République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A. Mira de Béjaia

Faculté des Sciences exactes

Département de Recherche Opérationnelle

Mémoire préparé

En Vue de l'Obtention du Diplôme de Master en Recherche Opérationnelle

Spécialité : **Modélisation Mathématique et Évaluation de Performance Des Réseaux**

Thème

Optimisation dans les réseaux

Présenté par :

M^{lle} BOUDJEMA Wissame M^{lle} BOURAS Sonia

Devant le jury composé de :

Président :	M^{r} BOUDRIES Abdelmalek
Rapporteur :	M^{r} KABYL Kamal
Examinateur :	M^{r} ATHMANI Mouloud
Examinateur :	M^{r} TAOUINET Smail

Juin 2016

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

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

Nous tenons également à remercier les membres du jury pour l'honneur qu'il nous ont fait en acceptant de juger ce travail, et d'avoir consacrer leurs temps pour sa lecture.

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

Nous tenons à remercier M^r S.Ali pour son Précieux aide.

Enfin nous tenons à rendre hommage à toutes nos familles et nos amis pour le soutien qu'ils nous ont apportés durant toutes ces années d'études .

***J**e dédie ce travail à :*

***M**es très chers parents à qui je doit tout et à qui je rendrais jamais assez*

***M**es très chères soeurs*

***M**on très cher frère*

***M**es très chers beaux frères*

***M**on adorable nièce*

***M**a très chère I.Fairouz et à* M^{r} *A.El Hocine*

***T**oute ma famille et tous mes amis*

Bouras Sonia

Je dédie ce travail :

***A** mes très chers parents à qui je doit tout et à qui je rendrais jamais assez.*

A ma très chère sœur

A mes très chers frères.

A toute ma famille.

A l'ensemble de mes amis.

Boudjema Wissame

Table des matières

Introduction générale

La recherche opérationnelle peut être considérée comme un ensemble de méthodes utilisables pour élaborer des meilleures décisions. Elle permet de modéliser des problèmes issus des organisations du monde réel, identifier les méthodes de résolutions et les outils les plus adaptées face à un problème pratique. Elle fait partie de "l'aide à la décision" qui est un ensemble des techniques permettant pour une personne donnée d'opter pour la meilleure prise de décision possible.

Dans l'avant propos de son livre "Graphes et Hypergraphes", Claude Berge note que la théorie des graphes a eu un développement bien étrange, d'abord apparue sous forme de curiosités mathématiques (les pont de Konigsberg), puis devenue un outil pour l'´etude des circuits électriques (Kirchhoff), elle a été utilisée dans la chimie (modélisation des structures), la psychosociologie et l'économie avant même d'avoir été constituée. Elle est devenue aujourd'hui une des branches les plus florissantes de l'algèbre moderne, celle à laquelle on fait appel dans la plupart des problèmes de type combinatoire, elle n'a pu prendre sa forme actuelle que sous l'impulsion de nombreux spécialistes de la recherche opérationnelle motivée par des problèmes concrets. Les graphes pondérés sont très utilisés pour résoudre certaines situations concrètes, il suffit de les modéliser sous forme d'un graphe valué, puis les résoudre à l'aide d'un algorithme qui correspond au problème posé, parmi ces algorithmes on a l'algorithme de Dijkstra, Prim, kruscal. . . Dans notre travail, nous allons présenter certaines méthodes d'optimisation qu'on va utiliser pour résoudre certains problèmes réels.

Le but de notre travail est de résoudre l'un des problèmes réel en utilisant la théorie des graphes et plus particulièrement l'optimisation dans les réseaux et pour cela, on a réparti notre mémoire en trois chapitres, une introduction et une conclusion.
-Dans le premier chapitre, nous décrivons quelques généralités et notions de base de la théorie des graphes.
- Dans le deuxième chapitre, nous présentons les méthodes de résolution de quelques problèmes d'optimisation dans les réseaux.

