الجمهوريـة الجزائـريـة الديمقراطيـة الشعبيـة
République Algérienne Démocratique et Populaire
وزارة التعليـم العالـي والبحـث العلمـي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

N° Réf :……

Centre Universitaire de Mila

Institut des Sciences et de la Technologie **Département de mathématiques et informatique**

Mémoire préparé En vue de l'obtention du diplôme de licence

en : filière mathématique fondamental

Thème

Problème du voyageur de commerce et ses applications

Encadré par :

Mr : BOUFELGHA Ibrahim

Préparé par : **BOUGHEROUAT Sara**
DJELAMA Souad
KEDJTIT Besma
ZERARA Sabah

Année universitaire : 2013/2014

Remerciement

*Au terme de ce travail, nous commençons par remercier **DIEU** pour d'avoir nous donné la volonté et le courage pour terminer ce modeste travail.*

Nos tenons à exprimer notre gratitude et notre sincère remerciement à tous ceux qui ont contribué, de prés ou de loin à l'élaboration de ce mémoire.

*Nous remerciements très chaleureux sont adressé à Monsieur **BOUFELGHA Ibrahim** notre encadreur.*

Nos adressons également mes remerciements chaleureux aux membres de l'institut des sciences et de la technologie.

Nous remerciements sont également adressés à tous les enseignants qui ont contribué à notre formation.

Introduction

[1] Un graphe est une collection d'éléments mis en relation entre eux. Géométriquement, on représente ces éléments par des points (les sommets) reliés entre eux par des arcs de courbe (les arêtes). Selon que l'on choisit d'orienter les arêtes ou de leurs attribuer un poids (un cout de passage), on parle de graphes orientés ou de graphes pondérés.

La théorie des graphes s'intéresse a leurs multiple propriétés : existence de chemins les plus courts ou les moins couteux, de cycles particuliers (eulériens, hamiltoniens, …), nombre d'intersections dans le plan, problèmes de coloriage, etc.
On fait remonter la théorie des graphes au problème des ponts de Königsberg : peut-on organiser dans cette ville une promenade dont le trajet serait une boucle qui passe une fois et une seule par chacun des sept ponts ? En termes modernes, il s'agit de montrer l'existence ou non d'un cycle eulérien dans le graphe associé : Euler a montré en 1736 qu'un tel parcours n'existe pas.

Aujourd'hui, les graphes trouvent maintes applications dans la modélisation des réseaux (routiers, informatiques, etc.). Par ailleurs la théorie des graphes a fourni des problème en algorithmique (problème du voyageur de commerce, coloration de graphes, calcul du plus grand sous-graphe commun a deux graphes, etc.), qui sont cruciaux en théorie de la complexité (problème NP-complets, conjecture P=NP).

Un voyageur de commerce doit visiter n villes données en passant par chaque ville exactement une fois. Il commence par une ville quelconque et termine en retournant à la ville de départ. Les distances entre les villes sont connues. Quel chemin faut-il choisir afin de minimiser la distance parcourue ? La notion de distance peut-être remplacée par d'autres notions comme le temps qu'il met ou l'argent qu'il dépense , dans tous les cas, on parle de coût.

En 19 iéme siècle, les premières approches mathématiques exposées pour le problème du voyageur de commerce ont été traitées au 19 iéme siècle par les mathématiciens Sir William Rowan Hamilton et Thomas Penyngton Kirkman. Hamilton en a fait un jeu : Hamilton's Icosian game. Les joueurs devaient réaliser une tournée passant par 20 points en utilisant uniquement les connections prédéfinies. En 1930 Le PVC est traité plus en profondeur par Karl Menger à Harvard. Il est ensuite développé a Princeton par les mathématiciens Hassles Whitney et Merrill Flood. Une attention particulière est portée sur les connections s par Menger et Whitney ainsi que sur la croissance du PVC. En 1954 une solution du PVC pour 49 villes est donnée par Dantzig, Fulkerson et Johnson par la méthode du cutting-plane. En 1975 une solution pour 100 villes est donnée par Camerini, et en 1987 solution pour 532, puis 2392 villes par pad berg et Rinaldi.1998 solution pour les 13509 villes des Etats-Unis. 2001 solutions pour 15112 villes d'Allemagne par Apple gâte, Bixby, Chvàtal et Cook des universités de Rice et Princeton.

Notre mémoire comporte trois chapitres. Le premier chapitre est consacré pour des définitions et des notions qui nous seront utiles pour la compréhension de ce mémoire. Dans le deuxième chapitre nous définissons le problème du commerce, en suite nous donnons une modélisation sous forme d'un programme linéaire pour le PVC et nous citons quelques méthodes pour la résolution de ce problème avec des exemples explicatifs. Dans le troisième chapitre nous citons une situation pratique qui peut se formuler en problème de voyageur de commerce.

Chapitre1

Terminologie et notation

Dans ce chapitre nous citons les définitions de différentes notions que nous utilisons pour la description du problème du voyageur de commerce.

1.1 Terminologie et définitions générales [2]

1.1.1 Graphe non orienté

Un graphe non orienté $G = (V,E)$ est la donnée de deux ensemble, un ensemble fini et non vide de sommets V et un ensemble fini d'arêtes E. Une arête e ∈ E est une paire de sommets (u, v), notée e = uv où u et v sont les extrémités de e, et on dira dans ce cas que u et v sont adjacents et que (u, v) est incidente à u et à v. Un sommet est dit $isolé$ s'il n'est adjacent à aucun sommet de G.

Le nombre de sommets de G est appelé $ordre$ de G et le nombre d'arêtes est appelé la taille de G .Un graphe est dit $non\ trivial$ s'il est d'ordre supérieur à1.

1.1.2 Multigraphe

Un mulitgraphe est un graphe dans lequel il peut exister plusieurs arêtes entre une paire de sommets.

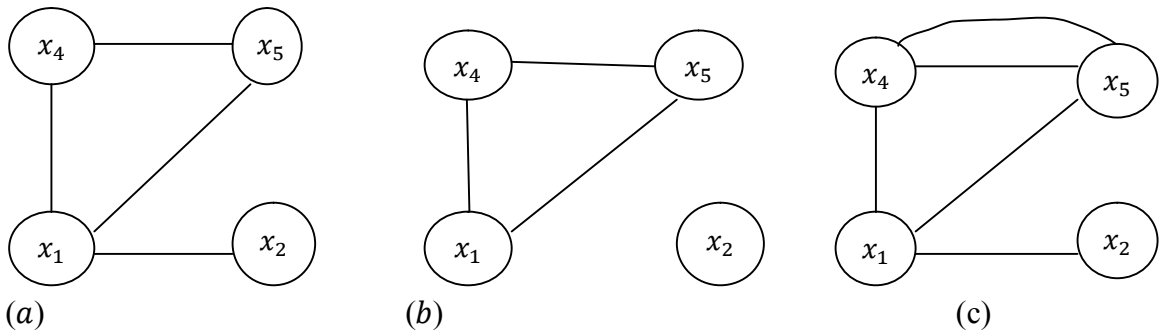


FIG.1.1-(a) Un graphe simple –(b) un graphe simple avec sommet isolé-(c) un multi graphe

1.1.3 Graphe simple

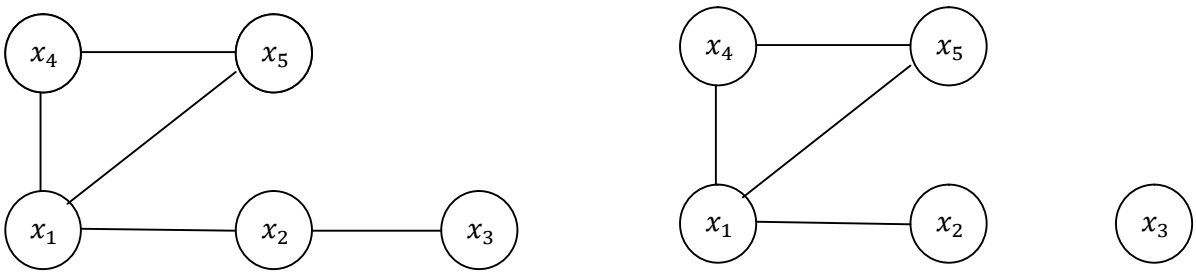
Un graphe est dit simple s'il est sans boucles et sans arêtes multiples.

1.1.4 Graphe connexe

Un graphe est dit $connexe$ si pour toute paire de sommets du graphe, il existe une chaîne les reliant. Une $composante\ connexe$ d'un graphe G est un sous-graphe connexe maximal de G. Un graphe est non connexe s'il possède au moins deux composantes connexes.

Exemple 1.1

Le graphe illustré Dans FIG. 1.2 − (a) est un graphe connexe, par contre celui FIG .1.2-(b) ne l'est pas.



(a) (b)

FIG.1.2-(a) un graphe connexe-(b) un graphe non connexe

1.1.5 Voisinage

Soit G = (V, E) un graphe et $v \in V$.Le *voisinage* ouvert de v l'ensemble

N (v)=$\{u \in V \mid uv \in E\}$ et l Le voisinage $f$ermé de v est l'ensemble

N $[v] =$ N $[v] \cup \{v\}$.

Soit sCV.Le voisinage ouvert de S est l'ensemble $N(s) = \cup_{v \in s} N(v)$ et le voisinage fermé de s est l'ensemble N[S]=N(S)∪ S.

1.1.6 Degré d'un sommet

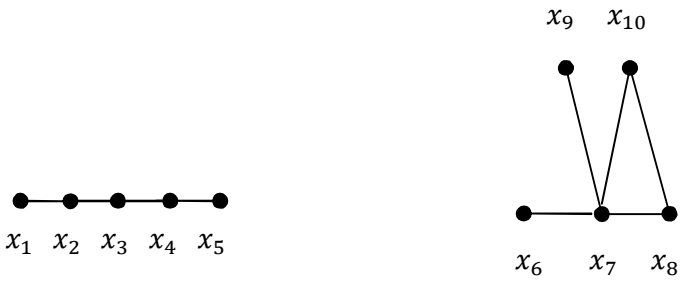
Le degré d`un sommet $v$ , noté d $(v)$ , est le nombre de sommet adjacents a`v .On note par $\Delta(G) = \max_{v \in V(G)} d(v)$ le degré maximum du graphe G et par δ(G) le degré minimum de G. En particulier, les sommets isolés sont de degré égal à 0 et les sommets pendants sont les sommets de degré égal à 1.

1.1.7 Chaîne

Une chaîne p_n dans un graphe G= (v, E) est une séquence finie de sommets $x_1, x_2,$ ...,x_n .Telle que pour tout entier$i$, $11 \leq i \leq n-1$,$e_i = x_i x_{i+1} \in E$.
L'entier n-1 représente la *longueur* de p_n et les sommets x_1 et x_n sont appelés *extrémité initiale* et *extrémité finale* respectivement de la chaîne p_n.
Une chaîne est dite *élémentaire* si tous ses sommets sont distincts. Une chaîne est dite simple si toutes ses arêtes sont distinctes.

