherche opérationnelle** peut être définie comme l'ensemble des méthodes et techniques rationnelles orientées vers la recherche du meilleur choix dans la façon d'opérer en vue d'aboutir des résultats visé ou de meilleurs résultats possible, elle vise l'amélioration des opérations des entreprises, des organismes publics ou quotidiens par l'application de l'approche scientifique. Elle repose sur l'utilisation : Des mathématiques appliquées, De l'informatique et des sciences de la gestion. Ainsi la recherche opérationnelle a permis de résoudre plusieurs problèmes réels. Dans ce rapport, on a donné une vue d'ensemble de deux problèmes classique de la recherche opérationnelle :

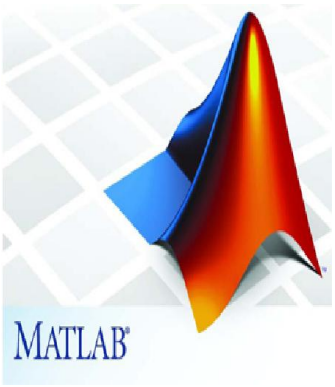
- Problème de transport : ce problème consiste à minimiser le coût de transport total d'un plan d'expédition. Toute les entreprises qu'elle que soit sa taille, son domaine d'activité est amenée à faire face à ce type de problèmes

- Problème de chargement : Ce problème fait partie des 21 problèmes NP-complets identifiés par Richard Karp en 1972. Ces 21 problèmes sont réputés comme les problèmes les plus difficiles en optimisation combinatoire. Un grand nombre d'autres problèmes NP-complets peuvent se ramener à ces 21 problèmes de base. Et il s'applique aux plusieurs domaines comme les systèmes financiers, en cryptographie, et dans le chargement de cargaisons (avions, camions, bateaux…) ;

Notre travail est structuré sur trois principaux chapitres. Le premier, Notion de base, c'est une brève présentation de quelque notion, définition et propriété de la théorie des graphes. Le deuxième chapitre est le problème de transport, ou nous avons posé le problème, ainsi nous avons essayé de le résoudre par la méthode de Stepping-Stone qui nécessite une solution de base prédéfini. Cette dernière a était généré par la méthode de coin Nord-ouest. Ce qui est du troisième chapitre nous avons traité le problème de chargement connu aussi sous le nom du problème du sac à dos, résolu par la méthode de séparation et évaluation (branch and bound) basé sur une solution fourni par la méthode du simplexe. Enfin nous achevant notre travaille par une conclusion.

Technologies utilisés :

MATLAB (abréviation de l'anglais «Matrice de laboratoire».) - Un logiciel pour résoudre le calcul technique et langage de programmation éponyme .Avec MATLAB, on peut facilement produire des calculs matriciels, de visualiser des fonctions mathématiques et des données expérimentales, la mise en œuvre des algorithmes de calcul, la conception interface graphique utilisateur pour des tâches spécifiques, ainsi que par le biais des interfaces spéciales d'interagir avec d'autres langues et programmes programmation**.**



Dev-C++ est un environnement de développement intégré (IDE) permettant de programmer en C et en C++. Développé avec Borland Delphi 6

Langages utilisés

MATLAB - un langage de calcul scientifique de haut niveau, un environnement interactif pour le développement d'algorithmes et d'outils modernes d'analyse des données.

C++ est un langage de programmation compilé, permettant la programmation sous de multiples paradigmes comme la programmation procédurale, la programmation orientée objet et la programmation générique.

Chapitre 1 : Notions de base

Quel que soit le domaine étudié nous sommes confronté à plusieurs situations nécessitant une modélisation concrète afin de pouvoir passer à l'étape de résolution. La représentation graphique est l'une des méthodes les plus utilisés afin de modéliser certain problèmes où situations tel que :

- Les interconnexions routière, ferroviaire ou aériennes entre différentes agglomérations,
- Les liens entre les composants d'un circuit électronique,
- Le plan d'une ville et de ses rues en sens unique,...

La représentation graphique permet de manipuler plus facilement des objets et leurs relations avec une représentation graphique naturelle. L'ensemble des techniques et outils mathématiques mis au point en Théorie des Graphes permettent de démontrer facilement des propriétés, d'en déduire des méthodes de résolution ou encore des algorithmes, ...

I. Les graphes : Définitions et exemples

Un graphe permet de décrire un ensemble d'objets et leurs relations, c'est à dire les liens entre les objets.

- Les objets sont appelés les *nœuds*, ou encore les *sommets* du graphe.
- Un lien entre deux objets est appelé une *arête.*

Définition :

Un **graphe *G*** est un couple ***(V,E)*** où

- ***V*** est un ensemble (fini) d'objets. Les éléments de ***V*** sont appelés les **sommets** du graphe.
- ***E*** est sous-ensemble de ***V*x*V***. Les éléments de ***E*** sont appelés les **arêtes** du graphe.

Exemple **:**

La figure ci-dessous Figure 1) présente un exemple d'un graphe de huit sommets et comportant dix arêtes

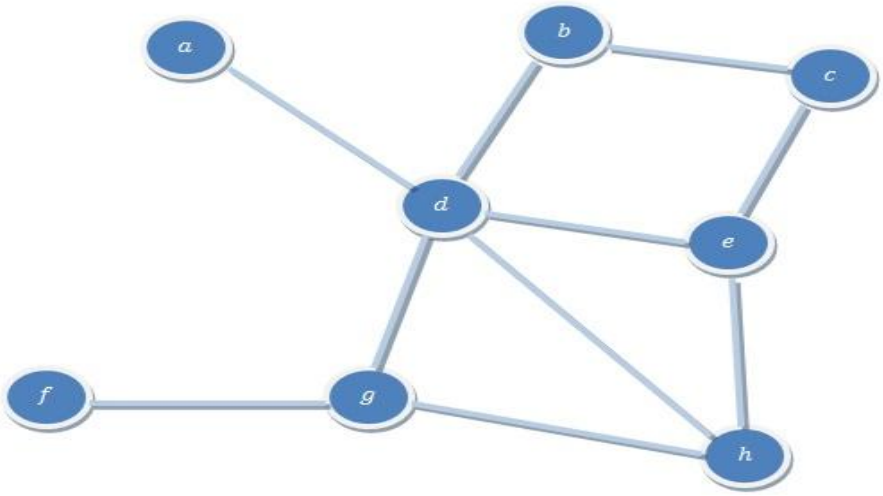


Figure 1 : Graphe de 8 sommets et 10 arêtes

G=(V,E)
V={ a, b, c, d,e, f, g, h }

E={ (a,d),(b,c), (b,d),(d,e), (e,c),(e,h), (h,d),(f,g), (d,g),(g,h) }

Pour caractériser de manière moins locale la structure d'un graphe, il est possible de rechercher des parties remarquables du graphe, en restreignant soit l'ensemble des sommets (sous-graphe), soit l'ensemble des arêtes (graphe partiel).

Définition :

Pour un graphe ***G = (V, E)***

● Un **sous-graphe** de ***G*** est un graphe *H=(W, E(W))* tel que *W* est un sous-ensemble de *V*, et *E(W)* sont les arêtes induites par *E* sur *W*, c'est à dire les arêtes de *E* dont les 2 extrémité sont des sommets de *W*. E(W)={$(x, y) \in E \mid x, y \in W$}.

● Un **graphe partiel** de *G* est un graphe *I= (V, F)* tel que *F* est un sous ensemble de *E*. Un sous-graphe *H* de *G* est entièrement défini (induit) par ses sommets *W*, et un graphe partiel *I* par ses arêtes *F*.

Exemple :*Le sous-graphe **H** induit par l'ensemble **W={b, c, d, g,h}** des sommets (Figure 2).*

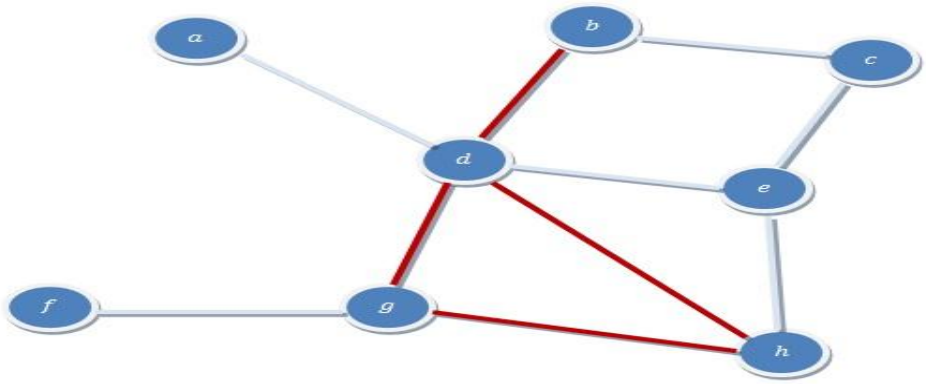


Figure 2 : sous graphe H

Le graphe partiel I défini par l'ensemble F={ (a,d),(b,d), (d,e),(e,h), (h,d),(f,g) } d'arêtes (Figure 3).

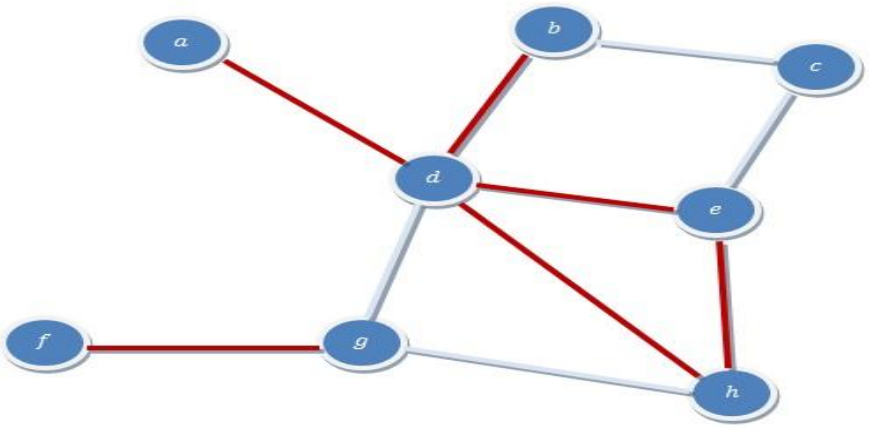


Figure 3 : graphe partiel I

Dans un graphe il est naturel de vouloir se déplacer de sommet en sommet en suivant les arêtes. Une telle marche est appelée une *chaine* ou un *chemin*. Un certain nombre de questions peuvent alors se poser : pour 2 sommets du graphe, existe-t-il un chemin pour aller de l'un à l'autre? Quel est l'ensemble des sommets que l'on peut atteindre depuis un sommet donné? Comment trouver le plus court chemin pour aller d'un sommet à un autre?