-Dans le dernier chapitre, nous allons présenter quelques situations concrètes qu'on va résoudre en utilisant les algorithmes cités dans le chapitre précédent, d'où notre application qui consiste à résoudre un problème de transport en utilisant la théorie des graphes et plus particulièrement l'optimisation dans les graphes pondérés. L'utilisateur qui va bénéficier de ce service disponible dans notre application va pouvoir trouver la solution optimale de ce problème dont l'objectif consiste à trouver la quantité transporté de chaque origine (ou usine) vers les destinations (ou clients), de telle manière que le coût totale de transport soit minimal et que la quantité transporté soit maximal.

- A la fin de notre travail, nous donnons une conclusion générale.

Chapitre 1

Définitions de bases et notations

Les graphes permettent de modéliser toute situation dans laquelle il y a des interactions entre les objets. Les techniques utilisées en théorie des graphes permettent de répondre à beaucoup de problèmes algorithmique posés.
L'objet de ce chapitre est de présenter brièvement, certaines définitions de bases ou concepts généraux sur les éléments de la théorie des graphes que nous allons utiliser par la suite.

1.1 Graphe

Définition 1.1.1. On appelle graphe $G = (X, A)$ la donnée d'un ensemble X dont les éléments sont appelés sommets et d'une partie de A symétrique $(x, y) \in A \Leftrightarrow$ (y , x) $\in A$ dont les éléments sont appelés arêtes.
En présence d'une arête $a = (x, y)$ qui peut être notée simplement xy , on dit que x et y sont les extrémités de a , que a est incidente en x et en y , et que y est un successeur ou voisin de x (et vice versa).
On dit que un graphe est sans boucle si A ne contient pas d'arête de la forme (x, x), c'est-à-dire joignant un sommet à lui même.
Le nombre de sommet est appelé ordre du graphe.
Un graphe ne possédant pas de boucle ni d'arêtes parallèles (deux arêtes distinctes joignant la même paire de sommets) est appelé graphe simple.
On appelle degré sortant ou demi-degré-extérieur d'un sommet x le nombre d'arcs de la forme $a = (x, y)$ avec $y \neq x$, c'est-à-dire le nombre d'éléments de $\Gamma(x)^+ \setminus \{x\}$.
On note $d^+(x)$ ce degré.
On appelle degré entrant ou demi-degré-intérieur d'un sommet x le nombre d'arcs de la forme $a = (x, y)$ avec $y \neq x$, c'est-à-dire le nombre d'éléments de $\Gamma(x)^- \setminus \{x\}$.
On note $d^-(x)$ ce degré.
On appelle degré de x la somme du degré entrant et du degré sortant $(d(x) = d^-(x) + d^+(x))$.
Un sommet de degré entrant non nul et de degré sortant nul est appelé puit, tandis qu'un sommet de degré entrant nul et de degré sortant non nul est appelé source.

Un sommet n'ayant pas d'arcs incidents est appelé sommet isolé, ces sommets ont un degré nul [1].
La Figure1.1 montre un graphe à 4 sommets et 5 arêtes.

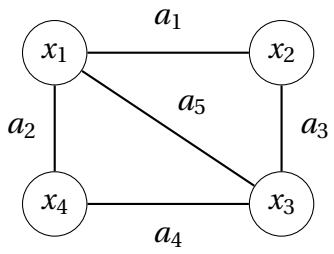


FIGURE 1.1 – Graphe à 4 sommets et 5 arêtes

On distingue deux types de graphes :orienté et non orienté .

1.1.1 Graphe orienté

On dit que G est un graphe orienté s'il y a une distinction entre les liens (x, y) et (y, x), c'est-à-dire $(x, y) \neq (y, x)$. Dans ce cas le lien est appelé un arc. On représente $a = (x, y)$ graphiquement par une flèche qui part de x pour joindre y qui sera la pointe de cette flèche. Dans ce cas, y sera appelé un successeur de x (ou x est un prédécesseur de y) et chaque sommet peut avoir plusieurs successeurs et plusieurs prédécesseurs. On appelle une boucle tout arc (x, x) dont ses extrémités se coïncident. La Figure1.2 montre un graphe orienté à 4 sommets et 5 arcs.