Exemple 1.2 le graphe illustré dans FIG.1.3 -
(a) représente la chaîne simple élémentaire

Mais dans FIG.1.3 $-(a)$, *la chaîne est simple mais non élémentaire.*



F*IG*. 1.3 $-(a)$ Lchaîne x_1, x_2, x_3, x_4, x_5 *est simple et* élémentair $-(b)$ La Chaîne $x_6, x_7, x_8, x_{10}, x_7, x_9$ *est simple mais non élémentair*

1 .1.8 Corde

Une corde est une arête qui relie deux sommets non consécutifs dans une chaîne.
Une chaîne minimale induite par n sommets, notée par p_n, est une chaîne élémentaire sans corde.

1.1.9 Cycle

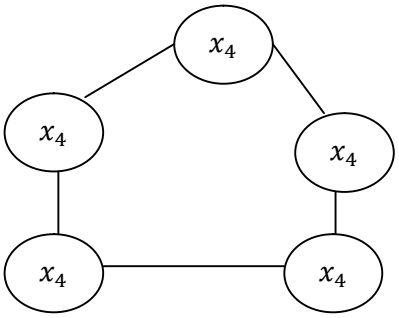
Un cycle élémentaire C_n induit par n sommets est un cycle dont les sommets sont distincts.

Exemple 1.3

Le graphe illustré dans la $figure$ FIG.1.4 représente le cycle d'ordre $5 C_5 = x_1, x_2, x_3, x_4, x_5, x_1$.

1.1.10 Distance entre deux sommets

La distance d (u, v) entre deux sommets u *et* v d'un graphe G est le nombre d'arêtes dans une plus courte chaîne reliant u à v.



F IG. 1.4-Le cycle $c_5 = \{x_1, x_2, x_3, x_4, x_5, x_1\}$

1.1.11 Excentricité d'un sommet

$L'excentricité$ D'un sommet u dans un graphe G est la plus grande distance entre le sommet u et n'importe quel autre sommet v de G, C'est-à-dire diam $e(u) = max_{v\in V}\{d(u,v)\}$.

1.1.12 Diamètre d'un graphe

Le diamètre d'un graphe G est la plus grande excentricité dans le graphe G , c'est à dire $diam(G) = max_{u\in V} e(u)$.

1.1.13 Rayon d'un graphe

Le rayon du graphe G est l'excentricité minimum sur tous les sommets de G, c'est à dire rad(G) = $\min_{u\in v} e(u)$

1.1.14 Sous-graphe

Un graphe H = (v_H,E_H) est un sous-graphe de G= (v(g), E(G)) SI $V_H \subseteq v(G)$et $E_H \subseteq E(G)$:

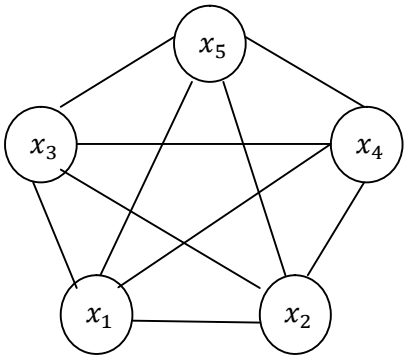
Pour un ensemble sommets de sommets $S \subset V(G)$, le sous graphe de G induit par S est le graphe noté S ayant S pour ensemble de sommets. Les arêtes de sont celles de E (G) dont les deux extrémités sont dans S.

1.1.15 Quelques familles de graphe

1.1.15.1 Graphe complet

Un graphe simple est dit complet si toute paire de sommets de G est reliée par arête

Exemple 1.4 *le graphe* K_5 illustré dans FIG.1.5 est un graphe complet à 5 sommets.



F_{IG}.1.5-Le graphe complet K_5

1.1.15.2 Arbre

Un arbre est un graphe connexe sans cycle. D'autres définitions équivalentes sont possibles pour qu'un graphe G d'ordre n soit un arbre:

- G est connexe et possède *n-1* arêtes.
- G est sans cycle et possède *n- 1* arêtes.
- G est connexe et minimal pour cette propriété.
- G est sans cycle et maximal pour cette propriété.
- Entre toute paire de sommets, il existe une unique chaîne les reliant.

On distingue trois types de sommets dans un arbre :

- La racine.
- Les feuilles : ce sont les sommets de degré égal à1.

1.2. Complexité algorithmique des problèmes d'optimisation combinatoire [2]

On peut classer certains problèmes d'optimisation combinatoire dans différentes classes selon leurs difficultés intrinsèques.

1.2.1 Classe P

Un problème est dit appartenir a la Classe P s il

Peut être résolu par un algorithme polynomiale. On dit que les problèmes de la classe P sont faciles. Notons que la non-connaissance d un algorithme polynomial pour résoudre un problème donné ne signifie pas qu'il ne soit pas dans la classe P

1.2.2 Classe NP

Un problème appartient à la classe NP si, lorsqu' une solution à ce problème est proposée, Alors on peut vérifier en un temps polynomiale que la réponse correspondante est VRAI.

1.2.3 Réduction polynomiale (au sens de Turing)

Soient P_1 et P_2 deux problème de décision. P_1 Se réduit polynomiale ment à P_2 s il existe un algorithme pour P_1 qui fait appel (comme sous programme) à un algorithme de résolution de P_2 , et si cet algorithme de résolution de P_1 est polynomial lorsque la résolution de P_2 est comptabilisée comme opération élémentaire.

1.2.4 Classe du problème NP-complets

Un problème de décision est dit NP-complet si tout problème de la classe NP se réduit polynomiale ment à lui. Un problème d'optimisation combinatoire dont le problème de décision associé est NP-complet est dit NP-dur.

Chapitre 2

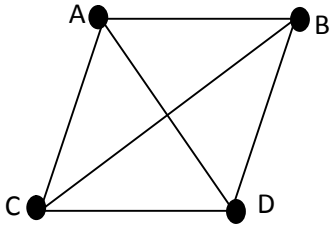
Problème de voyageur de commerce

Dans les industries de transport, l'optimisation des itinéraires et des tournées de véhicules est très intéressant pour minimiser les couts. C'est dans ce domaine que se rencontrent de multiples variantes du problème dit du «voyageur de commerce ». Dans ce chapitre, nous donnons une modélisation pour le PVC sous forme d'un programme linéaire ainsi les méthodes proposées pour la résolution de ce problème.

2.1 Définition [4]

Le problème de voyageur de commerce, ou TSP (pour traveling salesman problem), est le suivant : un représentant de commerce ayant n villes à visiter souhaite établir une tournée qui lui permette de passer exactement une fois par chaque ville et de revenir à son point de départ avec un moindre coût, c'est-à-dire en parcourant la plus petite distance possible. C'est un des problèmes les plus anciennement et largement étudiés en optimisation combinatoire.
Ses applications sont nombreuses. Par exemple directement sous forme d'un PVC et certains problème, comme les problèmes de transport, sont plus complexes que le PVC mais présentent une structure sous-jacente de type PVC.

Le problème de voyageur de commerce est un problème mathématique qui consiste à donner un ensemble de villes séparées par des distances données et à trouver le plus court chemin qui relie toutes les villes. Il s'agit d'un problème d'optimisation pour lequel on ne connait pas d'algorithme permettant de trouver une solution exacte en un temps polynomial. De plus, la version décisionnelle de l'énoncé (pour une distance D, existe-t-il un chemin plus court que D passant par toutes les villes?) est connue comme étant un problème NP-complet.



Si un voyageur part du point A et que les distances entres toutes les villes sont connues, quel est le plus court chemin pour visiter touts les points et revenir au point *A* ?

2.2 Circuits Hamiltoniens

2.2.1 Définition, typologie, applications

Soit $G = (V, E)$ un graphe, et soit d une fonction positive sur les arêtes, on appelle circuit Hamiltonien un chemin fermé passant par tous les sommets $v \in V$. De manière duale, on appelle circuit Eulérien un chemin fermé passant par touts les arêtes e ϵ E.
Le problème du voyageur de commerce revient ainsi à trouver un circuit Hamiltonien de longueur minimale dans le graphe.
Bien que les définitions des circuits Eulériens et Hamiltoniens soient très proches, ces

deux problèmes sont de difficultés très différentes. Le simple problème de l'existence d'un circuit est NP-complet pour les circuits Hamiltoniens et polynomial pour les circuits Eulériens. En effet, il existe un circuit Eulérien si et seulement si le graphe est connexe et tous les sommets sont de degré pair. Dans ce cas, la recherche que d'un tel circuit peut se faire avec un algorithme de complexité *O(m)* (on rappelle que *m= lEl)* désigne le nombre d'arêtes de *G*.
Pour le problème de minimisation de parcours, la fonction d peut être quelconque. Des algorithmes spécifiques peuvent être proposés quand il s'agit d'une distance L_1, L_2 ou L_∞, ou simplement quand la fonction d vérifie l'inégalité triangulaire. Enfin, la matrice de distance peut aussi être asymétrique (par exemple, si l 'on considère le cas d'une ville avec des sens uniques). Le cas particulier du TSP symétrique.
L'algorithme que nous proposerons s'appliquera aussi bien au TSP asymétrique que symétrique.

Enfin, ce problème du voyageur peut être envisagé avec des contraintes supplémentaires. Les plus courantes viennent du domaine du transport et sont liées soit à une formulation temporelle, soit à des problèmes de capacité.

- Dans la métaphore habituelle du voyageur de commerce, la fonction d représente

une distance géographique. Elle peut tout aussi bien représente un temps de trajet. Si l'on particularise une ville comme ville de départ (appelée le déport), d'où le tour commerce au temps 0, on peut associer une heure de visite à chaque ville. Dans cette description temporelle du problème, on peut considère deux types de contraintes supplémentaires. On peut contraindre les dates de visite en associant à chaque ville une fenêtre de temps pendant laquelle le tour doit y passer. Le problème est alors dénoté par les initiales TSPW (Traveling Salesman problem with Time Windows). D'autres contraintes peuvent aussi faire dépendre le temps de trajet entre deux endroits (c'est-à-dire la distance d) de l'heure à laquelle le trajet est effectué. Cette formulation rend bien compte des situations d'embouteillages autour des grandes villes le matin et le soir.