II Chemin et cycle

Définition

●Un **chemin** est une liste p=$(x_1, \dots, x_k)$ de sommets telle qu'il existe dans le graphe une arête entre chaque paire de sommets successifs $\forall i = 1, \dots k-1\ (x_i, x_{i+1}) \in E$.

● La **longueur** du chemin correspond au nombre d'arêtes parcourues : ***k-1***.

Exemple : Un chemin de longueur *5* dans le graphe reliant les sommets *f* à *b (Figure 4)*.

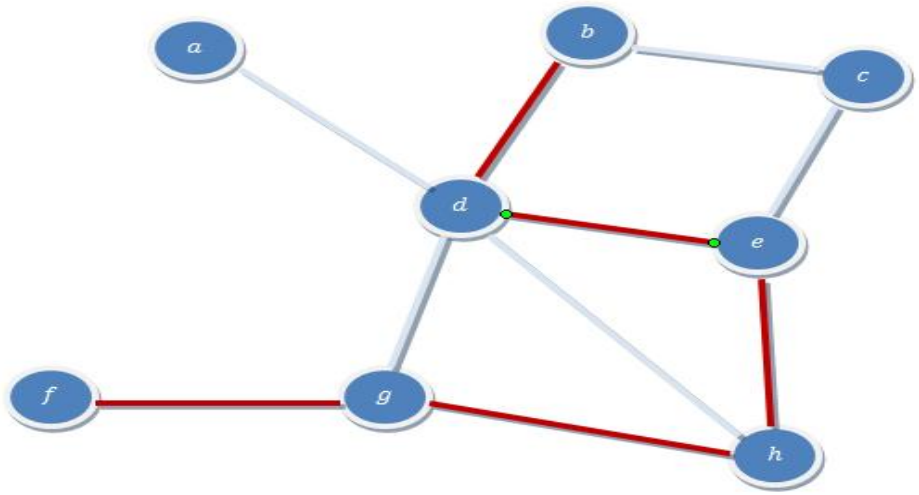


Figure 4 : chemin reliant f à b

Remarque : Il existe bien d'autres chemins pour aller de ***f*** à ***b*** : par exemple ***(f, g, d, b)*** de longueur 3, le chemin ***(f, g, d, h, e, d, b)*** de longueur 6, ou encore ***(f, g, d, h, e, d, h, e, d, b)*** de longueur 9,... ***(d, h, e, d)*** est appelé un cycle. Ce cycle pouvant être emprunté autant de fois que l'on veut, il y a un nombre infini de chemins de ***f*** à ***b***

Définition :

● **Un chemin** p est simple si chaque arête du chemin est empruntée une seule fois.
● **Un cycle** c=$(x_1, \dots, x_k)$ est un chemin simple finissant à son point de départ :$x_1 = x_k$.

Exemple :

Les chemins ***(f, g, d, b)*** et ***(f, g, d, h, e, d, b)*** sont simples. Le chemin ***(f, g, d, h, e, d, h, e, d,b)*** ne l'est pas : le cycle***(d, h, e, d)*** est emprunté 2 fois (Figure 5).

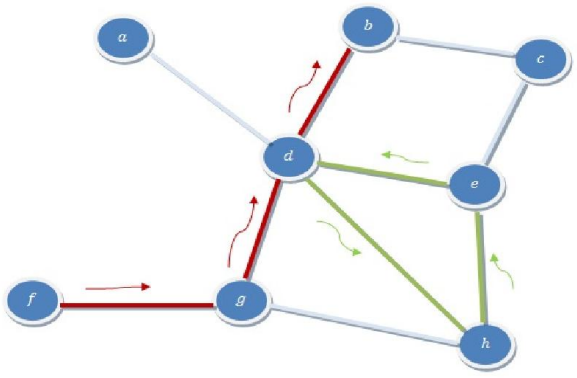
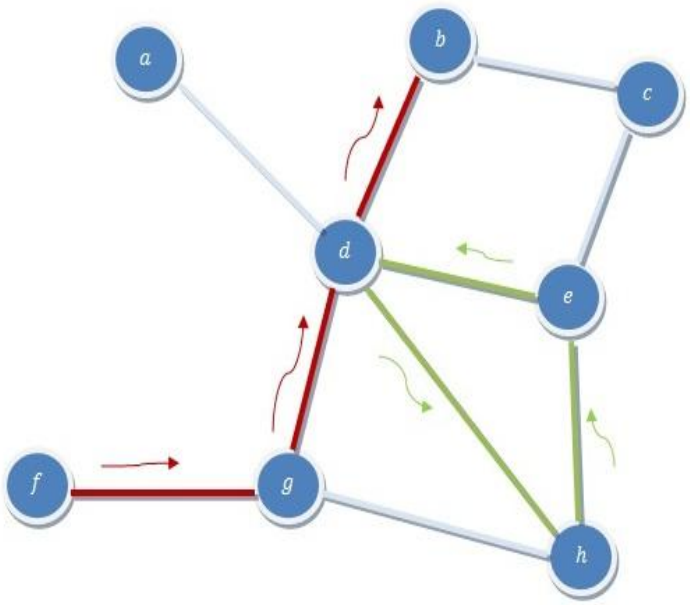


Figure 5 : chemin et cycle

Définition : Chemin élémentaire.

● Un chemin $p = (x_1, \dots, x_k)$ est **élémentaire** si chacun des sommets du parcours est visité une seule fois :$\forall i, j = 1, \dots, k, i \neq j, x_i \neq x_j$.
● Un chemin élémentaire est donc un chemin simple et sans cycle.

Exemple :



Le chemin ***(f, g, d, b)*** est élémentaire, le chemin ***(f, g, d, h, e, d, b)*** ne l'est pas : le sommet ***d*** est visité 2 fois, ce qui crée le cycle ***(d, h, e, d).***

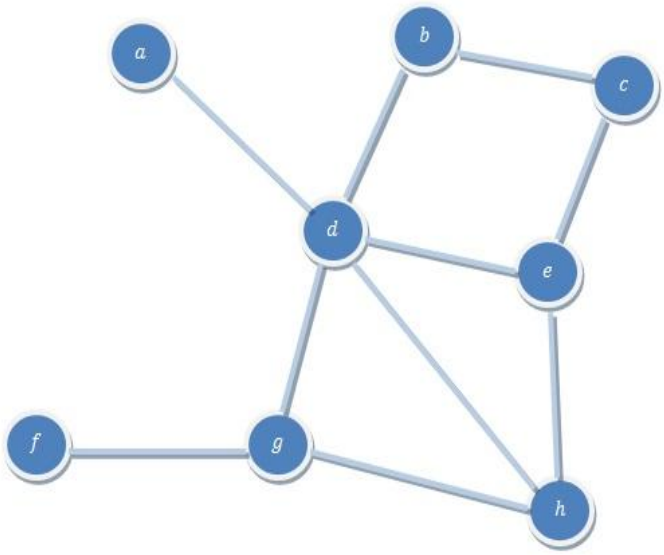
Figure 6 : exemple d'un chemin élémentaire

III Graphe connexe

Définition : Connexité

Un graphe est **connexe** ssi il existe un chemin entre chaque paire de sommets.

Exemple :



Le graphe nous servant d'illustration depuis le début est un graphe connexe.

Figure 7 : Graphe connexe

Que se passe-t-il si le graphe ***G*** n'est pas connexe? Il apparaît alors comme un ensemble de graphes
connexes "mis" les uns à coté des autres. Chacun de ces graphes est un sous-graphe particulier de ***G***,
appelé *composante connexe*. Il est souvent utile de se placer sur les composantes connexes d'un
graphe pour se ramener au cas d'un graphe connexe.

Définition : Composante connexe

Une composante connexe d'un graphe ***G*** est un sous-graphe ***G'= (V',E')*** connexe maximal (pour l'inclusion) : il n'est pas possible d'ajouter à ***V'*** d'autres sommets en conservant la connexité du sous-graphe.

Exemple :

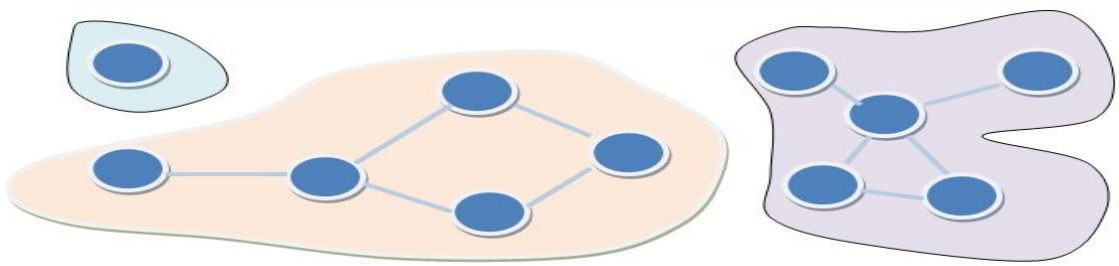


Figure 8 : des composantes connexes

Le graphe ci-dessus possède 3 composantes connexes, dont un sommet isolé.

Remarque :

● Un graphe ne possédant qu'une seule composante connexe est simplement un graphe connexe.
● Un sommet isolé (de degré 0) constitue toujours une composante connexe à lui seul.
● La relation sur les sommets "il existe un chemin entre ..." est une relation d'équivalence (réflexive, symétrique et transitive). Les composantes connexes d'un graphe correspondent aux classes d'équivalences de cette relation.

VI. Arbre

Définition
Un arbre est un graphe connexe sans cycle.

Il existe de nombreuses caractérisations pour les arbres que nous détaillons dans la propriété ci-dessous. Si nous regardons la propriété de connexité, les arbres apparaissent comme des graphes connexes minimaux, au sens où retirer la moindre arête les déconnecte. Si nous regardons les arbres sous l'angle des graphes acycliques, ils sont maximaux, au sens où ajouter la moindre arête crée un cycle. Il existe également une relation très forte entre le

nombre d'arêtes et le nombre de sommets d'un arbre : un arbre ***T=(V,E)*** comporte exactement ***|V|*-1** arêtes.

Caractérisations d'un Arbre

Pour un graphe ***T*** d'ordre ***n***, il y a équivalence entre les propriétés :
1. ***T*** est un arbre
2. ***T*** est un graphe connexe à ***n*-1** arêtes
3. ***T*** est connexe, et la suppression de toute arête le déconnecte
4. ***T*** est acyclique à ***n*-1** arêtes
5. ***T*** est acyclique et l'ajout de toute arête le rend cyclique.