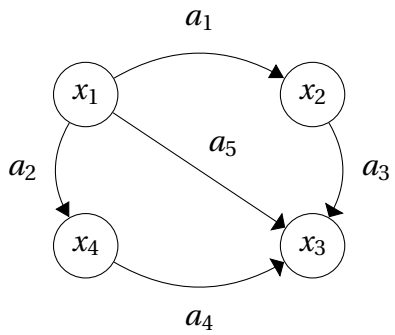


FIGURE 1.2 – Graphe orienté à 4 sommets et 5 arcs

1.1.2 Graphe non orienté

On dit que G est un graphe non orienté, si la précision de sens de lien (x, y) et la distinction entre extrémité initiale et extrémité terminale ne jouent aucun rôle. On appelle tout élément

$(x, y) \in A$ une arête, qui est représentée graphiquement par un segment sans flèche liant les deux nœuds x et y.
La Figure1.3 montre un graphe non orienté à 4 sommets et 5 arêtes.

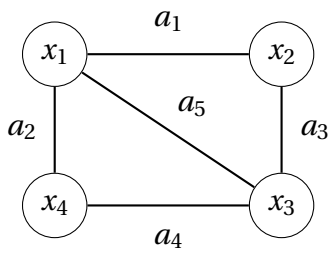


FIGURE 1.3 – Graphe non orienté à 4 sommets et 5 arêtes

1.1.3 Sous-graphe, graphe partiel

Pour un graphe $G = (X, A)$

• Un sous-graphe de G est un graphe $H = (X', A(X'))$ tel que X' est un sous-ensemble de X, et $A(X')$ sont les arcs induits par A sur X', c'est-à-dire les arcs de A dont les 2 extrémités sont des sommets de X'.
$A\ (X') = \{(i,j) \in A \mid i\ ,\ j \in X'\}$

• Un graphe partiel de G est un graphe $I = (X, F)$ tel que F est un sous ensemble de A [2].

Remarque

Un sous-graphe H de G est entièrement défini (induit) par ses sommets X', et un graphe partiel I par ses arcs F.
Étant donné un graphe $G = (X, A)$ et $X' \subset X$, $F \subset A$, le sous-graphe partiel engendré par X' et F est le graphe partiel de H engendré par F. La Figure 1.4(G) montre un Graphe,1.4(G') montre un sous graphe et 1.4(G") montre un graphe partiel.

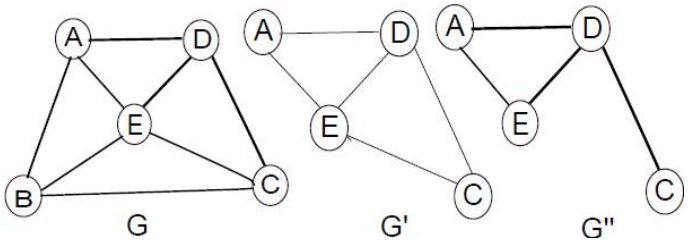


FIGURE 1.4 – Graphe, sous graphe et graphe partiel

Le graphe complémentaire $\overline{G}$ de G est le graphe dont l'ensemble des nœuds est X et l'ensemble des arêtes est $X \setminus A$. Si on considère le sous-graphe A de G comprenant les arêtes {AB, AE, BC, CD, DF, EF}. Alors le graphe complémentaire de A, noté A', est le graphe comprenant les sommets {A, B, C, D, E, F} et l'ensemble d'arêtes {AC, AD, AF, BD, BE, BF, CE, CF, DE}.