- Si l'on considère que le voyageur de commerce est en fait un transporteur qui peut à

chaque endroit prendre et déposer de la marchandise (les pickups and delivery problems), on est amené à rajouter deux types de contraintes. D'une part des contraintes de précédences (aller chercher la marchandise avant de la livrer), et d'autre part des contraintes de capacité vérifiant que l'on est bien physiquement capable de transporter toutes les marchandises. Ceci peut éventuellement interdire des trajets qui commenceraient par procéder à tous les enlèvements avant de faire la moindre livraison.
On traitera dans ce chapitre de la résolution du TSP pur et de la prise en compte d'éventuelles fenêtres de temps. Le TSPTW est en effet particulièrement intéressant

puisqu'il est à mi-chemin entre un TSP et un problème d'ordonnancement disjonctif. Si toutes les fenêtres de temps sont égales à *[0, M],* on obtient le TSP pur, et si les distances d_{ij} ne dépendent que de i (ce qui peut correspondre à un temps de visite au noeud i et un temps de trajet nul entre *i* et *j*), on retrouve un job-shop à une machine. Il y a donc un continuum entre le TSP auquel on ajoute des fenêtres de temps et l'ordonnancement disjonctif auquel on ajoute des temps d'adaptation de la ressource entre le traitement de deux tâches successives. Mentionnons enfin qu'il existe de nombreuses autres variantes du TSPTW, suivant ce que l'on cherche à optimiser : au lieu de chercher à minimiser la date de fin, on peut vouloir minimiser une distance géographique (on n'additionne que les temps de parcours et on ignore les temps d'attente), ou bien minimiser la somme des temps d'attente. On peut aussi autoriser le tour à ne pas commencer en *0* au dépôt, mais un peu plus tard afin de pouvoir réduire certains temps d'attente intermédiaires.

2.3 Programmation linéaire [1]

[3] On cherche à optimiser une fonction f linéaire sous des contraintes linéaires.
La forme générale d'un programme linéaire est la suivante *:*

$$min \sum_{j=1}^{n} c_j x_j$$

Sous les contraintes:

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \ , i \in M$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \ , i \in \bar{M}$$

Avec *M* et $\bar{M}$ un ensemble d'indices, $x_j \in \mathbb{R}_+ pour\ j \in M\ et\ j \in \bar{M}$.

2.3.1 Problèmes de production

On a un centre de production disposant de ressources numérotées de1 à m, et produisant n
Types de produits. On note :
– a_{ij} la quantité de la ressource j requise pour produire une unité du produit i ;
– $\mathrm{x_i^{min}}(resp.\ x_i^{max}$) la quantité minimale (resp. maximal) de produit i à produire ;
– b_j la quantité de ressource j disponible ;
– m_i le bénéfice réalisé en produisant une unité de produit i ;
– x_i la quantité produite dei.
Le problème se formalise alors de la façon suivante :

$$\begin{cases} max \sum_{i=1}^{n} m_i x_i \\ x_i \leq x_i^{max} \quad i\epsilon[1,n] \\ x_i \leq x_i^{min} \quad i\epsilon[1,n] \\ \sum_{i=1}^{n} a_{ij} x_i \leq b_j \qquad j\epsilon[1,m] \\ x_i \geq 0 \qquad i\epsilon[1,n] \end{cases}$$

Ce n'est pas un programme linéaire. Mais on peut réécrire ce problème pour le ramener à un tel problème.
On applique alors l'algorithme du simplexe.

2.3.1.1 Optimisation fractionnaire

On considère le problème suivant :

$$\begin{cases} max\, r(x) = \dfrac{b_0 + \sum_{j=1}^{n} b_j x_j}{c_0 + \sum_{j=1}^{n} c_j x_j} \\ \sum_{j=1}^{n} a_{ij} x_j \leq d_i \qquad i\epsilon[1,m] \\ x_i \geq 0 \qquad j\epsilon[1,n] \\ b_0, b_j, a_j \geq 0 \\ c_0 > 0 \end{cases}$$

En posant

$$y_j = \frac{x_j}{c_0 + \sum_{j=1}^{n} c_j x_j} \qquad\qquad t = \frac{1}{c_0 + \sum_{j=1}^{n} c_j x_j}$$

Le problème se réécrit de la manière suivante :

$$\begin{cases} max b_0 t + \sum_{j=1}^{n} b_j y_j \\ \sum_{j=1}^{n} a_{ij} y_j \leq d_i t \quad i\epsilon[1,m] \\ c_0 t \sum_{j=1}^{n} c_j + y_j = 1 \\ y_j \geq 0 \quad j\epsilon[1,n] \\ t \geq 0 \end{cases}$$

2.3.1.2 Interprétation géométrique d'un programme linéaire

On considère le problème suivant :

$$\max x_1 + 3x_2$$

s.c.

$$\begin{aligned} x_1 + x_2 &\leq 5 \\ x_1 &\leq 4 \\ x_2 &\leq 3 \\ x_2, x_2 &\geq 0 \end{aligned}$$

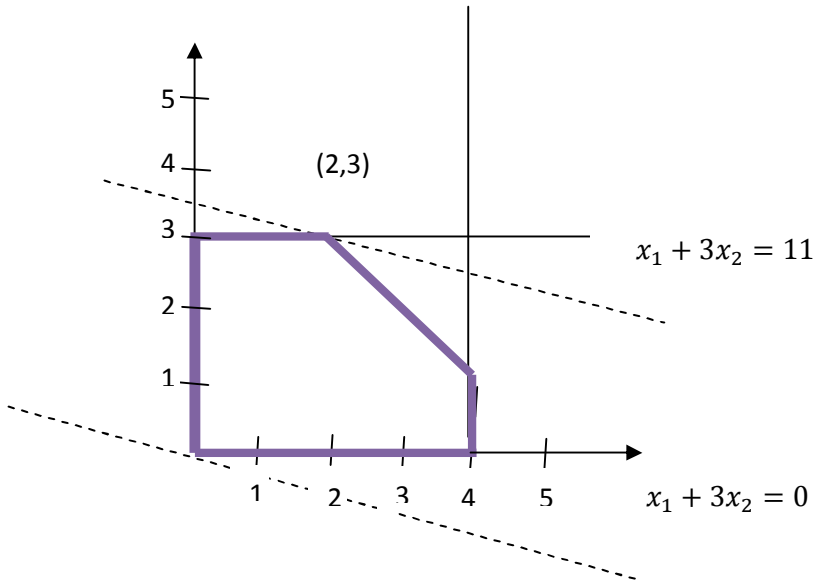


FIGURE 2.1 – Interprétation géométrique d'un programme linéaire
Le point optimal est (2, 3) ; la valeur obtenue est 11.

2.3.1.3 Forme générale d'un programme linéaire

Le problème peut se présenter sous deux formes différentes :

Forme standard	**Forme canonique**
$min\ cx$	$min\ cx$
$Ax = b$	$Ax \geq b$
$x \geq 0$	$x \geq 0$

Afin de passer de la forme générale à la forme canonique, il faut remplacer la condition

$$\sum_{j=1}^{n} a_{ij}x_j = b_j$$

Par les deux conditions

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

Afin de passer de la forme générale à la forme standard, il faut ajouter une variable d'écart si ¸ *0*. Ainsi, on transforme la condition.

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i$$

En

$$\sum_{j=1}^{n} a_{ij}x_j + s_t = b_t, s_t \geq 0$$

2.3.1.4 Base et solution de base

On suppose que $rg(A) = m$
On appelle matrice de base la sous-matrice carrée $(m \times m)$de *A* régulière.
Ainsi :

$$A = (B \quad N)(par\ blocs)$$

Et

$$x = \begin{pmatrix} x^{\mathfrak{B}} \\ x^{\mathcal{N}} \end{pmatrix}(par\ blocs)$$

Les colonnes de *B* représentent les variables de base, et celles de *N* représentent les variables hors base.
D'où :

$$Ax = b$$
$$Bx^{\mathfrak{B}} + Nx^{\mathcal{N}} = b$$
$$B^{-1}Bx^{\mathfrak{B}} + B^{-1}Nx^{\mathcal{N}} = B^{-1}b$$

Et donc

$$x^{\mathfrak{B}} + B^{-1}Nx^{\mathcal{N}} = B^{-1}b$$

La solution de base relativement à *B* est la solution telle que $x^{\mathcal{N}} = 0\ et\ x^{\mathfrak{B}} = B^{-1}b$.

Exemple

Considérons le problème suivant :

$$min 3x_1 + x_2 + 4x_3 + 2x_4 - 2x_5$$

s.c.

$$\begin{aligned} x_1 + 2x_4 + 3x_5 &= 2 \\ 2x_2 + 4x_4 &= 6 \\ 2x_1 + x_2 + 2x_3 + 2x_5 &= 9 \\ x_1, x_2, \dots, x_5 &\geq 0 \end{aligned}$$

Ici la matrice A vaut :

$$A = \begin{pmatrix} 1 & 0 & 0 & 2 & 3 \\ 0 & 2 & 0 & 4 & 0 \\ 2 & 1 & 2 & 0 & 2 \end{pmatrix}$$

En utilisant les colonnes *1, 2,* et *3* de A (car A est de rang *3* et ces colonnes sont linéairement indépendantes), on a :

$$B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 2 & 1 & 2 \end{pmatrix} \qquad B^{-1} = \frac{1}{4}\begin{pmatrix} 4 & 0 & 0 \\ 0 & 2 & 0 \\ -4 & -1 & 2 \end{pmatrix} \qquad b = \begin{pmatrix} 2 \\ 3 \\ 9 \end{pmatrix}$$

La solution de base réalisable est

$x_1 = 2$ $x_1 = 3$ $x_3 = 1$ $x_4 = 0$ $x_5 = 0$

De valeur 13.

L'ensemble $X = \{x : Ax \leq b, x \geq 0\}$est un polytope convexe.

Définition [1]

On appelle point extrême tout point $x \epsilon X$qui ne peut être exprimé comme combinaison convexe d'autres points $y \epsilon X (y \neq x)$.

Théorème [1]

L'ensemble des points extrêmes de Xcorrespond à l'ensemble des solutions de base réalisables.

Corollaire [1]

X Admet un nombre fini de points extrêmes. Le nombre de solutions de base est inférieur ou égal à C_n^m.

Corollaire [1]

Tout point d'un polytope convexe X peut s'exprimer comme une combinaison convexe des points extrêmes de X.

Théorème (Théorème central de la programmation linéaire) [1]

L'optimum de z fonction linéaire sur le polytope convexe X est atteint en au moins un point extrême.

S'il est atteint en plusieurs points extrêmes, alors il est atteint en tout point combinaison convexe de ces points extrêmes.

Démonstration

Soient $y^1, y^2 \ldots, y^k$ points extrêmes de X.

Soit$z^* = min_{k\epsilon[1,n]} z(y^k)$. Montrons que z^* est le minimum de *z* sur *X*.

Soit $x\epsilon X$. Alors

$$x = \sum_{k=1}^{k} \lambda_k y_k \qquad \lambda_k \geq 0, \sum \lambda_k = 1$$

Par linéarité dez, on obtient

$$\begin{aligned} z(x) &= \sum_{k=1}^{k} \lambda_k z(y^k) \\ &\geq \textstyle\sum_{k=1}^{k} \lambda_k z^* \\ &\geq z^* \end{aligned}$$

Donc z*est la valeur de la solution optimale.

2.3.2 Algorithme du simplexe [1]

L'algorithme fait une exploration "dirigée" des points extrêmes.

2.3.2.1 Caractérisation des solutions de base réalisables optimales

On rappelle le problème décomposé en variables de base et hors base :

$$x^{\mathfrak{B}} + B^{-1}Nx^{\mathcal{N}} = B^{-1}b$$

Définition [1]

Une solution de base est dégénérée si certaines variables de base sont nulles.