Exemple : le graphe suivant est un arbre

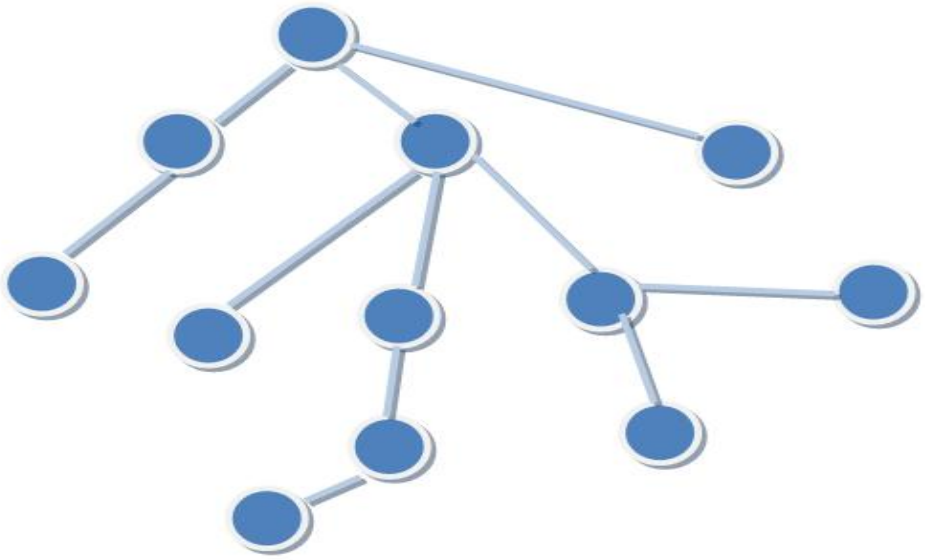


Figure 9 : Graphe d'un arbre

Chapitre 2 : Problème de transport

I. Introduction :

C'est en 1941 que Frank Lauren Hitchcock à formuler pour la première fois le problème de transport, et en suit par le mathématicien français Gaspard Monge en 1781. D'importants développements ont été réalisés dans ce domaine pendant la Seconde Guerre mondiale par le mathématicien et économiste russe Leonid Kantorovich, ce problème consiste à minimiser le coût de transport total d'un plan d'expédition. Le fait de minimiser à la fois la distance totale et le coût de transport fait partie de la théorie des flux de réseaux. Le problème de transport « classique »est en fait un cas particulier d'un problème de flux de réseaux. Le problème du transport est un problème linéaire que peut être représenté sous forme d'un graphe et qu'on peut le résoudre en utilisant les différentes méthodes de résolution des problèmes linéaire qu'on va présenter par la suite.

Dans ce chapitre nous avons posé le modèle mathématique du problème et on a essayé de le résoudre par une méthode graphique : Stepping-Stone. Ainsi nous avons obtenu la solution de base par la méthode de Nord-Ouest.

II. Positionnement de problème :

Le problème de transport peut être défini comme l'action de transporter des marchandises ou des produits fabriqué par m origines(ou usines) vers n destinations(ou clients), d'une manière que le coût total de transport soit minimale. Donc, la résolution d'un problème de transport consiste à organiser le transport de façon à minimiser le coût total de transport.

III. Formulation

Des marchandises doivent être transportées des n usines aux m clients, le transport d'une marchandise de l'origine i à la destination j ayant un coût C_{ij}, La disponibilité a_i en marchandises de l'usine i et la demande b_j de marchandises du client j sont également données.

Variables :

x_{ij} : la quantité à transporter de i vers j $\forall i, j \in \{1, \dots, n\} \times \{1, \dots, m\}$.

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

Contraintes :

- la quantité totale de marchandises partant de i est égale à la disponibilité $\sum_j x_{ij} = a_i$.

- la quantité totale de marchandises reçue par j est égale à la demande $\sum_i x_{ij} = b_j$.

- pour résoudre le problème, il faut impérativement que la demande globale soit égale à la disponibilité globale : $\sum_i a_i = \sum_j b_j$.

- si ce n'est pas le cas, par exemple si $\sum_j b_j > \sum_i a_i$,il faut créer un client fictif avec une demande b et attribuer un coût énorme pour qu'aucune marchandise réelle ne soit transportée vers le client fictif.

III .1. Modèle mathématique :

$$\begin{cases} \min z = \sum_i \sum_j C_{ij} x_{ij} \\ s.c \\ \sum_j x_{ij} = a_i \ \forall j \in \{1, \dots, m\} \\ \sum_i x_{ij} = b_j \ \forall i \in \{1, \dots, n\} \\ a_i \in IN \forall i \in \{1, \dots, n\} \\ b_j \in IN \forall j \in \{1, \dots, m\} \\ x_{ij} \in IN \forall (i,j) \in \{1, \dots, n\} \times \{1, \dots m\} \end{cases}$$

IV La résolution du problème de transport :

La résolution du problème passe par deux étapes essentielles :

• La première c'est de trouvé une solution de base initiale.

• La deuxième étape est de trouvé la solution optimale à partir la solution de base.

IV.1. Recherche d'une solution de base initiale

IV.1.1. Définition :

On appelle solution de base une solution vérifiant les contraintes du problème et qui comporte exactement (m-1)(n-1) flux nuls.

IV.1.2. Méthode du coin Nord-ouest :

C'est la méthode la plus rapide et la plus simple pour déterminé une solution de base car elle ne fait pas entré les coûts de transport c'est a cette raisons la que généralement la solution obtenue par cette méthode est loin de la solution optimale.

Règle du coin Nord-Ouest : On considère à chaque étape, la case la plus Nord à l'Ouest de la matrice des coûts. On part donc de la route (i_1, j_1); on sature soit la ligne i_1 soit la colonne j_1. Puis on recommence sur la sous-grille formée des lignes et des colonnes non saturées. Donc L'idée de la méthode est de remplir au maximum la case du tableau en haut, à gauche, puis compléter sur la ligne ou la colonne (de façon à atteindre l'offre ou la demande) et continuer ainsi à compléter les cases immédiatement à droite et en dessous alternativement. On peut résumer la méthode dans l'algorithme suivant :

IV.1.3. Algorithme :

1- i=1, j=1

2- $Q_{ij} = min\ (a_i;\ b_j)$. Si $Q_{ij} = a_i$passer à (3) sinon passer à (4).

3- Poser $b_j = b_j - a_i$et i=i+1, si i ≤ n passer à (2) sinon fin.

4- Poser $a_i = a_i - b_j$et j=j+1, si j ≤ m passer à (2) sinon fin.

IV.1.4. Application

Une entreprise fabrique un seul produit et possède 3 usines et 4 clients. La production des usines est respectivement de 320, 480 et 340 unités. On doit acheminer vers les clients respectivement 250,300, 260 et 330 unités. Le coût du trajet pour expédier une unité du produit de chacune des industries vers chacun des client est donné dans le Tableau 6, On utilisera les notations suivants :

Puisque les usines sont les origines on notera l'usine numéro i par O_i

Puisque les clients sont les destinations on notera le client j par D_j

Origine	**Destination**				**Production**
	D_1	D_2	D_3	D_4	
O_1	32	34	28	30	320
O_2	28	36	24	26	480
O_3	34	36	28	32	340
Demande	250	300	260	330	1140

Tableau 1 : Tableau des données 1

Dans le coin Nord-Ouest on mettre la plus grand valeur qui va soit satisfaire une demain ou bien épuisé une disponibilité. Dans l'exemple on va épuiser la 1ere demande.

Origine	**Destination**				**Production**
	D_1	D_2	D_3	D_4	
O_1	250 →				320-250=70
O_2					480
O_3					340
Demande	250-250=0	300	260	330	890

Tableau 2 : Etape de l'algorithme de Nord-Ouest

Ensuite on refait la même chose mais sur le nouveau sous tableau qu'on obtient après la 1ere modification.

Origine	Destination				Production
	D_1	D_2	D_3	D_4	
O_1	250	70 ↓			70-70=0
O_2					480
O_3					340
Demande	0	300-70=230	260	330	820

Tableau 3 : Etape 2 de l'algorithme de Nord-Ouest

Origine	Destination				Production
	D_1	D_2	D_3	D_4	
O_1	250	70			0
O_2		230 →			480-230=250
O_3					340
Demande	0	230-230=0	260	330	590

Tableau 4 : Etape 3 de l'algorithme de Nord-Ouest

A la fin on obtient notre solution de base qui vérifie les contraintes de problème

Origine	Destination				Production
	D_1	D_2	D_3	D_4	
O_1	250	70			320
O_2		230	250		480
O_3			10	330	340
Demande	250	300	260	330	1140

Tableau 5 : Solution de base obtenue par l'algorithme de Nord-Ouest

Le coût total de cette solution :

$$Z_1 = 250 \times 32 + 70 \times 34 + 230 \times 36 + 250 \times 24 + 10 \times 28 + 330 \times 32 = 35500DH.$$

Comme on peut représente la solution de base dans le graphe suivant :

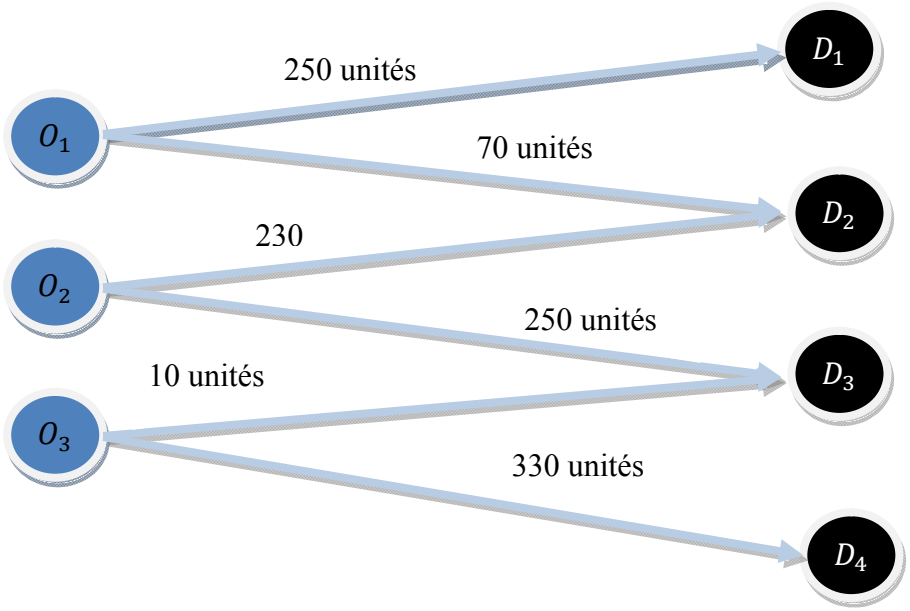


Figure 10 : représentation da la solution de base

IV.2. Recherche d'une solution Optimal

III.2.1 Méthode de Stepping-Stone :

L'algorithme du Stepping-Stone est un algorithme itératif (donc par étapes successives) vise à améliorer (donc faire baisser le coût global) une solution de base.