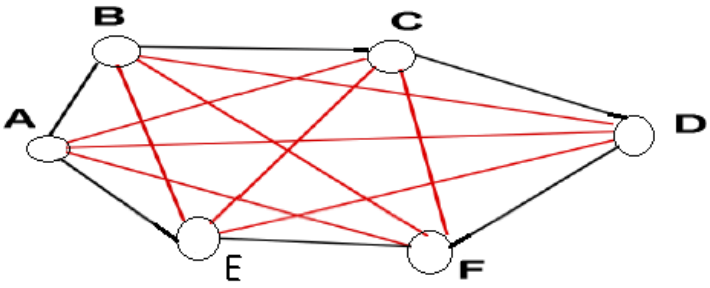


FIGURE 1.5 – Graphe complémentaire

1.2 Chaînes, cycles, chemins et circuits

1.2.1 Chaîne :

Une chaîne est une séquence finie et alternée de sommets et d'arêtes, débutant et finissant par des sommets, telle que chaque arête est incidente avec les sommets qui l'encadre dans la séquence.
Le premier et le dernier sommet sont appelés (sommets) extrémités de la chaîne.
La longueur de la chaîne est égale au nombre d'arêtes qui la composent.

- Si aucun des sommets composant la séquence n'apparaît plus d'une fois, la chaîne est dite **élémentaire**.
- Si aucun des arêtes composant la séquence n'apparaît plus d'une fois, la chaîne est dite **simple**.
- Une chaîne **eulérienne** est une chaîne empruntant une et une seule fois chaque arête de G.

- On appelle chaîne **hamiltonienne** une chaîne passant une et une seule fois par chacun des sommets de G.

1.2.2 Chemin :

Un chemin est une séquence finie et alternée de sommets et d'arcs, débutant et finissant par des sommets, telle que chaque arc est sortant d'un sommet et incident au sommet suivant dans la séquence (cela correspond à la notion de chaîne ≪ orientée ≫).

- Si aucun des sommets composant la séquence n'apparaît plus d'une fois, le chemin est dit élémentaire.
- Si aucune des arêtes composant la séquence n'apparaît plus d'une fois, le chemin est dit simple.
- Un chemin **eulérien** est une chaîne empruntant une et une seule fois chaque arête de G
- On appelle chemin **hamiltonien** un chemin passant une et une seule fois par chacun des sommets de G

1.2.3 Cycle :

Un cycle est une chaîne dont les extrémités coïncident.

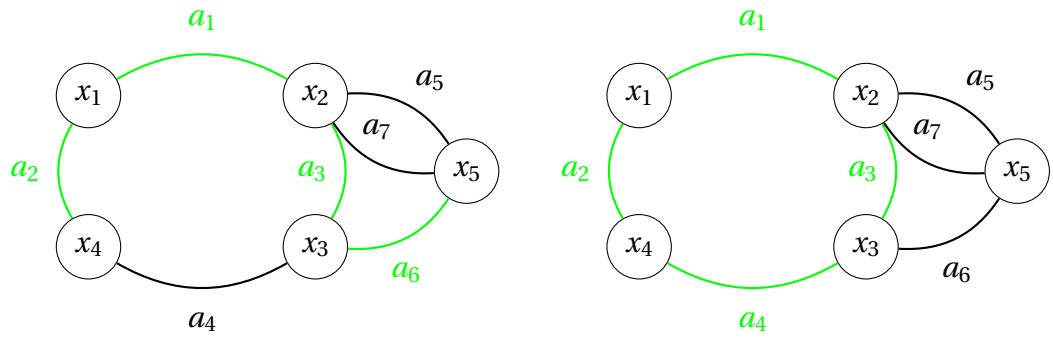
- Un cycle **élémentaire** (tel que l'on ne rencontre pas deux fois le même sommet en le parcourant) est un cycle minimal pour l'inclusion, c'est-à-dire ne contenant aucun autre cycle.
- Un cycle **eulérien** est une chaîne eulérienne dont les extrémités coïncident.
- Un cycle **hamiltonien** est un cycle qui passe une et une seule fois, par chacun des sommets de G.

1.2.4 Circuit :

C'est un chemin dont les extrémités coïncident.