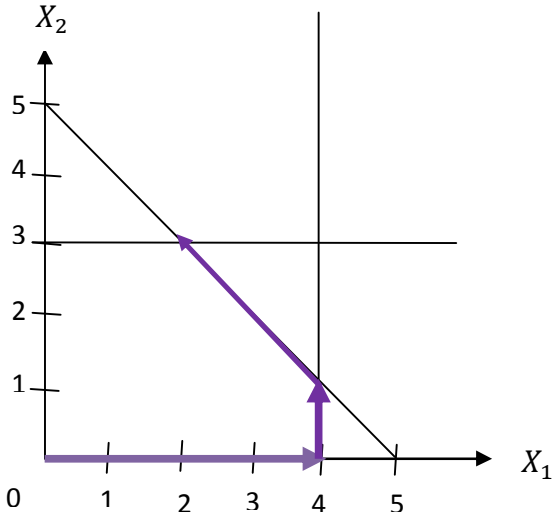


FIGURE2.2– Exploration dirigée d'un polytope

Théorème (Condition nécessaire et suffisante d'optimalité) [1]
La base $\mathfrak{B}$ non dégénérée est optimale ssi

$$\forall j \epsilon \mathcal{N}, \Delta_j = c_j - \sum_{i \epsilon \mathfrak{B}} c_i \bar{a}_{ij} \geq 0$$

Démonstration. Condition suffisante
Soiti ϵ $\mathfrak{B}$. On a

$$x_1 + \sum_{j \epsilon \mathcal{N}} \bar{a}_{ij}\, x_j = \bar{b}_i$$

D'où

$$x_i = \bar{b}_i - \sum_{j \epsilon \mathcal{N}} \bar{a}_{ij}\, x_j$$

Soit x une solution admissible. On a alors

$$\begin{cases} z(x) = \sum_{j=1}^{n} c_j x_j \\ = \sum_{i \epsilon \mathfrak{B}} c_i x_i + \sum_{j \epsilon \mathcal{N}} c_j x_j \\ = \sum_{i \epsilon \mathfrak{B}} c_i \left(\bar{b}_i - \sum_{j \epsilon \mathcal{N}} \bar{a}_{ij} x_j \right) + \sum_{j \epsilon \mathcal{N}} c_j x_j \\ = \sum_{i \epsilon \mathfrak{B}} c_i \bar{b}_i + \sum_{j \epsilon \mathcal{N}} x_j \left(c_j - \sum_{i \epsilon \mathfrak{B}} c_i \bar{a}_{ij} \right) \\ = \bar{z} + \sum_{j \epsilon \mathcal{N}} \Delta_j x_j \end{cases}$$

Où

$$\bar{z} = \sum_{i \epsilon \mathfrak{B}} c_i \bar{b}_i \qquad \Delta_j = c_j - \sum_{i \epsilon \mathfrak{B}} c_i \bar{a}_{ij}$$

Si pour tout $j \epsilon \mathcal{N}$,les $\Delta_j$sont positifs ou nuls alors toutes les solutions admissibles donnent à z une valeur supérieure ou égale à $\bar{z}$.
Condition nécessaire
Supposons Δ_s pour uns ϵ $\mathcal{N}$. Soit $i \epsilon \mathfrak{B}$. Alors :

$$\begin{cases} x_i = \bar{b}_i - \sum_{j\epsilon\mathcal{N}} \bar{a}_{ij}x_j \\ = \bar{b}_i - \sum_{\substack{j\epsilon\mathcal{N} \\ j\neq s}} \bar{a}_{ij}x_j - \bar{a}_{ij}x_s \\ z = \sum_{i\epsilon\mathfrak{B}} c_i\bar{b}_i + \sum_{\substack{j\epsilon\mathcal{N} \\ j\neq s}} \Delta_j x_j + \Delta_s x_s \end{cases}$$

Sans toucher aux autres variables hors base, on peut augmenter x_sd'une valeur θ telle que :

$$\hat{\theta} = min_{\substack{i\epsilon\mathfrak{B} \\ \bar{a}_{is}>0}} \frac{\bar{b}_i}{\bar{a}_{is}}$$

$$= \frac{\bar{b}_i}{\bar{a}_{is}}$$

La nouvelle solution de base est alors telle que :

$$\begin{cases} x_s = \hat{\theta} \\ x_j = 0\ j\epsilon\mathcal{N}, j \neq s \\ x_r = 0 \\ x_i = \bar{b}_i - \bar{a}_{is}\ i\epsilon\mathfrak{B}, i \neq r \end{cases}$$

Ainsi

$$\bar{z} = \sum_{i\epsilon\mathfrak{B}} c_i\bar{b}_i + \Delta_i\hat{\theta}$$

Exemple 1

$$minZ(X) = 2x_1 - 3x_2 + 5x_3$$

s.c.

$$x_1 - 2x_2 + x_2 - x_2 = 4$$

$$x_2 + 3x_3 + x_5 = 6$$

$$2x_1 + x_3 + 2x_4 + x_6 = 7$$

$$x_1, \dots, x_6 \geq 0$$

Ici, $\mathfrak{B} = \{1,2,3\}$ et$\mathcal{N} = \{2,4,5\}$. Autrement dit :

$$B = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 3 & 0 \\ 2 & 1 & 1 \end{pmatrix} B^{-1} = \frac{1}{3}\begin{pmatrix} 3 & -1 & 0 \\ 0 & 3 & 0 \\ -6 & 1 & 3 \end{pmatrix}$$

$$\bar{B}^{-1}N = \frac{1}{3}\begin{pmatrix} -7 & -3 & -1 \\ 3 & 0 & 1 \\ 13 & 12 & 1 \end{pmatrix} B^{-1}b = \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}$$

La solution de base est alors :

$x_1 = 2$ $x_3 = 2$ $x_6 = 1$

$x_2 = 0$ $x_4 = 0$ $x_5 = 0$

$\bar{z} = 14$

Et

$\Delta_2 = 0$ $\Delta_4 = 0$ $\Delta_5 = -1$

C'est une solution dégénérée. En appliquant le théorème précédent :

$$\begin{cases} z = 14 + 2x_4 - x_5 \\ x_1 = 2 + \frac{7}{3}x_2 + x_4 + \frac{1}{3}x_5 \\ x_2 = 2 - \frac{1}{3}x_2 - \frac{1}{3}x_5 \\ x_6 = 1 - \frac{13}{3}x_2 - 4x_4 - \frac{1}{3}x_5 \end{cases}$$

On trouve ici $\hat{\theta} = 3$. Une nouvelle solution de base est donc :

$\mathfrak{B}' = \{1,3,5\}$ $\mathcal{N}' = \{2,4,6\}$

D'où

$x_1 = 3$ $x_2 = 0$ $\bar{z}' = 11$

$x_3 = 1$ $x_4 = 0$

$x_5 = 3$ $x_6 = 0$

On recalcule les ¢i et on constate qu'ils sont tous positifs ; ceci est notre critère d'arrêt.

2.3.2.2 Algorithme du simplexe pour la minimisation

L'algorithme 1 détaille l'algorithme du simplexe dans le cas d'un problème de minimalisation.

2.3.2.3 Méthode des tableaux

Soit le problème :

$$max 4x_1 + 12x_2 + 3x_3$$

s.c.

$$\begin{cases} x_1 \leq 1000 \\ x_2 \leq 500 \\ x_3 \leq 1500 \\ 3x_1 + 6x_2 + 2x_3 \leq 6750 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

Mettons-le d'abord sous forme standard. On a alors le problème équivalent :

$$max 4x_1 + 12x_2 + 3x_3$$

Algorithme 1 Algorithme du simplexe pour la minimisation

Déterminer une solution de base réalisable
Calculer $b^{-1}N = (a_{ij})$ avec $i \epsilon \mathfrak{B}, j \epsilon \mathcal{N}$
Calculer les coûts réduits $\Delta_j = c_j - \sum_{i \epsilon \mathfrak{B}} c_i \bar{a}_{ij}$ $(j \epsilon \mathcal{N})$
Si $\forall j \epsilon \mathcal{N}, \Delta_j \geq 0$ alors
La solution considérée est optimale ; fin
Sinon
j Ensemble des indices de $\mathcal{N}$ *t.q.* $\Delta_j < 0$
Si pour un indice $j \epsilon J, \forall i \epsilon \mathfrak{B}, \bar{a}_{ij} < 0$ alors
Optimum $= -\infty$ *; fin*
Sinon
Choisir s *t.q.* $\Delta_s = min_{j \epsilon J} \Delta_j$ (critère d'entrée)
Déterminer r $\epsilon \mathfrak{B}$ *tel que* $\frac{\bar{b}_r}{\bar{a}_{ij}} = \min_{\substack{i \epsilon \mathfrak{B} \\ \bar{a}_{is} > 0}} \frac{\bar{b}_i}{\bar{a}_{is}}$
Fin si
Fin si
Considérer la nouvelle base déduite de $\mathfrak{B}$ en remplaçant la colonne r par la colonnes.
Reprendre à partir des calculs de $B^{-1}N$
S.C.

$$\begin{cases} x_1 + x_4 \leq 1000 \\ x_2 + x_5 \leq 500 \\ x_3 + x_6 \leq 1500 \\ 3x_1 + 6x_2 + 2x_3 + x_7 \leq 6750 \\ x_1, \dots, x_7 \geq 0 \end{cases}$$

En posant le problème sous la forme $Ax = b$, la matrice A vaut donc :

$$A = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 3 & 6 & 2 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad b = \begin{pmatrix} 1000 \\ 500 \\ 1500 \\ 6750 \end{pmatrix}$$

Une solution de base réalisable est :

$$x_1 = 0 \qquad x_4 = 1000$$

$$x_2 = 0 \qquad x_5 = 500$$

$x_3 = 0$ $x_6 = 1500$
$\bar{z} = 0$ $x_7 = 6750$

On trouve$\Delta_1 = 4, \Delta_2 = 12, \Delta_3 = 3, et\ \Delta_4 = \cdots = \Delta_7 = 0.$
En appliquant l'algorithme du simplexe, on trouve $s = 2\ et\ r = 2.$
On a alors :

$$c_i = \begin{pmatrix} 0 \\ 12 \\ 0 \\ 0 \end{pmatrix}\ i \in \mathfrak{B} = \begin{pmatrix} 4 \\ 2 \\ 6 \\ 7 \end{pmatrix} A = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 3 & 0 & 2 & 0 & -6 & 0 & 1 \end{pmatrix} b = \begin{pmatrix} 1000 \\ 500 \\ 1500 \\ 3750 \end{pmatrix}$$

Et $\Delta_1 = 4, \Delta_2 = 0, \Delta_3 = 3, \Delta_4 = 0, \Delta_5 = -12, \Delta_6 = \Delta_7 = 0$. Ce qui donne :

$x_1 = 0$ $x_4 = 1000$
$x_3 = 0$ $x_2 = 500$
$x_5 = 0$ $x_6 = 1500$
$\bar{z} = 6000$ $x_7 = 3750$

Et une dernière itération en utilisant la ligne 1 comme pivot :

$$c_i = \begin{pmatrix} 4 \\ 12 \\ 0 \\ 0 \end{pmatrix}\ i \in \mathfrak{B} = \begin{pmatrix} 1 \\ 2 \\ 6 \\ 7 \end{pmatrix} A = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & -3 & -6 & 0 & 1 \end{pmatrix} b = \begin{pmatrix} 1000 \\ 500 \\ 1500 \\ 750 \end{pmatrix}$$

Et $\Delta_3 = 3, \Delta_4 = -4, \Delta_5 = -12, \Delta_1 = \Delta_2 = \Delta_6 = \Delta_7 = 0.$
Il faut ensuite encore quelques autres itérations. La solution optimale qu'on trouve ainsi est :

$x_1 = 250$ $x_4 = 750$
$x_2 = 500$ $x_5 = 0$
$x_3 = 1500$ $x_6 = 0$
$\bar{z} = 0\ 11500$ $x_7 = 0$

Cet algorithme n'est pas fait pour être fait à la main, cependant.