On peut appliquer l'algorithme à n'importe quelle solution de base. On va l'appliquer, par exemple, à la solution fournie par la méthode du Coin Nord-Ouest.

IV.2.2 Déroulement de l'algorithme

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

Définition :

On appelle cout marginale la quantité (positive ou négative) qui s'ajout au cout globale lorsqu'on veut transporter une unité sur un arc de flux nul.

Algorithme :

1-Déterminer une solution de base.

2-calculer les couts marginaux $\partial_{ij} = C_{ij} - (t_j - t_i)$avec $t_j - t_i est$la tension de l'arc(i,j),t_i s'appelle le potentiel du sommet i de l'arc (i,j).

Si tous les couts marginaux sont positifs ou nuls alors Fin la solution est optimal, ***sinon*** passer à **3**

3-pour tous les négatifs, chercher la chaine de substitution et déterminer la quantité maximale qui peut être déplacé et passer à (4).Alors le gain correspondant est égale au produit de cette quantité par le cout marginale.

4-Retenir la substitution qui réalise la plus grande diminution du cout de transport, l'effectuer et revenir à (2).

a) 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…..en soustrayant de D à O et en ajoutant de O à D.

b) Comment déterminer les δ ?

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

c) Comment déterminer les quantités à transporter(q) ?

On déterminera les quantités qu'on peut ajouter aux cases vides uniquement pour celles dont le δ est négatif ; il ne sert à rien, en effet, de remplir une case qui fait augmenter le coût ! Pour remplir une case vide, il faut diminuer une case pleine….donc constituer un circuit de cases pleines qu'on vide et remplit alternativement

Pour bien comprendre l'algorithme, on va l'appliquer sur la solution trouvé dans la section III.1.4

IV.2.3. Application

Puisque on peut appliquer l'algorithme de Stepping-Stone à n'importe quelle solution de base donc on va considérer la solution de base obtenue par la méthode de coin Nord-Ouest :

Origine	Destination				Production
	D_1	D_2	D_3	D_4	
O_1	250	70			320
O_2		230	250		480
O_3			10	330	340
Demande	250	300	260	330	1140

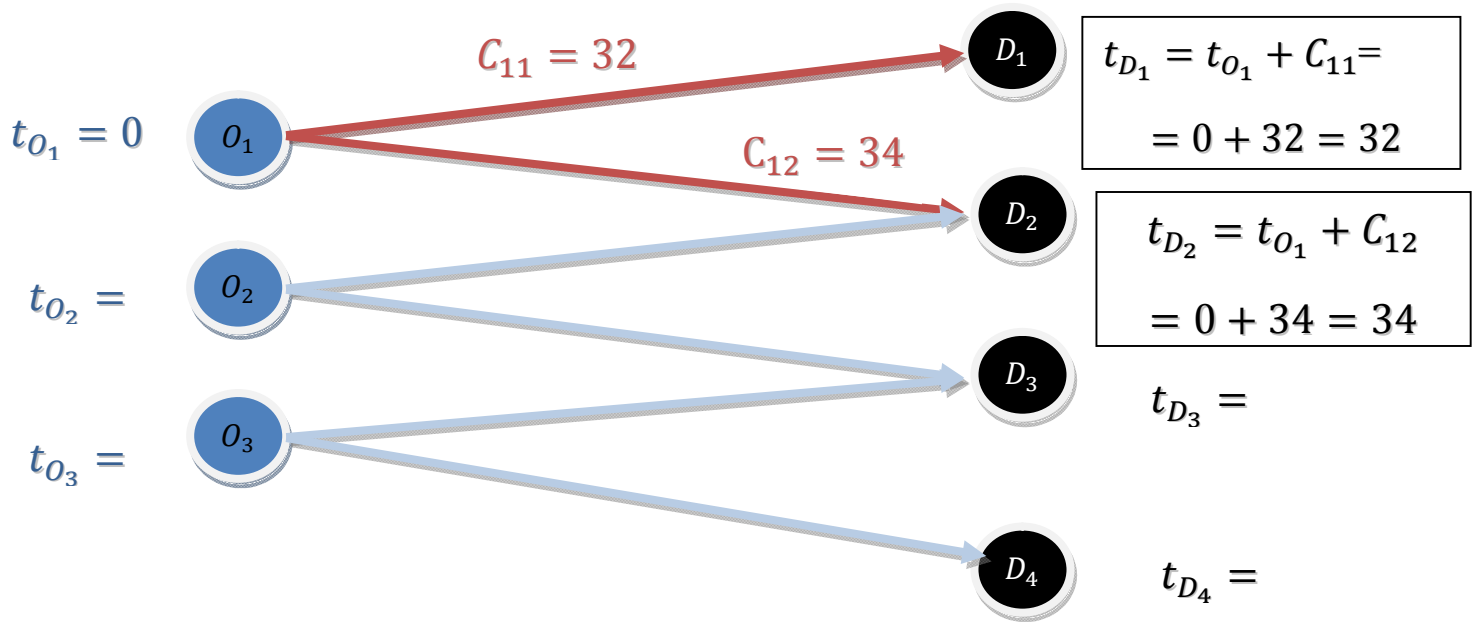
Tableau 6 : Solution de base

a) Calcul des potentiels

On commence par calculer les potentiels de chaque sommet (origine et destination) du graphe biparti correspond à la solution. Pour cela on va donner un potentiel nul à un sommet (par exemple ici $t_{O_1} = 0$), en suit on calcul les autres potentiels en utilisant la méthode suivante :

- $t_j = t_i + C_{ij}$ pour calculer le potentiel d'une destination.
- $t_i = t_j - C_{ij}$ pour calculer le potentiel d'une origine.

Les potentiels calculés dans cette étape sont représentés dans les graphes suivant :



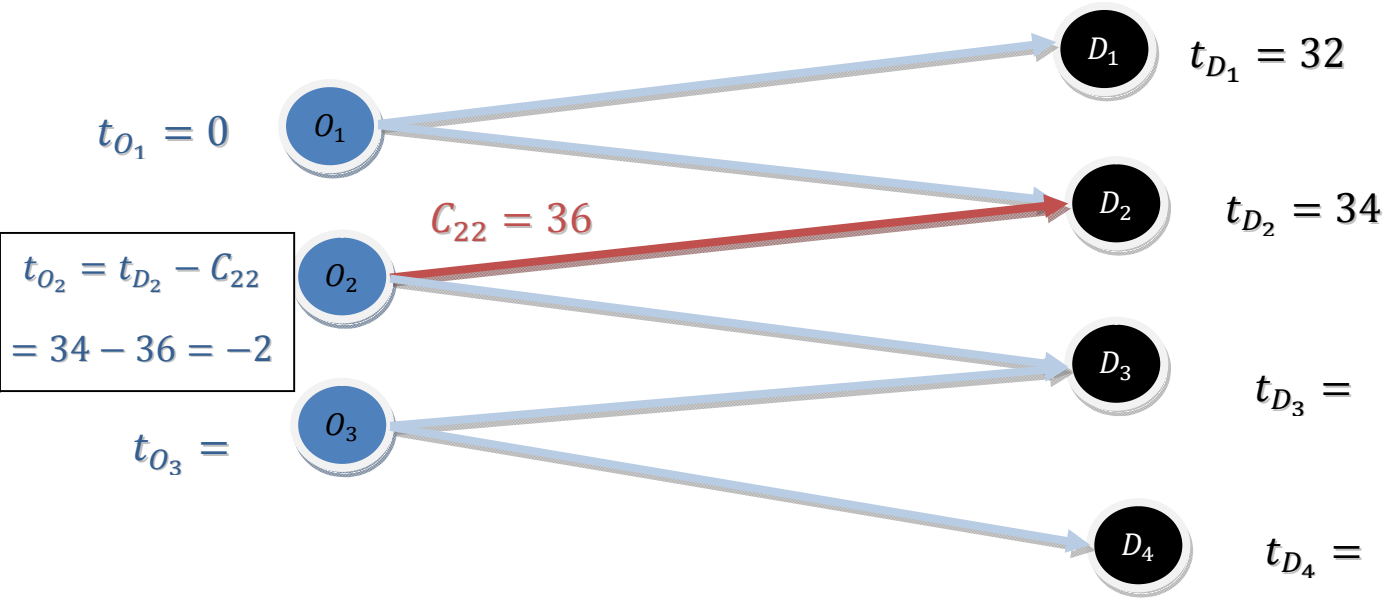


Figure 11 : calculs des potentiels

et ainsi de suite, on obtient les résultats suivants :

$$t_{O_1} = 0 \quad t_{O_2} = -2 \quad t_{O_3} = -6 \quad t_{D_1} = 32 \quad t_{D_2} = 34 \quad t_{D_3} = 22 \quad t_{D_4} = 26$$

b) calcul des couts marginaux∂_{ij}

On utilisera les cases où la quantité transitée est nulle

Origine	Destination				Production
	D_1	D_2	D_3	D_4	
O_1	x	x	$\partial_{13} = C_{13} - (t_{D_3} - t_{O_1})$ =28-(22-0)=6	$\partial_{14} = C_{14} - (t_{D_4} - t_{O_1})$ =30-(26-0)=4	320
O_2	$\partial_{21} = C_{21} - (t_{D_1} - t_{O_2})$ ∂_{21}=28-(32-(-2))=-6	x	x	$\partial_{24} = C_{24} - (t_{D_4} - t_{O_2})$ =26-(26-(-2))=-2	480
O_3	$\partial_{31} = C_{31} - (t_{D_1} - t_{O_3})$ ∂_{31}=34-(32-(-6))=-4	$\partial_{31} = C_{32} - (t_{D_2} - t_{O_3})$ ∂_{31}=36-(34-(-6))=-4	x	x	340
Demande	250	300	260	330	1140