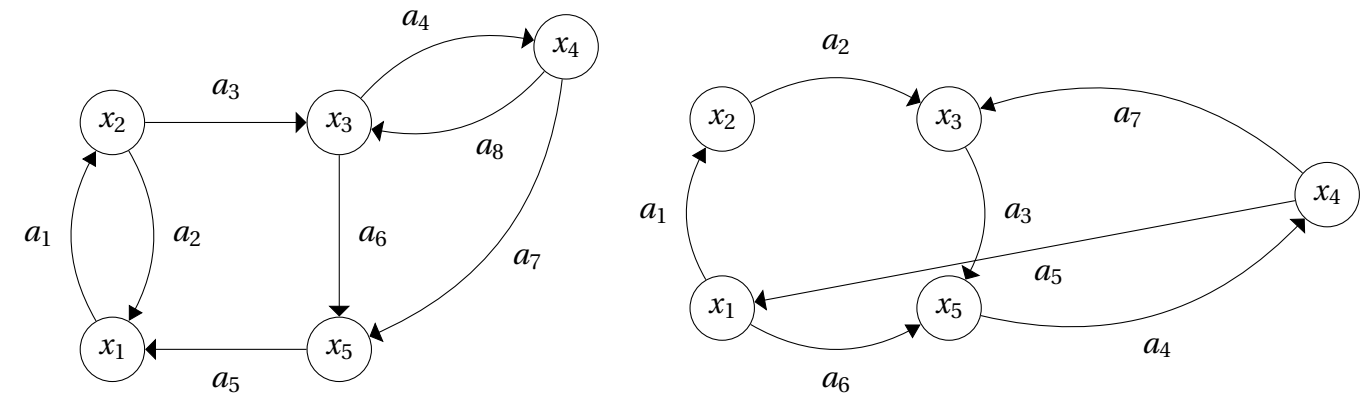
- Un circuit **élémentaire**, on ne rencontre pas deux fois le même sommet.
- Un circuit **eulérien** est un chemin eulérien dont les extrémités coïncident.
- Un circuit **hamiltonien** est un circuit qui passe une et une seule fois par chacun des sommets de G.

La Figure 1.6(G) montre une chaîne et 1.6(G') montre un cycle .



G G′

FIGURE 1.6 – Chaîne et Cycle



G G′

FIGURE 1.7 – Graphe orienté

Dans la Figure1.7 (G) le chemin eulérien est $\{a_1, a_3, a_4\}$, et le circuit eulérien est $\{a_1, a_3, a_6, a_5\}$
A partir de la Figure1.7 (G′) H=$\{x_1, x_2, x_3, x_5, x_4\}$ est un chemin **Hamiltonien**
C=$\{x_1, x_2, x_3, x_5, x_4, x_1\}$ est un circuit **Hamiltonien**

1.3 Connexité et forte connexité

On définit la connexité dans un graphe par la relation entre deux sommets de la manière suivante : deux sommets x et y où une relation de connexité si et seulement s'il existe une chaîne entre x et y ou bien $x = y$ [3].

1.3.1 Composantes connexes :

On appelle composante connexe un ensemble de sommets qui ont deux à deux la relation de connexité, de plus tout sommet en dehors de la composante n'a pas de relation de connexité avec cette composante.

La Figure 1.8 montre un graphe avec 2 composantes connexes.

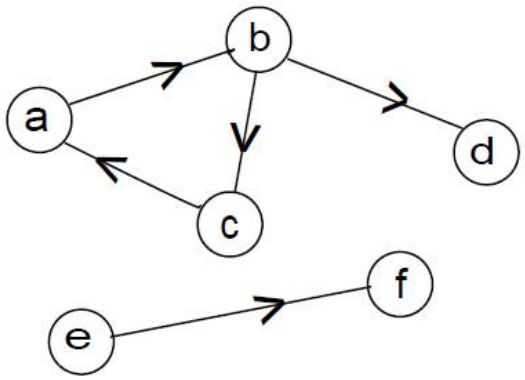
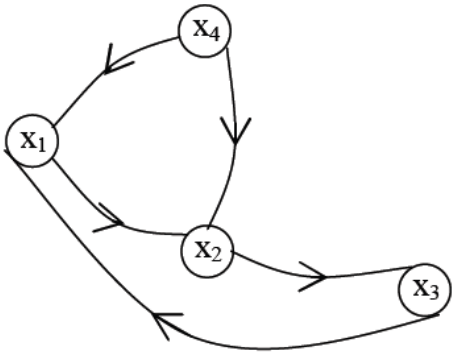