2.4 Formulation et modélisation du problème

[4] On peut formuler le TSP de manière équivalente en associant à chaque couple *(i, j)* de villes à visiter *(i = 1 à n, j = 1 à n et i ≠ j)* une distance $\delta_{i,j}$ égale à $d_{i,j}$ s'il existe un moyen d'aller directement de *i* à *j (i.e,(i, j) Є U* dans G), fixée à ∞ sinon et une variable de succession, $x_{i,j}$, binaire, qui prend la valeur *1* si la ville *j* est visitée immédiatement après la ville i dans la tournée et qui prend la valeur *0* sinon. Le TSP est alors modélisé par :

$$\begin{cases} Min \sum_{i=1}^{n}\sum_{i=1}^{n} \delta_{i,j} x_{i,j} \\ \sum_{j=1}^{n} x_{i,j} = 1, \forall i = 1..n \\ \sum_{i=1}^{n} x_{i,j} = 1, \forall j = 1..n \\ \sum_{i \in S, j \notin S}^{n} x_{i,j} \geq 2, \forall S \subset X, S \neq \emptyset \\ x_{i,j} \in \{0,1\}, \forall i = 1..n, \forall j = 1..n \end{cases}$$

Les deux premières contraintes traduisent le fait que chaque ville doit être visitée exactement une fois; la troisième contrainte interdit les solutions composées de sous-tours disjoints, elle est généralement appelée contrainte d'élimination des sous-tours.

2.5. Les méthodes de résolution du problème de voyageur de commerce

2.5.1. Méthode algorithmique [3]

Considérons un graphe non orienté et connexe, sur lequel deux sommets ont un rôle particulier : source (ou origine) O, puits (ou destination) T. A chaque arête $\{i, j\}$, nous associons une distance $c_{ij} \geq 0$. Nous cherchons le chemin non orienté (ou chaine) le plus court reliant *O* à *T*. Par chemin le plus court, nous désignons celui dont la distance totale (somme des distances des arêtes du chemin) est minimale parmi tous les chemins de *O* à *T*. Afin de procéder, nous affectons à chaque sommet du graphe une étiquette, représentant la meilleure distance connue à un moment donné, de l'origine à ce sommet.

- **Algorithme de Dijkstra**

Il s'agit d'une méthode itérative. *A* chaque itération, nous choisissons le sommet j le plus prés de O et nous fixons $d(j)$, la variable calculant la distance entre 0 et j (le sommet j est dit marqué). Au départ, seul *O* est marqué et $d(0) = 0$. Le sommet le plus prés est choisi parmi les sommets non marqués reliés à au moins un sommet marqué. Le sommet choisi j est celui qui atteint

$$\min_{\text{sommets k non marqués}} \left\{ \min_{\text{sommets i marqués}} d(j)+c_{ik} \right\}$$

$d(j)$ est fixée a cette valeur. Nous nous arrêtons lorsque T est marqué si nous cherchons à connaître le chemin allant de O à T spécifiquement, ou sinon jusqu'à avoir marqué tous les sommets. Par la suite, nous suppose-rons dans la description que nous volons marquer tous les sommets. Plus spécifiquement, nous avons l'algorithme ci-dessous. L'algorithme de Dijkstra s'applique aussi sur un graphe orienté.

- **Enoncé de l'algorithme de Dijkstra**

Soit $G = (N, A)$ un graphe connexe, de longueur d'arêtes non-négative (la longueur des arêtes est donnée par une fonction $\ell: A \longrightarrow \mathbb{R}^+$, une source $O \in N$. Désignons par S l'ensemble des sommets marqués.
Initialisation. Pour tout $u \in N$, *faire*$- dist(u) \coloneqq \infty; -pred(u) \coloneqq NULL$. Où $\text{pred}(u)$ indique le prédécesseur de u sur le plus court chemin de O à u .Le nœud origine prend un rôle particulier, et comme il est évitent que le plus court chemin de O à O a une longueur nulle, nous posons $dist(O) = 0$. Comme à ce stade, aucun nœud n'a encore été marqué, nous posons $S = \emptyset$.
2. Tant que $\neq N$, prenons u tel que

$$dist(u) \leq dist(v)\ ;\ \forall\, v \in N \backslash S.$$

Posons $S : = S\ U\ \{\}$,et pour tout arc (u, v) existant, si

$$dist(v) < dist(u) + \ell(u, v)$$

Posons

$$dist\ (v) : = dist(u) + \ell(u, v)$$
$$Pred\ (v) = u$$

Il est possible de montrer qu'une fois un sommet u marqué, $dist(u)$ donne la longueur du plus court chemin de O à u.

Exemple
Un étudiant en quête d'une université projette de visiter les campus de trois universités du Maine au cours d'un voyage unique, débutant et finissant a l'aéroport de Portland. Les trois établissements sont dans les villes de Brunswick, Lewiston, et Waterville, et l'étudiant ne veut visiter chaque ville qu'une seule fois, tout en maintenant le trajet total le plus court possible. Les distances entre ces villes sont données dans la Table 1.1

ville	Portland	Brunswick	Lewiston	Waterville
Portland	0	26	34	78
Brunswick	26	0	18	52
Lewiston	34	18	0	51
Waterville	78	52	51	0

Table 2.1-Distances entre les villes (miles)

L'étape la plus importante dans la construction d'un modèle est le choix des variables qui vont entrer en jeu. Dans la présente cas, puisque n'importe quel trajet consiste en une série de petits déplacements entre deux villes, il est raisonnable d'assigner des variables aux décisions de partir ou non d'une ville vers une autre. Pour plus de faciliter, numérotons et *4* pour Waterville. Ainsi, nous aurons une variable $x_{1,2}$égale a *1* si l'étudiant voyage de Portland a Brunswick au cours de son parcours total, et 0 sinon. Puisqu'il n'y a pas de voyage d'une ville vers cette même ville, nous avons d'ores et déjà les contraintes

$$x_{i,i} = 0, i = 1 \dots,4 \qquad (1.1)$$

Une fois les variables choisies, nous pouvons essayer de formuler le problème. Ce processus et en fait souvent une manière utiles pour guider le choix des variables. Chaque ville ne devant être visitée qu'une seule fois, elle ne peut apparaitre qu'une seule fois comme ville d'arrivé. En d'autres termes, pour j fixé, x_{ij} ne peut être non-nul que pour un i donné, avec$i \neq j$. Une manière plus simple d'encoder cette information est d'écrire, pour $j = 1, \dots,4$,

$$x_{1,j} + x_{2,j} + x_{3,j} + x_{4,j} = 1$$

Ou de manière plus concise.

$$\sum_{i=1}^{4} x_{ij} = 1, \quad j = 1, \dots.,4. \qquad (1.2)$$

Les contraintes formulées jusqu'à présent ne garantissent aucune forme de trajet ayant même départ et arrivée. Par exemple, l'affectation $x_{1,2} = 1, x_{1,3} = 1, x_{1,4} = 1, x_{2,1} = 1$, et toutes les autres variables égales a 0, satisfont les contraintes (1) et (2). Cette solution décrit toutefois un schéma de visites impossible puisque Portland est l'origine de tous les déplacements aux trois autres villes universitaires, mais n'est destination que depuis Brunswick. Nous avons évidemment aussi besoin des contraintes

$$\sum_{j=1}^{4} x_{ij} = 1 \quad , i = 1, \dots.4. \qquad (1.3)$$

Afin d'assurer que chaque ville ne serve d'origine que pour exactement un déplacement vers une notre ville. Finalement, afin d'obtenir un véritable trajet ayant même origine et départ, nous devons rejeter les affectations qui décrivent des groupes déconnectés de petits déplacements comme $x_{1,2} = 1, x_{2,1} = 1, x_{3,4} = 1, x_{4,3} = 1$, avec toutes les autres variables égale à *0*. Nous pouvons forcer ceci avec les contraintes.

$$x_{i,j} + x_{j,i} \leq 1, i = 1 \dots,4, et\ j = 1, \dots,4.$$

Cette contrainte exclut tout mini-cycle.

Les contraintes définies, nous devons décrire la distance totale associée à n'importe quel parcours autorisé. Puisque nos variables ont seulement comme valeurs possibles *0* ou *1*, nous pouvons multiplier chacune d'elle par la distance correspondante entre les deux villes indexes, et les additionner :

$$\sum_{i=1}^{4}\sum_{j=1}^{4} x_{i,j}a_{i,j}$$

Notre modèle mathématique consiste à minimiser cette fonction, dite fonction objectif par rapport aux variables $x_{i,j}$, tout en satisfaisant les contraintes préalablement décrites :

$$\begin{cases} min_x \quad \Sigma_{i=1}^{4}\Sigma_{j=1}^{4} x_{i,j}a_{i,j}, \\ x_{i,i} = 0, i = 1, \dots, 4, \\ \sum_{i=1}^{4} x_{i,j} = 1, j = 1, \dots, 4, \\ \sum_{j=1}^{4} x_{i,j} = 1, i = 1, \dots, 4, \\ x_{i,j} + x_{j,i} \leq 1, i = 1, \dots, 4, et\ j = 1, \dots, 4, \\ x_{i,j} \epsilon \{0,1\}, i = 1, \dots, 4, et\ j = 1, \dots, 4, \end{cases}$$

Ici,$x = (x_{i,j})i = 1, \dots, 4, j = 1, \dots, 4$, et s.c, ''sous les contraintes''. Le problème d'optimisation ainsi construit constitue un programme mathématique.
Le problème de visites d'universités est assez petit que pour être résolu explicitement, sans recourir à des méthodes d'optimisation numérique. Puisqu'il y a chacun d'eux pourrait âtre facilement calculée, et nous choisissons le parcours de longueur minimale, qui est ici

Portland→ Brunswick→ Waterville →Lewiston →Portland

Avec une distance totale de 163 miles. Il est clair qu'une telle stratégie de résolution ne fonctionne plus comme le nombre de villes augmente.