Tableau 7 : Calcul des coûts marginaux

c) Recherche des chaines de substitution et amélioration de la solution

On remarque d'abord que la solution obtenue n'est pas optimal, car les coûts marginaux des arcs $(O_2, D_1), (O_3, D_1), (O_3, D_2)$ et (O_2, D_4)sont négatifs.la solution peut être améliorée grâce à l'un des ces arcs.si on ajoute par exemple l'arc (O_2, D_1) sur le graphe de la solution :

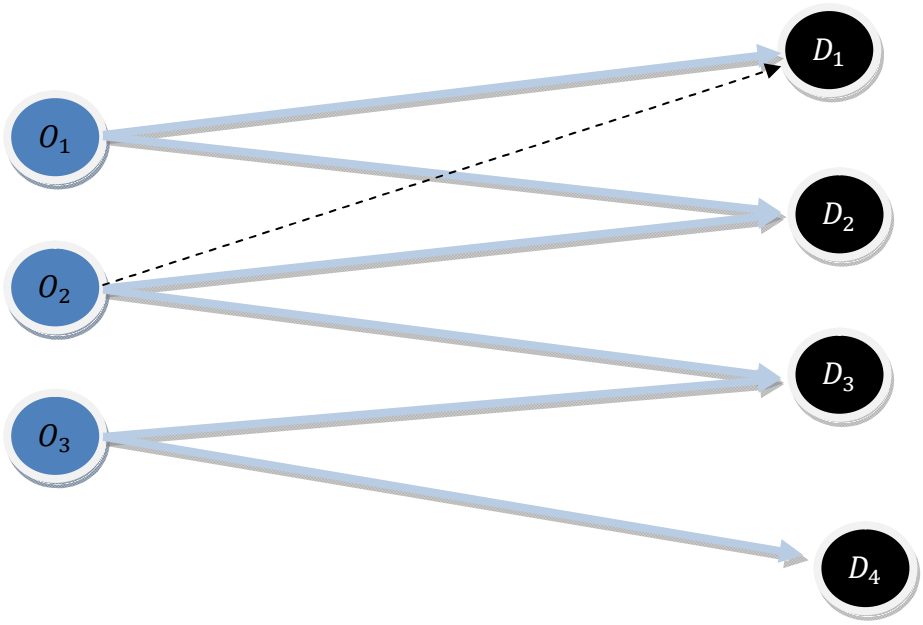


Figure 12 : recherche de la chaine de substitution

On voit que le cycle de l'amélioration est (O_2, D_1, O_1, D_2) :

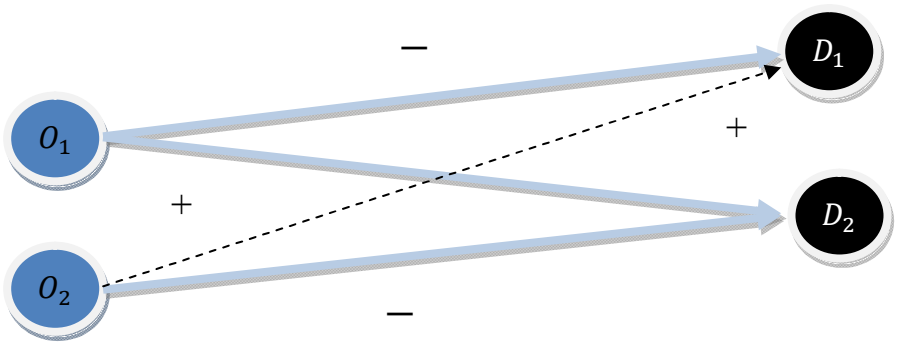


Figure 13 : cycle d'amélioration

La quantité transportée sur chaque arc de signe '−' est :

$Q(O_1, D_1) = 250$ $Q(O_2, D_2) = 230$

D'où la plus grande quantité qui peut être diminuée sur ce cycle est :$Q(O_2, D_2) = 230$.

Le gain de ce cycle est : $230 \times 6 = 1380$.

On fait la même chose aux autres arcs pour déterminer le gain maximal on trouve que :

-Le cycle de l'amélioration de l'arc (O_3, D_1) est $(O_3, D_1, O_1, D_2, O_2, D_3)$

La plus grande quantité qui peut être diminuée sur ce cycle est : 230.

Le gain de ce cycle est :$230 \times 4 = 920$.

-Le cycle de l'amélioration de l'arc (O_3, D_2) est (O_3, D_2, O_2, D_3)

La plus grande quantité qui peut être diminuée sur ce cycle est : 10.

Le gain de ce cycle est :$10 \times 4 = 40$.

-Le cycle de l'amélioration de l'arc (O_2, D_4) est (O_2, D_3, O_3, D_4)

La plus grande quantité qui peut être diminuée sur ce cycle est : 250.

Le gain de ce cycle est :$250 \times 2 = 500$.

D'où la chaine qui offre le plus grand gain est : (O_2, D_1, O_1, D_2)

Les changements de l'amélioration sont présentés dans le graphe suivant :

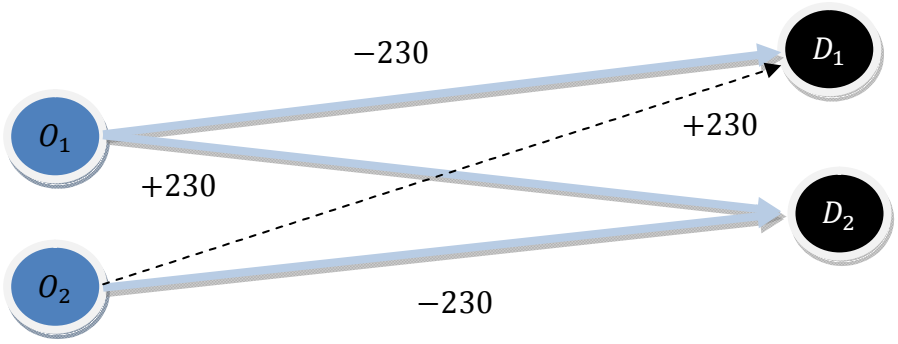


Figure 14 : changements de l'amélioration

La nouvelle solution est :

Origine	Destination				Production
	D_1	D_2	D_3	D_4	
O_1	20	300			320
O_2	230	0	250		480
O_3			10	330	340
Demande	250	300	260	330	1140

Tableau 8 : Nouvelle solution améliorée

Le coût de transport de la nouvelle solution est :

$$Z_2 = 20 \times 32 + 300 \times 34 + 0 \times 36 + 230 \times 28 + 250 \times 24 + 10 \times 28 + 330 \times 32 = 34120$$

On remarque que $Z_2 < Z_1$ et $Z_2 - Z_1 = 1380$ c'est le gain calculé déjà du cycle d'amélioration.

La nouvelle solution est mieux que la première mais on ne sait pas encore s'elle est optimale.

Pour avoir la solution optimale il faut qu'on refasse les étapes a),b) et c) jusqu'à obtenir tous les coûts marginaux positifs, mais à notre époque refaire les mêmes étapes, les mêmes calculs,

nous pousse à réfléchir s'il existe une machine, un logiciel ou un programme qui nous permettra de se débarrasser de ces répétitions.

IV. Résultats numériques

Nous avons implémenté notre travaille sur une machine Windows 7, 2G dans les rames, ainsi nous avons obtenue les résultats suivant :

Le programme est effectué de façon à afficher tous les détails des calculs des étapes a) b) et c) et ne pas pour donner le résultat seulement, on va afficher quelques résultats

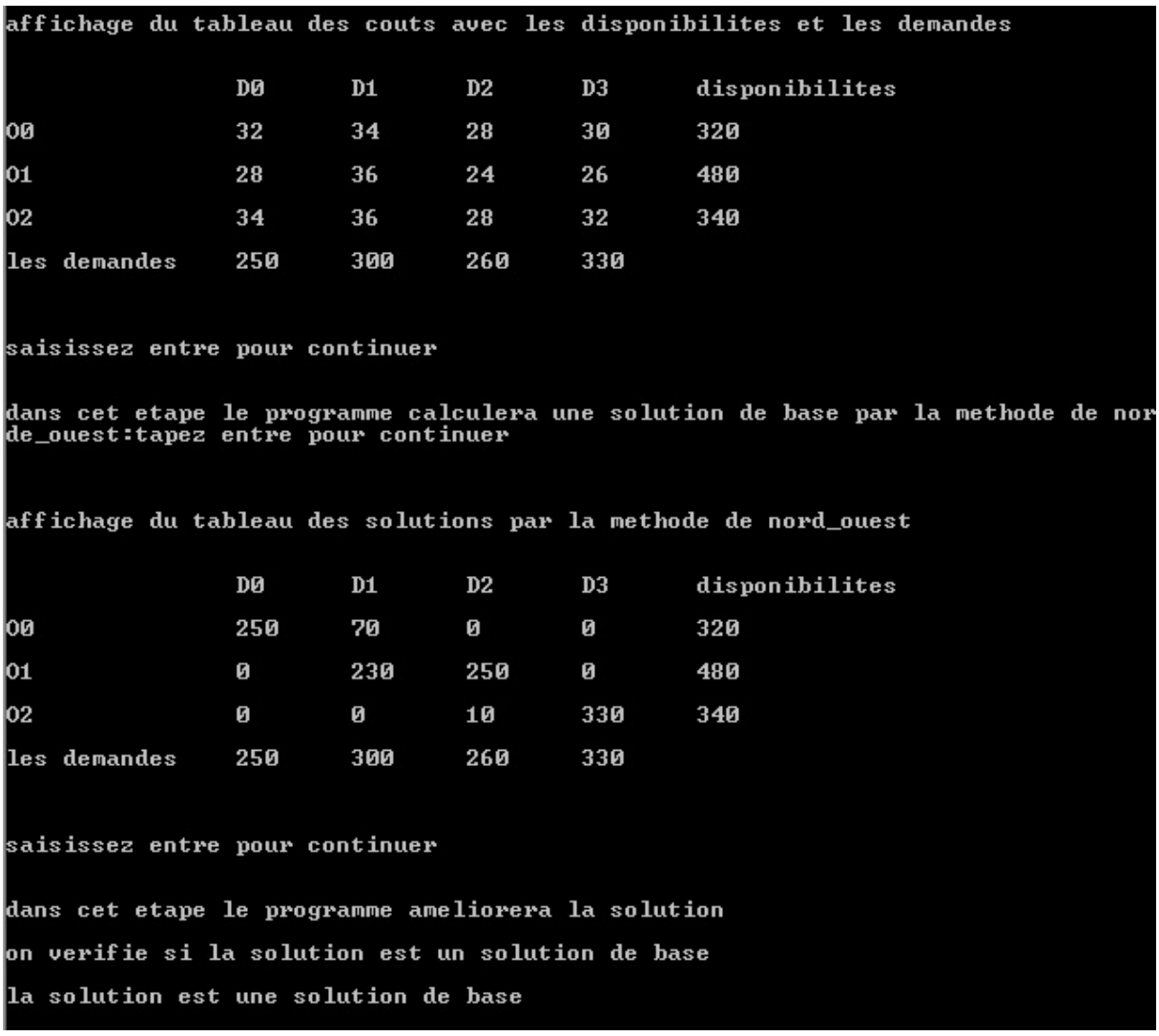


Figure 15 : Solution de base par la méthode de coin Nord-Ouest

Cette figure représente la solution de base de la méthode Nord-Ouest obtenue par notre programme, nous remarquons que le résultat obtenu analytiquement est le même résultat obtenu par notre programme.