FIGURE 1.8 – Graphe avec 2 composantes connexes

Les sommets a , b , c , d ont deux à deux la relation de connexité, donc l'ensemble $\{a, b, c, d\}$ forme ainsi la $1^{ère}$ composante connexe, on la note C_1.
- L'ensemble $\{e, f\}$ forme la $2^{ème}$ composante connexe, on la note C_2.
On constate que les sommets de C_1 n'ont pas de relation de connexité avec les sommets de C_2
-Un graphe qui contient une seule composante connexe est appelé graphe connexe .

1.3.2 Forte connexité

On définit la forte connexité dans un graphe par une relation entre deux sommets de la manière suivante :
deux sommets x et y ont une relation de forte connexité $\Leftrightarrow$ il existe un chemin de x vers y et un chemin de y vers x ou bien $x = y$.
On considère La Figure suivante :

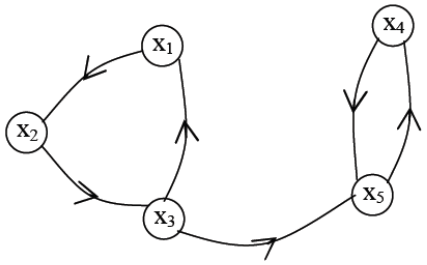


– On a un chemin reliant le sommet x_1 et x_3 et un chemin reliant le sommet x_3 au sommet x_1 alors x_1 et x_3 ont une relation de forte connexité.

– On a un chemin reliant le sommet x_4 à x_3 , mais on n'a pas de chemin reliant le sommet x_3 au sommet x_4 alors x_4 et x_3 n'ont pas de relation de forte connexité.

1.3.3 Composantes fortement connexes :

On appelle composante fortement connexe un ensemble de sommets, qui ont deux à deux la relation de forte connexité, de plus tout sommet en dehors de la composante n'a pas de relation de forte connexité avec aucun élément de cette composante [3].
On considère La Figure suivante :



– Les sommets x_1 , x_2 , x_3 ont deux à deux la relation de forte connexité, donc l'ensemble $\{x_1, x_2, x_3\}$ forme ainsi la $1^{ère}$ composante fortement connexe, on la note C_1.
– L'ensemble $\{x_4, x_5\}$ forme la composante fortement connexe, on la note C_2.

On constate que les sommets de C_1 n'ont pas de relation de forte connexité avec les sommets de C_2.

Le graphe fortement connexe :

Un graphe G est dit fortement connexe si tous ses sommets ont deux à deux la relation de forte connexité, autrement dit si G contient une seule composante fortement connexe.

1. Le graphe G précédant contient deux composantes fortement connexes $C_1 = \{x_1, x_2, x_3\}$ et $C_2 = \{x_4, x_5\}$, le graphe n'est pas fortement connexe.
2. Si on ajoute l'arc (x_5, x_3) au graphe précédant, le graphe obtenu contient une seule composante fortement connexe $C = \{x_1, x_2, x_3, x_4, x_5\}$, alors il est fortement connexe.

1.3.4 Couplage

Un couplage d'un graphe $G = (X, A)$ est un ensemble des sommets M d'arêtes deux à deux non adjacentes. Un sommet de graphe est dit saturé, s'il est l'extrémité d'une arête de M , il est dit insaturé sinon, si tout sommets de G est saturé alors le couplage est dit parfait.

Un couplage est dit maximum s'il a plus grand nombre possible d'arêtes.

Le couplage maximum est le couplage couvrant le plus grand nombre de sommets possibles, en laissant donc le moins des sommets isolés [3].

La Figure 1.9 représente un couplage parfait .