2.5.2. Méthode heuristique [5]

[5] Imaginons un voyageur de commerce qui doit visiter de nombreuses villes, son point de départ. Si l'on dispose de la matrice des couts de transport de villes a ville (qu'on pourra supposer non symétrique pour corser le problème), le voyageur de commerce recherche le circuit Hamiltonien de valeur minimale. Pendant de longues années, de nombreux chercheurs ont tenté d'inventer un algorithme pour résoudre ce problème, on n'en trouvait pas et l'on n'était pas certain de la validité des solutions ingénieuses proposées par les uns et les autres pour des exemples numérique relativement

Importants (entre 40 et 50 villes). Bien entendu, ces solutions étaient obtenues à l'aide d'heuristique, car il n'était pas question d'énumérer les *(n-1) !* Circuits possibles. C'est alors que Little et autre (novembre 1963) ont appliqué au problème une procédure de recherche arborescente, qui a permet d'obtenir des résultats exactes en des temps de calcul parfois assez longs. De nombreux perfectionnements sont apportés continuellement a des méthodes de ce type, nous citerons seulement la méthode de Lin (2), qui parait aujourd'hui polariser les recherches, a cause de sa rapidité bien qu'elle ne soit qu'approchée (k-optimale). En fait de recherche arborescent, nous n'avons l'intention de ne traiter ici que l'exemple du voyageur de commerce, ne posant aucune question quant a la convergence et à la finitude de la méthode employée. Nous laissons au lecteur le soin de se reporter, en cas de besoin, pour d'autre problème de recherche arborescente mois classiques, à l'énorme littérature consacrée, même en langue française, à ces sujets.

Signalons, néanmoins que les initiales *S.E.P.*, très employées en France pour caractériser la classe des méthodes mises au point, en 1964-65, par les chercheurs d'une société de services, signifient ''séparation et évaluation progressives''. Nous allons justement donner un exemple d'un principe de séparation dichotomique, consistant à augmenter ou non un ensemble d'un élément, en mesurant l'efficacité de l'une ou de l'autre décision par l'évaluation de la borne inférieure d'un cout.

Pour éviter des longueurs, l'exemple choisi sera celui d'un voyageur de commerce demeurant dans la ville *A* et désireux de se rendre une fois et une seule dans les villes *B, C, D, E* et *F*, avant de revenir chez lui. La matrice des coûts est donnée ci après (tirets remplacent des valeurs infinies), il s'agit évidemment de déterminer, parmi les *120* circuits hamiltoniens, celui de valeur minimale. Nous diviserons les opérations en blocs.

Bloc *A*. Comme cette opération ne change évidement rien au problème, on soustrait le plus petit événement de chaque rangée (d'abord les lignes, puis les colonnes), la matrice résultante comporte donc au mois un zéro par rangé

	A	B	C	D	E	F
A	–	6	7	3	1	3
B	7	–	8	2	9	7
C	5	10	–	10	1	7
D	8	6	5	–	5	1
E	7	7	6	7	–	4
F	9	8	8	5	3	–

Matrice 1

Bloc B. On calcule une borne inférieure de la valeur du circuit de valeur minimale cherché. Elle n'est autre que la somme des valeurs soustraites de la matrice du bloc *A*. En effet, si la matrice *3* permettait de trouver un circuit utilisant uniquement des arcs de cout *0*, la valeur de ce circuit coïnciderait avec la somme des coûts retranchés à la matrice initiale.

	A	B	C	D	E	F
A	–	5	6	2	0	2
B	5	–	6	0	7	5
C	4	9	–	9	0	6
D	7	5	4	–	4	0
E	3	3	2	3	–	0
F	6	5	5	2	2	–

Matrice 2

	A	B	C	D	E	F
A	–	2	4	2	2	2
B	2	–	4	0	7	5
C	1	2	–	9	0	6
D	4	2	2	–	4	0
E	0	0	0	3	–	0
F	3	2	3	2	2	–

Matrice 3

Dans ce cas présente : *B*=1+2+2+2+4+3+3+3+2=20.
La racine *R* de l'arborescence recevra l'évaluation *B*=20.

Bloc C. on calcule les coûts de substitution des arcs de cout minimal *(0)* et l'on retient le maximum minimorum (ou l'un d'eux en cas d'égalité).
Expliquons d'abord cette notion de regret(1) : soit, par exemple, le 0 qui occupe la relation *(B, D)* dans la matrice *3*, il signifie qu'on a intérêt a employer l'arc *(B, D)* qui, a priori, est a recommander, était de cout résiduel le plus petit sur la ligne. Combien paierait-on si l'on décidait de ne pas l'utiliser ? Comme il faudrait tout de même passer par les points *B* et D, la meilleur solution consisterait a atteindre *D* par *(A, D)* : cout 2- ou encore *(F, D)* : cout *2* – et a quitter *B* par *(B, A)* : cout *2*.
La somme de ces deux minimums, soit 2+2=4, donne le regret minimal relatif au zéro de la relation *(B, D)*.
On détermine ainsi un cout de substitution pour tous les zéros de la matrice *3*.
Finalement, on choisit d'examiner les conséquences qu'auraient la sélection ou le rejet de l'arc correspondant au plus fort des regrets minimaux. En cas d'égalité, on se donne une règle arbitraire pour retenir l'un des regrets maximaux, par exemple le premier rencontré lors du balayage de la matrice.
On obtient ainsi la matrice *4*, dont le regret maximal, soit *4*, concerne l'arc *(B, D)*.

Bloc *D*. on sait a présent, après qu'on a procédé à l'évaluation des couts de substitution des zéros d'une matrice, et cela a toute étape du problème, qu'il reste possible :

- Soit de renoncer à utiliser la relation *(X, Y)* de regret le plus fort
- *p(X, Y)* parmi les regrets minimaux, étant entendu que la valeur de ce regret doit être ajoutée à la borne inférieure, calculée plus haut ;

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{pmatrix} & & & & 2 & \\ & & & 4 & & \\ & & & & 1 & \\ & & & & & 2 \\ 1 & 2 & 2 & & & 2 \\ & & & & 2 & \end{pmatrix} \end{array}$$

Matrice 4

Regrets correspondant aux zéros de la matrice 5

- Soit de prendre en considération la relation *(X, Y),* de manière examiner les conséquences de ce choix.

C'est ainsi que se développera l'arborescence réellement parcourue dans la recherche, par la création d'un sommet de type I : *NON (X, Y)* et d'un sommet de type *II : (X, Y).*
D'où déjà, pour le sommet de type I, correspondant à la décision *NON (X, y),* c'est-à-dire la décision de renoncer a l'utilisation de *(X, Y)* :

$$B := B + P(X, Y).$$

Dans l'exemple cité, la borne du sommet *NON (B, D)* de l'arborescence est $B := 20+4=24$.

Bloc E. 1) Au contraire, si l'on prend en considération l'arc *(X, Y),* supprimer la ligne X et la colonne Y de la matrice des couts réduits, puis, par l'introduction de cout infinis, interdire les circuits parasites (c'est-a-dire qui ne peuvent être empruntés sans empêcher la continuation de la recherche d'un circuit hamiltonien).

N.B. les circuits parasites n'existent plus lorsqu'on est parvenu à une matrice de dimension 1×1.

2) Il faut maintenant vérifier, s'il existe toujours dans la matrice réduite *(matrice 5)* un zéro par ligne et par colonne. Sinon, faire apparaitre un zéro par rangée *(matrice* bis). Dans l'exemple, il n'y a rien à modifier ; la *matrice* 5.

3) la borne du sommet *(X, Y)* de l'arborescence devient, le cas échéant :

B : = B+sommes des éléments ôtés au & 2 du bloc E.
Dans notre exemple, elle reste égale à 20.

$$\begin{array}{c} \\ A \\ B \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & E & F \end{array} \\ \begin{pmatrix} - & 2 & 4 & 0 & 2 \\ 1 & 6 & - & 0 & 6 \\ 4 & \infty & 2 & 4 & 0 \\ 0 & 0 & 0 & - & 0 \\ 3 & 2 & 3 & 0 & - \end{pmatrix} \end{array}$$

Matrice 5

4) Si l'on a attient une matrice 1 × 1, arrêter les calculs, car on a la solution. Sinon passer au bloc suivant. C'est ce qu'on doit faire en poursuivant le traitement de l'exemple.

Bloc F. A cette étape du problème, inspecter d'abord les bornes des sommets pendants de l'arborescence et choisir le sommet de borne la plus petite.
Dans l'exemple, on a *NON (B, D) :* 24*, (B, D) :* 20 ; on choisit donc *(B, D).*Si ce sommet correspond au type *II : (X, Y),* il faut revenir au bloc C. sinon, il faut aller au bloc *G*. Dans l'exemple, on ira a *C*, puisqu'on a choisi un sommet de type II : *(B, D).*

Bloc G. il s'agit du cas ou l'on a choisi, au bloc *F*, d'après la valeur de borne, un sommet de type I, c'est-à-dire *NON (X, Y).* Dans ce cas, interdire *l'arc (X, Y)* en mettant un coût infini dans la case correspondante de la matrice des coûts réduits. Puis soustraire le plus petit élément de la ligne X et de la colonne Y, ce qui correspond a l'emploi de la substitution envisagée au bloc C.
Enfin, revenir au bloc C. Continuation de l'exemple.
Rappelons d'abord la situation à laquelle la procédure nous a conduits. Le problème se ramène à l'étude de la matrice 5 bis (qui se confond avec la matrice 5, dans l'exemple). Sur l'arborescence, on a crée les deux sommets *NON (B, D) et (B, D)* ; le premier a *24* pour valeur de la borne inférieur, le second, 20. On a choisi le sommet *(B, D),* de type II faut donc revenir au bloc C.

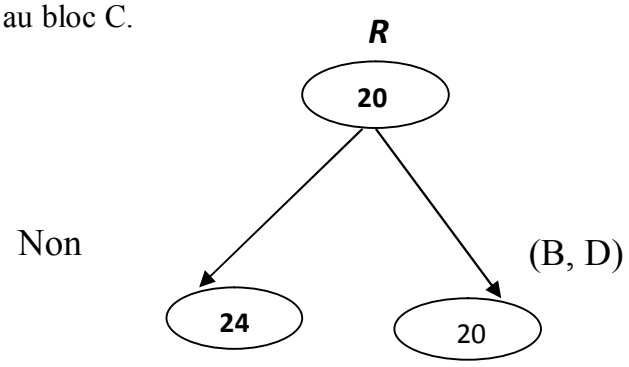


Figure2.3

1). Matrice 6 : évaluation des zéros de la matrice 5 bis. Prendre en considération le regret maximal, concernant *(A, E).* *A* partir du sommet *(B, D)* = 20, développer l'arborescence : *NON (A, E)*= 20+2=22*, (A, E)= ?*
Supprimer la ligne *A* et la colonne *E* de la matrice *5* bis, interdire *(E, A),* d'où la matrice. Cette matrice ne comporte plus un zéro sur chacune de ses rangées.
Soustraire 1 de la ligne c et *2*de la ligne *F*.
On obtient la matrice 7 bis et la borne du sommet (A, E) : 20+3=23.
Choisir le sommet pendant de la borne la plus petite : c'est *NON (A, E),* de type I, avec *B*=22.