Le calcul des premiers coûts marginaux :

```
Affichage des potentiels

pot[O0]=0       pot[O1]=-2      pot[O2]=-6      pot[D0]=32      pot[D1]=34
pot[D2]=22      pot[D3]=26

saisissez entre pour continuer


dans cet etape on va calculer les couts marginaux




Affichage des cout marginaux


rmq:on calcule les couts marginaux pour les cases nulles dans le tableau de la s
olution

x       x       6       4
-6      x       x       -2
-4      -4      x       x
```

La nouvelle solution est :

```
le nouveau solution est :

                D0      D1      D2      D3      disponibilites

O0              20      300     0       0       320

O1              230     0       250     0       480

O2              0       0       10      330     340

les demandes    250     300     260     330


saisissez entre pour continuer


on va verifier si la solution est optimal dc on doit refaire les etapes jusqu a
le calcul des couts marginaux ou on peut destinguer un des cas suivants:
1)optimale c est terminer
2)n'est pas optimale on va refaire les etapes

on verifie si la solution est un solution de base

la solution est une solution de base


saisissez entre pour continuer


Calcul des potentiels:on va utiliser les notations suivants

O0=O0
O1=O1
O2=O2
D3=destination3
D4=destination4
D5=destination5
D6=destination6




Affichage des potentiels

pot[O0]=0       pot[O1]=4       pot[O2]=0       pot[D0]=32      pot[D1]=34
pot[D2]=28      pot[D3]=32
```

Figure 16 : résultats numériques de la méthode de Stepping-Stone

Donc la solution optimale est :

Origine	Destination				Production
	D_1	D_2	D_3	D_4	
O_1	0	300	0	20	320
O_2	250	0	0	230	480
O_3	0	0	260	80	340
Demande	250	300	260	330	1140

Tableau 9 : Solution optimale de l'application de problème de transport

D'un coût égal à :

$$Z_{\text{optimale}} = 33620.$$

Problème de chargement

L'**optimisation combinatoire** est une branche de l'optimisation en mathématiques appliquées et en informatique, également liée à la recherche opérationnelle, l'algorithmique et la théorie de la complexité. On parle également d'**optimisation discrète**.

Un problème d'optimisation combinatoire consiste à trouver la *meilleure* solution dans un ensemble discret dit ensemble des solutions réalisables. En général, cet ensemble est fini mais compte un très grand nombre d'éléments, et il est décrit de manière implicite, c'est-à-dire par une liste, relativement courte, de contraintes que doivent satisfaire les solutions réalisables.

Le problème de chargement en variable binaire fait partie des 21 problèmes NP-complets identifiés par Richard Karp en 1972. Ces 21 problèmes sont réputés comme les problèmes les plus difficiles en optimisation combinatoire. Un grand nombre d'autres problèmes NP-complets peuvent se ramener à ces 21 problèmes de base.

Nous pouvons retrouver le problème du sac à dos dans de nombreux domaines :

- en cryptographie, où il fut à l'origine du premier algorithme de chiffrement asymétrique en 1976 ;
- dans les systèmes financiers, où l'idée est la suivante : étant donné un certain montant d'investissement dans des projets, quels projets choisir pour que le tout rapporte le plus d'argent possible ;
- pour la découpe de matériaux, afin de minimiser les pertes dues aux chutes ;
- dans le chargement de cargaisons (avions, camions, bateaux…) ;
- ou encore, dès qu'il s'agit de préparer une valise ou un sac à dos pour une randonnée

I. Positionnement de Problème :

L'énoncé de ce problème est simple : « Étant donné plusieurs objets possédant chacun un poids et une valeur et étant donné un poids maximum pour le sac, quels objets faut-il mettre dans le sac de manière à maximiser la valeur totale sans dépasser le poids maximal autorisé pour le sac ? ».

II. Formulation mathématique

Toute formulation commence par un énoncé des données. Dans notre cas, nous avons un sac à dos de poids maximal P et n objets. Pour chaque objet i, nous avons un poids p_i et une valeur v_i.

Variables : il nous faut définir les variables qui représentent en quelque sorte les actions ou les décisions qui amèneront à trouver une solution. On définit les variables comme suit :

$x_i = le\ nombre\ des\ objets\ de\ i\ choisi\ parmis\ les\ objets\ de\ i$ à $mettre\ de\ le\ sac$

Contraintes : Ici, il n'y en a qu'une : la somme des poids de tous les objets dans le sac doit être inférieure ou égale au poids maximal du sac à dos.

Cela s'écrit :

$$\sum_{i=1}^{i=n} x_i * p_i \leq P$$

Objectif : maximiser la valeur totale des objets dans le sac.

cela s'écrit :$maximiser \sum_{i=1}^{i=n} x_i * v_i$

Le modèle mathématique :

$$\begin{cases} \max \sum_{i=1}^{i=n} x_i * v_i \\ s.c \\ \sum_{i=1}^{i=n} x_i * p_i \leq P \\ x_i i \in \{1, \dots, n\} estentier \end{cases}$$

III. Méthodes de résolution

Il existe deux grandes catégories de méthodes de résolution de problèmes **d'optimisation combinatoire** : les méthodes exactes et les méthodes approchées. Les méthodes exactes permettent d'obtenir la solution optimale à chaque fois, mais le temps de calcul peut être long si le problème est compliqué à résoudre. Les méthodes approchées, encore appelées heuristiques, permettent d'obtenir rapidement une solution approchée, donc pas nécessairement optimale. Nous allons détailler un algorithme de résolution exacte

III.1 Méthode exacte

Pour trouver la solution optimale, et être certain qu'il n'y a pas mieux pour une taille assez acceptable de donnée, il est préférable une méthode exacte. Chaque problème possède des méthodes mieux adaptées que d'autres.

IV. Méthode de simplexe

L'algorithme du simplexe est un algorithme de résolution des problèmes d'optimisation linéaire. Il a été introduit par George Dantzig à partir de 1947. C'est probablement le premier algorithme permettant de minimiser une fonction sur un ensemble défini par des inégalités. De ce fait, il a beaucoup contribué au démarrage de l'optimisation numérique. L'algorithme du simplexe a longtemps été la méthode la plus utilisée pour résoudre les problèmes d'optimisation linéaire.

IV.1 variables d'écart et variables de surplus

Soit la contrainte suivant

$$2x_1 + 3x_2 + x_3 \leq 5 \qquad (1)$$

En posant : $x_4 = 5 - (2x_1 + 3x_2 + x_3)$

Alors (1) est équivalent à : $x_4 = 2x_1 + 3x_2 + x_3 - 5$ et $x_4 \geq 0$ (2)

x_4 : Dans ce cas s'appelle variable d'écart.

Considérant maintenant la contrainte suivant :

$$2x_1 + 3x_2 + x_3 \geq 5 \qquad (3)$$

Alors en posant : $x_5 = 2x_1 + 3x_2 + x_3 - 5$ et $x_5 \geq 0$ (2)

x_5 : S'appelle variable de surplus.

IV.2. Notion de base

Rappelons la définition d'une base.

IV.2. 1. Définition

Soit A une matrice de rang m. on appelle base de A toute sous matrice carrée régulière d'ordre m extraite de A.

IV.2. 2. Remarque

Comme rg(A)=m, nous pouvons toujours trouver une base.

Soit B une base, alors (en permutant les colonnes de A, si c'est nécessaire) nous pouvons toujours mettre A sous la forme : A=[B,N][1], où N est la sous matrice formée par les colonnes de A qui ne sont pas dans la base et B la sous matrice formée par les colonnes de A qui sont dans la base. De même, posons : $x = [\mathrm{x_B}, \mathrm{x_N}]^{t}$[2].Si $\bar{x}$ est une solution de (P), alors

$$A\bar{x} = b \quad \Leftrightarrow \quad B\bar{x}_B + N\bar{x}_N = b \qquad (5)$$

IV.2. 3. Définition

On appelle solution de base (associée à la base B), la solution particulière de (5) obtenue en posant $\mathrm{x_N} = 0$. si x est une solution de base, (5) devient alors :

$$Bx_B = b \quad \Leftrightarrow x_B = B^{-1}b$$

IV.2. 4. Définition

a) Une solution de base est dite réalisable si $\mathrm{x_B} \geq 0$.

b) Une base correspondante à une solution de base réalisable est appelée base réalisable.

c) Une solution de base est dite dégénérée si le vecteur $\mathrm{x_B} = \mathrm{B}^{-1}b$ a une ou plusieurs composantes nulles.

VI.3. Méthode matricielle du Simplexe

La méthode du Simplexe consiste à donner une procédure algorithmique permettant de trouver une base optimale en commençant par une base réalisable.