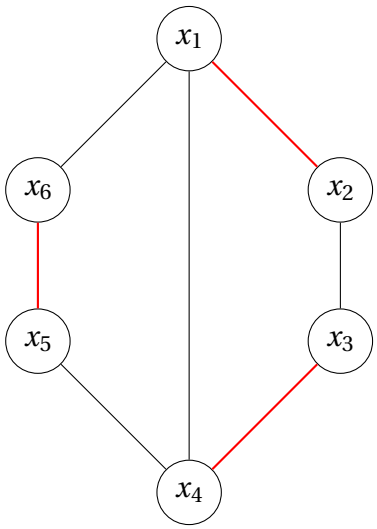


FIGURE 1.9 – Couplage parfait

1.3.5 Transversal

Un transversal d'un graphe est un sous-ensemble de sommet T tel que toute arête du graphe est incidente à au-moins un sommet de T.
Un transversal est minimum s'il a le plus petit nombre possible de sommets.
Le nombre transversal $\tau(G)$ d'un graphe G est le cardinal d'un transversal minimum de G.

1.3.6 Stable

Soit $G = (X, A)$ un graphe, un ensemble S de X est un stable s'il ne comprend que des sommets non adjacents deux à deux. Le cardinal de la plus grande partie est le nombre de stabilité de G, on le note $\alpha(G)$.
L'ensemble des sommets en oranges dans la Figure 1.10 est un stable maximal du graphe.

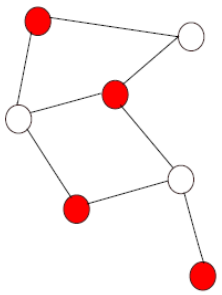


FIGURE 1.10 – Stable

1.3.7 Clique

Une clique dans un graphe est un ensemble de sommets tous relié deux à deux. $W(G)$: la taille de la plus grande clique dans un graphe G.
La Figure 1.11 représente 3-clique $\{x_1, x_2, x_3\}$.

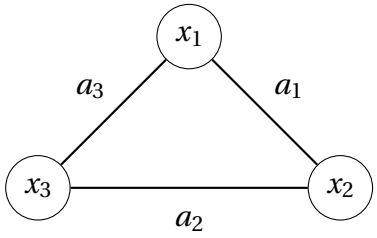


FIGURE 1.11 – Clique

1.3.8 Distances dans les graphes

Étant donné deux sommets x et y d'un graphe $G = (X(\mathrm{G}), A(\mathrm{G}))$, on appelle distance entre x et y , la longueur d'une plus courte (x, y) -chaîne (en terme de nombre d'arêtes) qui les relies et on la note $d_G(x, y)$ (ou $d(x, y)$ s'il n'y a pas de confusion). Une telle chaîne s'appelle géodésique.

- **L'excentricité**

Soit un graphe simple non orienté $G(X, A)$; $\forall\ x \in X$ l'excentricité de x est :
$e(a) = \mathrm{M}ax\{d(x, y) : x, y \in \mathrm{X}\}$
Autrement dit :

▷ l'excentricité de x désigne la distance qui sépare x du sommet le plus éloigné de x dans G;

▷ l'excentricité d'un sommet est la distance maximum de ce sommet aux autres sommets.

- **Centre**

On appelle centre d'un graphe, le sommet d'écartement minimal, (le centre n'est pas nécessairement unique). Par exemple, dans une clique, tous les sommets sont "centrés"

- **Diamètre**

Le diamètre d'un graphe est la plus grande distance entre deux sommets quelconques de ce graphe. Le diamètre de G est : $D = \mathbf{M}ax\{e(x) : x \in X\}$

- **Rayon**

On appelle rayon d'un graphe G, noté $\rho(G)$, l'écartement d'un centre de G. Autrement dit : le rayon d'un graphe est le minimum des excentricités des différents sommets.

1.4 Quelques graphes particuliers

Dans cette section, nous introduisons quelques graphes particuliers utilisés en théorie des graphes de manière courante.

1.4.1 Graphe complet

Un graphe G est dit complet si chaque sommet est relié avec tous les autres sommets. Le graphe complet d'ordre n est noté K_n, ou chaque sommet est de degré $n-1$

La Figure 1.12 représente un graphe complet .