2). Il faut évidement reprendre la matrice *5* (matrice des couts relative au sommet de l'arborescence précédant celui qu'on vient d'adopter) pour exécuter le bloc *G*, d'où la matrice *8*. Revenir C.

$$\begin{array}{c} \\ C \\ D \\ E \\ F \end{array}\begin{array}{c} \begin{array}{cccc} A & B & C & F \end{array} \\ \begin{pmatrix} 0 & 5 & - & 5 \\ 4 & \infty & 2 & 0 \\ \infty & 0 & 0 & 0 \\ 1 & 0 & 1 & - \end{pmatrix} \end{array}$$

Matrice 7 bis

$$\begin{array}{c} \\ A \\ C \\ D \\ E \\ F \end{array}\begin{array}{c} \begin{array}{ccccc} A & B & C & E & F \end{array} \\ \begin{pmatrix} - & 0 & 2 & \infty & 0 \\ 1 & 6 & - & 0 & 6 \\ 4 & \infty & 2 & 4 & 0 \\ 0 & 0 & 0 & - & 0 \\ 3 & 2 & 3 & 0 & - \end{pmatrix} \end{array}$$

Matrice 8

Matrice 9. Evaluation des zéros de la matrice *8*. Borne de *NON (D, F)*=22+2=24, borne de *(D, F)= ?*

Matrice 10 : suppression de la ligne *D* et de la colonne *F* de la matrice 8 interdiction de *(F, B)*.borne de *(D, F)* =22+0=22.

Calcul de la min 24, 24, 23,22 =22. Repartir de *(D, F),* de type *II*.

$$\begin{array}{c} \\ A \\ C \\ D \\ E \\ F \end{array}\begin{array}{c} \begin{array}{ccccc} A & B & C & E & F \end{array} \\ \begin{pmatrix} & 0 & & & 0 \\ & & & 1 & \\ & & & & 2 \\ 1 & 0 & 2 & & 0 \\ & & & 2 & \end{pmatrix} \end{array}$$

Matrice 9

$$\begin{array}{c} \\ A \\ C \\ E \\ F \end{array}\begin{array}{c} \begin{array}{cccc} A & B & C & E \end{array} \\ \begin{pmatrix} - & 0 & 2 & \infty \\ 1 & 6 & - & 0 \\ 0 & 0 & 0 & - \\ 3 & \infty & 3 & 0 \end{pmatrix} \end{array}$$

Matrice 10

3. La matrice 11 désigne *(F, E)*. Borne de *NON (F, E)*=22+3=25

Borne de *(F, G) =22+1=23* (passage de matrice *12* a la matrice *12 bis)*.

Calcule de la *min* {24, 24, 25, 23,23}=*23*. On décide de continuer a partir de *(F, E),* de borne 23. 4 .la matrice 13 désigne *(A, B)*. Borne de *NON (A, B)*=23+7=30.borne de *(A, B) = 23+0* (matrice 14). Calcul de la min {24, 24, 25, 30, 23,23}=23.
On décide de continuer a partie de *(A, B)* de borne *23*.

5. La matrice 15 laisse le choix entre *(C, A)* et *(E, C)*.Arbitrairement : choix de*(C,A)*.
Borne de *NON(C, A)* =23 + ∞ = +∞.

Borne de *(C, A)* = 23+0 (matrice *16*).

La matrice *16* est de format 1× 1:se garder d'interdire l'arc *(E, C)*.

6. Prendre *(E, C)*.
Borne de *NON (E, C)* =23+∞ = +∞.

Born de *(E, C)* =23++0 = +∞.

On a le circuit hamiltonien *(B, D, F, E, C, A, B),* de valeur 23.

6 bits. On remarque que la borne du sommet *(A, E)* est également 23 ; au numéro 5, on a d'ailleurs choisi de continuer en prenant *(E, C)* parce que la matrice 16 était de format1× 1.

$$\begin{array}{c} \\ A \\ C \\ E \\ F \end{array}\begin{array}{c} \begin{array}{cccc} A & B & C & E \end{array} \\ \begin{pmatrix} & 2 & & \\ & & & 1 \\ 1 & 0 & 2 & \\ & & & 3 \end{pmatrix} \end{array}$$

Matrice11

$$\begin{array}{c} \\ A \\ C \\ E \end{array}\begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} - & 0 & 2 \\ 1 & 6 & - \\ 0 & \infty & 0 \end{pmatrix} \end{array}$$

Matrice 12

$$\begin{array}{c} \\ A \\ C \\ E \end{array}\begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} & 0 & 2 \\ 0 & 5 & - \\ 0 & \infty & 0 \end{pmatrix} \end{array}$$

Matrice 12bis

$$\begin{array}{c} \\ A \\ C \\ E \end{array}\begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} - & 0 & 2 \\ 1 & 6 & - \\ 0 & \infty & 0 \end{pmatrix} \end{array}$$

Matrice 13

$$\begin{array}{c} \\ A \\ C \end{array}\begin{array}{c} \begin{array}{cc} A & C \end{array} \\ \begin{pmatrix} 0 & - \\ \infty & 0 \end{pmatrix} \end{array}$$

Matrice 14

$$\begin{array}{c} \\ C \\ E \end{array}\begin{array}{c} \begin{array}{cc} A & C \end{array} \\ \begin{pmatrix} \infty & \\ & \infty \end{pmatrix} \end{array}$$

Matrice 15

$$\begin{array}{c} \\ E \end{array}\begin{array}{c} \begin{array}{c} C \end{array} \\ (0) \end{array}$$

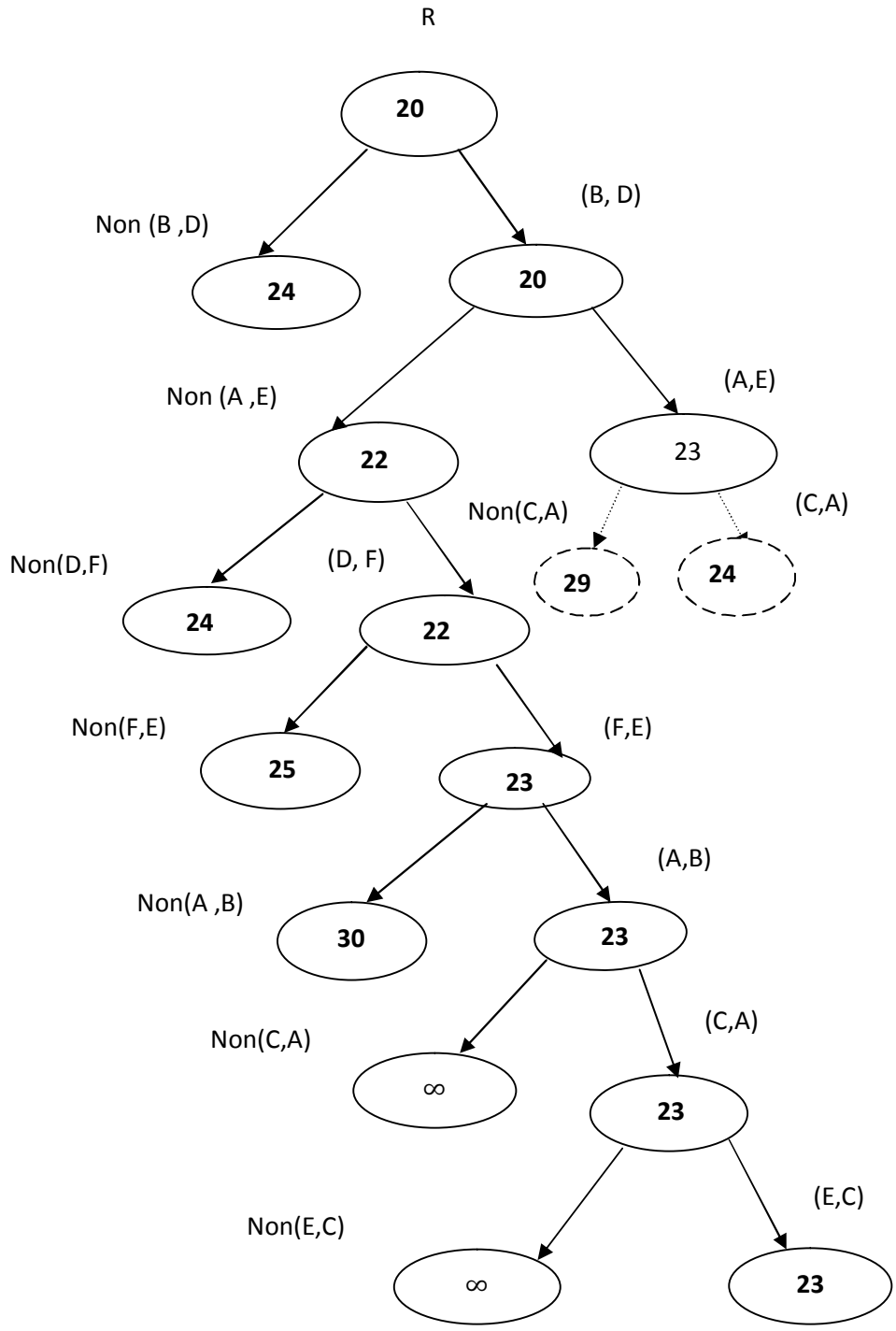
Matrice 16

Il se pourrait donc qu'il existe un autre circuit hamiltonien utilisant l'arc *(A, E)*. Revenant à la matrice 7 bits, on évalue ses zéro et l'on choisit ainsi l'arc*(C, A)*.on a ensuite : borne de *NON (C, A)=23+6=29* et borne de *(C,A)=23+1=24.*

$$\begin{array}{c} \\ C \\ D \\ E \\ F \end{array}\begin{array}{c} \begin{array}{cccc} A & B & C & F \end{array} \\ \begin{pmatrix} 6 & & & \\ & & & 2 \\ & 0 & 1 & 0 \\ & 1 & & \end{pmatrix} \end{array}$$

$$\begin{array}{c} \\ D \\ E \\ F \end{array}\begin{array}{c} \begin{array}{ccc} B & C & F \end{array} \\ \begin{pmatrix} \infty & 2 & 0 \\ 0 & \infty & 0 \\ 0 & 1 & - \end{pmatrix} \end{array}$$

Toutes les autres bornes des sommets pendants de l'arborescence étant maintenant strictement supérieures à 23, on est sur que le circuit optimal trouvé en 6 est unique. La figure 4 : représente très exactement la partie de l'arborescence parcourue ; elle est relativement peu étendue par rapport a l'arborescence ''potentielle'', celle qui contient les cent vingt chemins hamiltonien.



La figure 4

2.6. Le choix de la meilleure solution

Lorsque le temps alloué à la résolution est faible on utilisera plutôt des heuristiques tel que l'Algorithme. Toutefois les méthodes heuristiques ne donnent en général aucune preuve justifiant qu'elles obtiennent la meilleure solution, et même si elles peuvent en pratique être très bonnes, en toute généralité elles ne fournissent qu'une solution approchée.