Considérons le programme linéaire suivant :

$$(P)\begin{cases} \min \quad z = c^t x \\ sujet\ \text{à} \\ Ax = b \\ x \geq 0 \end{cases}$$

[1] Par exemple si A=$\begin{pmatrix}1 & 0 & 2\\ 2 & 3 & 1\end{pmatrix}$ avec B=$\begin{pmatrix}1 & 0\\ 2 & 3\end{pmatrix}$ et N=$\begin{pmatrix}2\\ 1\end{pmatrix}$ on peut écrire A=[B,N]

[2] Dans l'exemple A=$\begin{pmatrix}1 & 0 & 2\\ 2 & 3 & 1\end{pmatrix}$ avec B=$\begin{pmatrix}1 & 0\\ 2 & 3\end{pmatrix}$ et N=$\begin{pmatrix}2\\ 1\end{pmatrix}$ les colonnes de la base sont 1 et 2 alors pour x=$(x_1, x_2, x_3)\mathrm{x_B} = (x_1, x_2)$ et $\mathrm{x_N} = (\mathrm{x_3})$ et on écrit x=$(\mathrm{x_B}, \mathrm{x_N})$

Où A∈M(n,m) où B est une base de A et supposons qu'on a une solution de base réalisable $\begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$, la valeur de la fonction objectif z_0 est donnée par :

$$z_0 = c^t \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix} = (c_B^t, c_N^t) \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix} = c_B^t B^{-1} b\,.$$

Algorithme primal du Simplexe

Itération k=0

Choisir une base réalisable de départ B_0

k←k+1

Itération k

soient B la base courante et $x=[x_B, x_N]^t$ la solution de correspondante

Etape 1 : Calculer $\bar{b} = B^{-1}b$; $\pi = (c_B^t B^{-1})^t$ et $\bar{c}_N = c_N - \pi^t N$

ci$\bar{c}_N \geq 0$, stop : l'optimum est atteint

S'il existe s tel que $\bar{c}_s < 0$, soit $a_{.s}$ la $s^{ème}$ colonne de A

Etape 2 : calculer $\bar{a}_{.s} = B^{-1}a_{.s}$

Si $\bar{a}_{is} \leq 0\ \forall\ i = 1, \dots, m$, stop : le problèle est non borné inférieurement

Sinon, calculer $\hat{x}_r = \frac{\bar{b}_r}{\bar{a}_{rs}} = min\left\{\frac{\bar{b}_i}{\bar{a}_{is}}, i/\ \ \bar{a}_{is} > 0\right\}$

Soit $e_r = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ → $r^{ème}$ ligne

Etape 3 :

Calculer $a_{.r}$ tel que $B^{-1}a_{.r} = e_r$ et poser $\hat{B} = B + \{a_{.s}\} - \{a_{.r}\}$

Calculer $\hat{B}^{-1}$

Incrémenter k de 1 (k←k+1), remplacer B par $\hat{B}$ et retourner à l'Etape 1.

V. Branch and bound

V.1. Arborescence des solutions possibles

Soit E l'ensemble de solutions au problème. On le suppose discret et fini. On énumère tous les éléments de E en le séparant en n_1 sous-ensembles non vides contenant chacun une partie, de taille variable, des éléments de E (on n'impose pas toujours que ces sous ensembles soient disjoints, mais c'est en général le cas). On peut recommencer avec chaque sous ensemble qui contient plus d'un élément et ainsi de suite jusqu'a ce que tous les ensembles ne contiennent plus qu'un seul élément. Cette énumération peut se représenter par un arbre de la façon suivante :

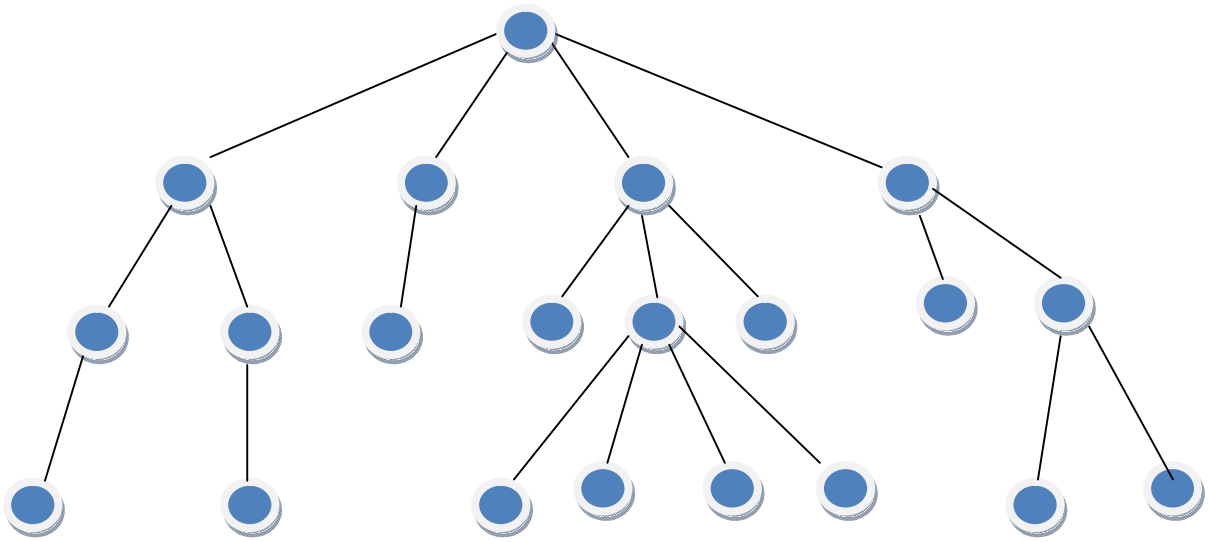


Figure 17 : exemple d'un arbre

V.2. Principe de la méthode

La dénomination séparation et évaluation (Branch&Bound) recouvre deux idées :

1. la séparation (branch) consiste à séparer un ensemble de solutions en sous-ensembles ;

2. l'évaluation (bound) consiste à évaluer les solutions d'un sous-ensemble de façon optimiste, c'est-à-dire en majorant la valeur de la meilleure solution de ce sous-ensemble.

V.3. Stratégies de parcours

En fonction de la structure de données utilisée pour la liste des nœuds actifs NA, l'algorithme peut avoir des performances expérimentales très différentes. La façon dont on parcourt l'arbre des solutions et donc le choix du prochain nœud actif à diviser sont cruciaux. Plusieurs stratégies sont à envisager.

Stratégie de recherche en profondeur : on choisit pour prochain nœud actif l'un des fils du nœud qui vient d'être divisé. Si aucun de ces nœuds n'est actif on revient en arrière (backtrack) dans l'arbre.

Stratégie en largeur : on divise les nœuds dans l'ordre de leur création.

Stratégie de la meilleure évaluation : on divise le nœud de meilleure évaluation.

V.4. Algorithme général :

• A chaque instant, on maintient

– une liste de sous-problèmes actifs,

– le coût U de la meilleure solution obtenue jusqu'alors.

– Valeur initiale de U : soit ∞, soit $c^t x$ pour un x admissible connu.

• Une étape typique est :

1. Sélectionner un sous-problème actif Fi

2. Si Fi est non admissible, le supprimer. Sinon, calculer b(Fi).

3. Si $b(Fi) \geq U$, supprimer Fi.

4. Si $b(Fi) < U$, soit résoudre Fi directement, soit créer de nouveaux sous-problèmes et les ajouter à la liste des sous-problèmes actifs.

VI. Application

Afin de résoudre le problème de sac à dos nous utilisant le Tableau 10 de données

Objets	o_1	o_2	o_3	o_4
v_i	**8**	**6**	**4**	**5**
p_i	**14**	**12**	**7**	**10**

Tableau 10 : données du problème de chargement.

Ce tableau représente 4 objets à placer dans notre sac, chaque objet possède un valeur vi et un poids pi .

Le poids maximal du sac est : 68 unités

Le problème sous forme canonique est :

$$\begin{cases} \max \ 8x_1 + 6x_2 + 4x_3 + 5x_4 \\ sous\ les\ contraintes \\ 14x_1 + 12x_2 + 7x_3 + 10x_4 < 68 \\ x_1, x_2, x_3, x_4\ sont\ des\ entiers \end{cases}$$

Grace à la méthode de simplexe, on trouve la solution optimale suivant du problème relaxé :

$$(x_1, x_2, x_3, x_4) = (4.8571,0,0,0)$$

La valeur optimale est $4.8571 \times 8 + 6 \times 0 + 0 \times 4 + 0 \times 5 = 38.8571$

Or la valeur $x_1 = 4.8571$ n'est pas entier d'où la solution n'est pas réalisable.

On utilise l'algorithme de Brunch and Bound avec la stratégie de meilleur solution on obtient :

• b(Fi) sera le coût optimal de la relaxation linéaire.

• Si la solution de la relaxation est entière, pas besoin de partitionner le sous-problème.

• Sinon, on choisit un x^*_i non entier, et on crée deux sous-problèmes en ajoutant les contraintes : $x_i \leq [x^*_i] \quad et \quad x_i \geq [x^*_i]$

• Ces contraintes sont violées par x*.

Etape 1 :

• U = -∞

• Liste des sous-problèmes actifs : {F}

• Solution de la relax. de F : $x^* = (4.8571,0,4,0)$

• b(F) = 38.8571

• Création des sous-problèmes en rajoutant les Contraintes : $x^*_1 \leq 4$ *ou* $x^*_1 \geq 5$

F_1	F_2
$\boldsymbol{max\ \ 8x_1 + 6x_2 + 4x_3 + 5x_4}$ $\boldsymbol{s.c}$ $\boldsymbol{14x_1 + 12x_2 + 7x_3 + 10x_4 \leq 68}$ $\boldsymbol{x_1 \leq 4}$ $\boldsymbol{x_1, x_2, x_3, x_4\ sont\ des\ entiers}$	$\boldsymbol{max\ \ 8x_1 + 6x_2 + 4x_3 + 5x_4}$ $\boldsymbol{s.c}$ $\boldsymbol{14x_1 + 12x_2 + 7x_3 + 10x_4 \leq 68}$ $\boldsymbol{x_1 \geq 5}$ $\boldsymbol{x_1, x_2, x_3, x_4\ sont\ des\ entiers}$

Tableau 11 : Etape 1 de l'algorithme Brunch and Bound

Liste des sous problèmes actifs $\{ F, F_1, F_2\}$

F2 est non admissible, le supprimer.

Solution de la relaxé de F_1: x* = (4 ,0,1.7143,0)

• b(F_1) = 38.8571

Etape 2 :

• U = -∞

• Liste des sous-problèmes actifs : $\{ F, F_1\}$

• Solution de la relaxé de F_1: x* = (4 ,0,1.7143,0)

• b(F_1) = 38.8571

• Création des sous-problèmes en rajoutant les Contraintes : $x^*_3 \leq 1$ *ou* $x^*_3 \geq 2$

F_{11}	F_{12}
$\boldsymbol{max\ \ 8x_1 + 6x_2 + 4x_3 + 5x_4}$ $\boldsymbol{s.c}$ $\boldsymbol{14x_1 + 12x_2 + 7x_3 + 10x_4 \leq 68}$ $\boldsymbol{x_1 \leq 4}$ $\boldsymbol{x_3 \leq 1}$ $\boldsymbol{x_1, x_2, x_3, x_4\ sont\ des\ entiers}$	$\boldsymbol{max\ \ 8x_1 + 6x_2 + 4x_3 + 5x_4}$ $\boldsymbol{s.c}$ $\boldsymbol{14x_1 + 12x_2 + 7x_3 + 10x_4 \leq 68}$ $\boldsymbol{x_1 \leq 4}$ $\boldsymbol{x_3 \geq 2}$ $\boldsymbol{x_1, x_2, x_3, x_4\ sont\ des\ entiers}$

Tableau 12 : Etape 2 de l'algorithme Brunch and Bound

Solution de la relaxé de F_{11}: x* = (4 ,0.4167,1,0)

• b(F_{11}) = 38.5

Solution de la relaxé de F_{12}: x* = (3.857143 ,0,2,0)

• b(F_{12}) = 38.85714

La meilleure solution est F_{12}

Etape 3 :

• U = -∞

• Liste des sous-problèmes actifs : { F,F_{11}, F_{12}}

• b(F_1) = 38.8571

• b(F_{12}) = 38.85714

La meilleure solution est F_{12}

Solution de la relaxé de F_{12}: x* = (3.857143 ,0,2,0)

• Création des sous-problèmes en rajoutant les Contraintes : $x^*_1 = 4\ ou\ x^*_1 \leq 3$

F_{121}	F_{122}
$\boldsymbol{max\ \ 8x_1 + 6x_2 + 4x_3 + 5x_4}$ $\boldsymbol{s.c}$ $\boldsymbol{14x_1 + 12x_2 + 7x_3 + 10x_4 \leq 68}$ $\boldsymbol{x_1 = 4}$ $\boldsymbol{x_3 \geq 2}$ $\boldsymbol{x_1, x_2, x_3, x_4\ sont\ des\ entiers}$	$max\ \ 8x_1 + 6x_2 + 4x_3 + 5x_4$ $\boldsymbol{s.c}$ $\boldsymbol{14x_1 + 12x_2 + 7x_3 + 10x_4 \leq 68}$ $\boldsymbol{x_1 \leq 3}$ $\boldsymbol{x_3 \geq 2}$ $\boldsymbol{x_1, x_2, x_3, x_4\ sont\ des\ entiers}$

Tableau 13 : Etape 3 de l'algorithme Brunch and Bound

F_{121}est non admissible, le supprimer.

Solution de la relaxé de F_{122}: x* = (3 ,0,1,3.714286,0) • b(F_{122}) = 38.8574

Etape 4 :

• U = -∞

• Liste des sous-problèmes actifs : { F,F_{11}, F_{122}}

• b(F_{11}) = 38.5

• b(F_{122}) = 38.8574

La meilleure solution est F_{122}

Solution de la relaxé de F_{122}: x* = (3,0,3.14286,0)

• Création des sous-problèmes en rajoutant les Contraintes : $x^*_3 \leq 3\ ou\ x^*_3 \geq 4$

F_{1221}	F_{1222}
$\boldsymbol{max\ \ 8x_1 + 6x_2 + 4x_3 + 5x_4}$ $\boldsymbol{s.c}$ $\boldsymbol{14x_1 + 12x_2 + 7x_3 + 10x_4 \leq 68}$ $\boldsymbol{x_1 \leq 3}$ $\boldsymbol{x_3 = 2}$ $\boldsymbol{x_1, x_2, x_3, x_4\ sont\ des\ entiers}$	$max\ \ 8x_1 + 6x_2 + 4x_3 + 5x_4$ $\boldsymbol{s.c}$ $\boldsymbol{14x_1 + 12x_2 + 7x_3 + 10x_4 \leq 68}$ $\boldsymbol{x_1 \leq 3}$ $\boldsymbol{x_3 \geq 3}$ $\boldsymbol{x_1, x_2, x_3, x_4\ sont\ des\ entiers}$

Tableau 14 : Etape 4 de l'algorithme Brunch and Bound

Solution de la relaxé de F_{1221}: x* = (3,1,2,0)

• b(F_{1221}) = 38

• U = 38

Solution de la relaxé de F_{1222}: x* = (3 ,0,3.714286,0)

• b(F_{1222}) = 38.85714

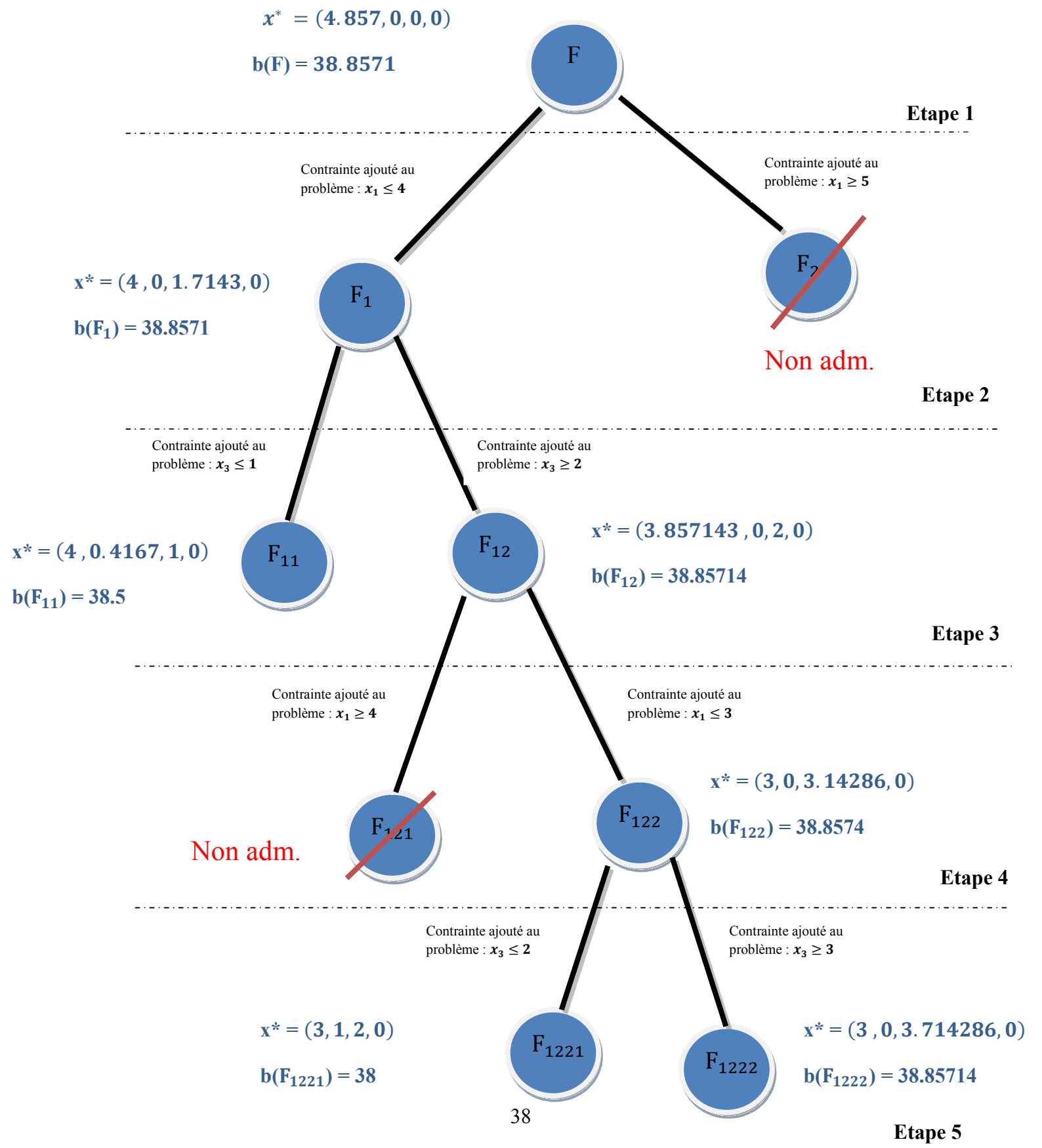
• Liste des sous-problèmes actifs : {F, F_{1221}}

• Solution de F_{1221} = (3,1,2,0)

• Solution de F = (3,1,2,0)

Résolution sous forme de graphes :

Figure 18 : résultats de l'application da l'algorithme brunch and bound



```
>> a=[14 12 7 10 1]

a =

    14    12     7    10     1

>> C=[8 6 4 5 0]'

C =

     8
     6
     4
     5
     0

>> b=68

b =

    68
```

```
>> [x y]=brunch_and_bound(C,a,b,[1 2 3 4 5],0.001)

x =

    4.0000
    1.0000
         0
         0
         0
    0.0000
    1.0000
    0.0000
         0
         0


y =

    38
```

Figure 19 Résultats numérique du problème de chargement.

$$Z_{optimale} = 38$$

Conclusion :

La recherche opérationnelle se situe au carrefour de différentes sciences et technologies et propose un ensemble de méthodes scientifiques pour résoudre des problèmes d'optimisation liés aux organisations du monde réel : problèmes de logistique, d'emploi du temps, de gestion des flux, de transport...

Dans ce modeste travaille nous avons essayé de résoudre deux classique problèmes de la recherche opérationnelle spécialement du domaine de logistique. Ainsi nous avons Implémenté la méthode de Stripping-Stone et la méthode de coin Nord-ouest pour le problème de transport et la méthode séparation et évaluation (branch and bound) avec la méthode de Simplex pour la résolution du problème de chargement. Mais grâce aux développements scientifiques, différente variation des deux problèmes ont était proposé, ainsi que différent méthode de résolutions exacte est approchée. Dans un prochain avenir nous espérons pouvoir continuer à travailler sur les problèmes de la recherche opérationnelle ainsi que leur résolution, car c'est un domaine vaste, riche et très intéressant.

Références :

[1] Pr Michel Bierlaire, cours de l'Optimisation en nombres entiers, ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE, 2015/2016.
[2] Pr Youssef Benadada & A. El Hilali Alaoui : «Programmation Mathématique, de la modélisation à la résolution», Edition Kawtar Print, Rabat, 2012.
[3] Mr. ETTAOUIL, Professeur à la Faculté des Sciences et Technique de Fès, Cours de la recherche opérationnelle année 2015/2016.
[4] Bernard Fortz, Cours : Recherche opérationnelle et applications, Université Libre de Bruxelles, 2012/2013.
[5] Sidi Mohamed Douiri, Souad Elbernoussi, Halima Lakhbab, Cours des méthodes de Résolution Exactes Heuristiques et Métaheuristiques, Université Mohammed V, Faculté des sciences de Rabat.

Sites web :

http://www.getlinkyoutube.com/channel/UCYPU27PO3Eg086WxlJjbQbw

http://cosy.univ-reims.fr/~pdelisle/fichiers/info0804_14-15/

https://fr.wikipedia.org

https://www.youtube.com