Chapitre 3

Applications du problème de voyageur de commerce

Dans ce chapitre nous donnons une situation réelle qui se formule en problème de voyageur de commerce.

Les applications du problème du voyageur de commerce sont nombreuses : d'une part, certains problèmes d'optimisation de parcoures en robotique ou en conception de circuits électronique ainsi que certains problèmes de séquencèrent passage de train sur une voie, atterrissage d'avions, processus de fabrication en industrie chimique, etc. s'expriment directement sous forme de TSP. D'autre part, et c'est sans doute la famille d'application la plus importante, les problèmes de transport .Supposez que vous soyez représentant de commerce et que vous ayez à visiter un certain nombre de villes, en sachant que : L'ordre dans lequel vous effectuez vos visites n'a pas d'importance, à la condition que vous les fassiez toutes ; le parcours commence et finit au même point (cette condition n'est pas toujours imposée et ne change pas sur le fond ce qui suit). Votre intérêt, mais plus encore celui de l'entreprise qui vous emploie, est de choisir le parcours le plus court.
Comment allez-vous procéder ? Par exemple, quel itinéraire proposeriez-vous pour visiter les cinq villes *A, B, C, D* et *E* de la figure ci-contre (les distances sont données en kilomètres ?

Supposons que l'on parte de la ville A (on doit donc y revenir). Par exemple, un chemin possible est *A→ C→ B→ D→ E→ A* et il peut être noté *ACBDE (A)* ; sa longueur totale est 440 km *(94+54+82+71+139)*. Ce n'est certainement pas le plus court, car il a été choisi au h.

Comment donc déterminer le chemin le plus court ? Ce qui importe n'est pas tant de trouver une solution pour le problème particulier ci-dessus que d'établir une démarche générale permettant de résoudre le problème pour n' importe quel ensemble de villes : cette démarche s'appelle un algorithme .La première idée, la plus simple et la naturelle, consiste a se rendre systématiquement dans la ville la plus proche de celle où l'on se trouver : sans y réfléchir plus , on pourrait penser que cette méthode résout effectivement le problème posé .Qu' oi de plus logique en effet ? Si l'on part par exemple de A, on obtient de cette façon le parcours *ABCDE(A) de* 391 km (60 +54+67+71+139) : le trajet est déjà plus court que le précédent ,déterminé au hasard Or ,à la question , en en conviendrez, essentielle, cet algorithme donne-t-il la réponse au problème posé ?il faut bien se résoudra e a rependre Le chemin proposé n'est pas le plus court de ceux qui commencent et finissent en A :nous verrons plus bas un itinéraire plus court ,obtenu de façon différent .
Comprenons bien : la méthode proposée peut donner par fois le meilleur chemin ; mais elle peut aussi donner des résultats très mauvais.

Par exemple :

Si l'on part de la ville C, on obtiendrait l'itinéraire *CBADE(C)* de longueur 435 km (54+60+138+71+112) .Soit a peine meilleur que le première parcours *ACBDE(A)* ?décrit a partir de *C(CBDEA(C)),* pourtant choisi au hasard. Notre première idée est donc inadaptée décevants dans ses résulta, malgré son apparence de

Simplicité. La deuxième idée est basée sur une étude exhaustive de tous les parcoure possible : c'est un algorithme inélégant, fastidieux que l'on qualifie parfois de ‹bestial› tant il manque d'originalité mais, lui, mois, donne la bonne réponse en examinant tous les cas : il faut Just être patient(ou avoir un ordinateur).
Comment donc déterminer tous les parcours possibles; sans en oublier un seul ?il suffit de remarquer.
Qu' il existe line correspondance biunivoque entre un itinéraire respectant les conditions imposées et une permutation de l' ensemble *(A ,B,C ,D,E):* ainsi ,a' la permutation *(DCBAE)* correspond l'itinéraire *DCBAE(D)* et à l'itinéraire *ACEBD(A)* la permutation*(ACEBD).*

Bref 5 !=120 itinéraires sont à examiner (autant que de permutations d'un ensemble à 5 éléments) et pour chacun, on calcule très facilement sa longueur. Avec l'aide de MAPLE qui a l'avantage , par l'instruction permute*(A ,B,C,D,E)* de générera liste de toutes les permutations de l'ensemble *A,B,C,D,E* ,l'ordinateur fournit en une fraction de seconde la réponse au problème posé : le parcours le plus court est *ABEDC(A)* ou ceux équivalents comme *BEDCA(B)* ou *CDEBA(C)* obtenus en parcourant les villes dans le même ordre ou dans l'ordre inverse, avec un point de départ différent , pour une distance de 378 km (60+86+71+67+94) . la réponse obtenue amène deux remarques. Tout d'abord, nous comprenons pourquoi notre première idée se rendre toujours dans la ville la plus proche échoue : dans le parcours minimal *ABEDC(A) ,* arrivé en *B*, il faut aller en E, alors que les villes C et D sont touts deux plus proche de B que E. Enfin, et dans la mesure où tous les itinéraires possible sont examinés, il est élémentaire de modifier la procédure MAPLE pour qu'elle donne le chemin le plus long : c'est *ADBCE (A*) avec 525 kilomètre, soit 147 km de plus que le parcours minimal Le gain est loin d'être négligeable et les différents itinéraires obtenue sont de longueur suffisamment variable pour que les mathématiciens se soient penchés sur le problème… Mais cela n'explique pas cependant pourquoi on dépense autant de temps pour les promenades des représentants de commerce…En fait, la résolution de ce problème est utile dans de nombreux autres domaines beaucoup plus stratégique économiquement que la tournée de Monsieur Dupont représentant en casseroles malgré tout le respect que l'on peut avoir pour son travail .Les problèmes d'optimisation sont quotidiens et essentiels dans le monde de l'industrie. Citons le cas d'une entreprise de télécommunications qui doit construire une ligne téléphonique reliant une centaine de postes : elle cherchera a en minimiser la longueur. Les exemples de cette nature sont légions…

Le problème posé dans l'exposition MATH 2000 se résout beaucoup plus difficilement : il y a cette fois 10 villes dont les distances mutuelles sont données dans le tableau suivant, dans une unité qui n'est pas précisée qui pourrait être des heures d'avion par exemple.

Athènes									
3	Berlin								
4	1	Bruxelles							
5	4	3	Lisbonne						
4	1	1	3	Londres					
4	3	2	1	2	Madrid				
4	1	2	5	2	4	Oslo			
3	2	1	2	1	2	2	Paris		
1	2	2	4	2	3	3	2	Rome	
3	2	2	5	2	4	2	3		Varsovie

10! =3628800 itinéraires sont à examiner : MAPLE, et la modeste mémoire de mon PC, ne peuvent plus gérer la liste de toutes les permutations d'un ensemble a 10 élément. En testant aléatoirement un certain nombre de permutations (environ 6 500 000) une a une et en calculant la longueur de l'itinéraire correspondant, les parcours les plus cours obtenus sont tous de longueur 16 ; citons par exemple(voir la carte ci-dessous) :

Athénes-Rome-Londres-Paris-Lisbonne-Madride-Bruxelles-Berlin-Oslo-Varsovir-Athénes

Est-ce le meilleur ? C'est très probable, vu le nombre d'itinéraire testés, mais ce n'est pas sûr .La question est ouverte. Signalons que la même méthode par essai aléatoire donne facilement de très mauvais itinéraires, comme Athénes-Rome-Londres-Paris-Lisbonne-Madride-Bruxelles-Berlin-Oslo-Varsovir-Athénes, de longueur 33, soit plus du double de la longueur précédente.
Par l'algorithme exhaustif, notre problème est résolu de manière théorique, mais sommes-nous satisfaits pour autant ? Pourquoi-nous par exemple prévoir par cette méthode l'itinéraire le plus cour d'un voyageur de commerce souhaitant visiter disons quelque 30 villes ce qui ne doit pas être si exceptionnel pour un représentant dynamique ? Cette fois, 30 ! Soit environ 2×10^{22} itinéraire, sont à examiner. Un ordinateur très puissant de nos jours est capable d'effectuer environ 10^{10} opérations élémentaires par seconde. En supposant que le calcul de la longueur d'un itinéraire

soit une de ces opérations élémentaires ce n'est bien sur qu'une très grossière approximation, cet ordinateur mettra donc environ.

$$\frac{2\times 10^{22}}{10^{10}} = 2\times 10^{22} \quad \text{secondes}$$

Soit encoure environ 6×10^{14} années. Le lecteur attentif aura remarqué que cette durée dépasse déjà largement l'âge de l'Univers estimé à quelque 10 ou 20 milliards d'années : la patience ne suffit plus… Notre algorithme se trouve dans l'impossibilité totale de répondre a la question posée pour un nombre aussi restreint de villes que 30…Il est clair que les performances obtenues sont directement liées a l'ordinateur utilisé, plus ou mois rapide , mais il faut avoir conscience que l'explosion combinatoire , liée a la présence de la factorielle, rend de toute façon l'algorithme exhaustif très vite inaccessible a tout ordinateur classique, aussi rapide soit-il faut-il se r esbigner ? Même dans ce qui apparait comme une impasse, les mathématiciens et les informaticiens ne manquent pas d'idées.

Conclusion

Dans notre travail, nous avons traité un sujet sur les mathématiques appliquées précisément sur la théorie des graphes. Nous avons étudié le problème de voyageur de commerce. Nous avons donné les définitions qui seront utiles pour la compréhension du mémoire, ensuite nous avons présenté la modélisation du problème et les méthodes proposées pour la résolution du problème. Nous avons fini par l'étude d'une situation en utilisant le PVC.

Quelques méthodes de résolution sont programmables par ordinateur. Nous avons remarqué que le temps d'exécution augmente quand le nombre de villes devient plus grand.

Bibliographie

[1] Alain Billionnet. Notes de cours 2010-2011. Mathématique-recherche opirationnelle.

[2] Boufelgha Ibrahim, Broad caste domination dans les prismes complémentaire, USTHB.

[3] Fabian Bastin : Modèles de Recherche Opérationnelle, page 65, 66. Université de Montréal, 2006.

[4] M .Minoux & A. Billionnet, informatique et recherche opérationnelle université Pierre et Marie Curie-Paris 6, (1999 /2000)

[5] Robert Faure : professeur titulaire de la chaire de recherche opérationnelle au conservatoire nationale des arêtes et métiers détache a la faculté des sciences de rabat Méthode de recherche opérationnelle méthode et exercices d'application.

Résumé

Dans notre travail, nous étudions le problème de voyageur de commerce. Nous commençons par la présentation de quelques notions et définitions que nous aurons besoin au long de ce travail. En suite nous donnons la modélisation de ce problème sous forme d'un programme linéaire, ainsi les méthodes proposées pour la résolution du PVC et quelques exemples explicatifs. Nous finissons notre travail par une application du PVC à résolution d'une situation réelle.

Table des matières

Chapitre 3 : Application du problème de voyageur de commerce

Conclusion générale.