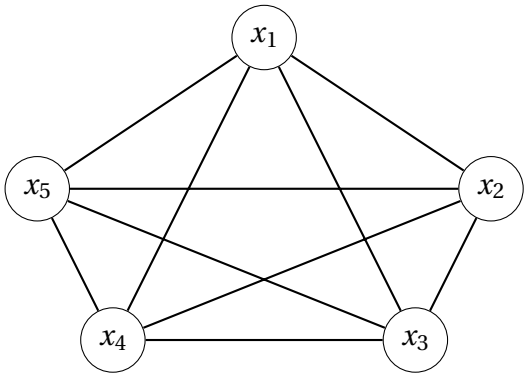


FIGURE 1.12 – Graphe Complet

1.4.2 Graphe biparti

Un graphe est dit biparti si ses sommets peuvent être divisés en deux classes X et Y en sorte que deux sommets de la même classe ne sont jamais reliés. les arêtes de ce graphe sont celle qui relient un sommet de X à un sommet de Y.

La Figure 1.13 représente un graphe biparti .

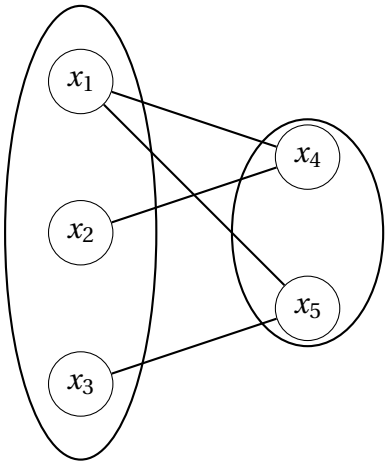


FIGURE 1.13 – Graphe biparti

1.4.3 Graphe biparti complet

Un graphe biparti est dit biparti complet (ou encore est appelé une biclique) si chaque sommet de X est relié à chaque sommet de Y. La Figure 1.14 représente un graphe biparti complet .

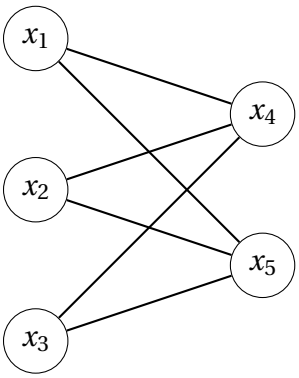


FIGURE 1.14 – Graphe biparti complet

1.4.4 Graphe biparti équilibré

Un graphe biparti G est équilibré si chaque ensemble de la bipartition est de même taille. Les graphes ayant des cycles de longueur paire, c'est-à-dire les graphes bipartis complets, pour lesquels $|X_1|=|X_2|$, sont des ensembles de graphes équilibrés. La Figure 1.15 montre un graphe biparti équilibré

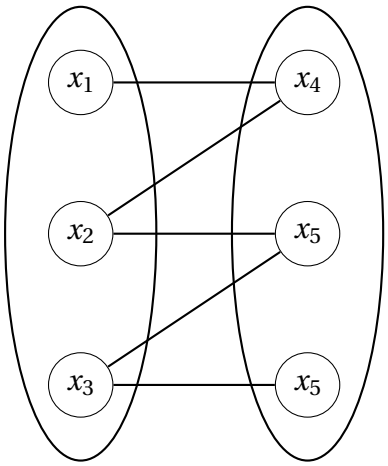


FIGURE 1.15 – Graphe biparti équilibré

1.4.5 Graphe planaire

Un graphe G est dit planaire si on peut le dessiner sur un plan de telle façon que les arêtes ne se coupent pas, en dehors de leurs extrémités [3].
La Figure 1.16 représente un graphe planaire .

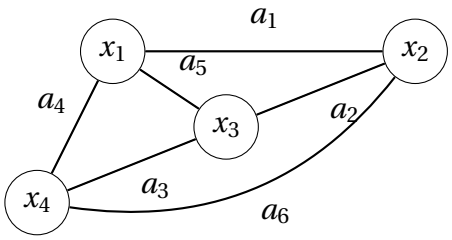


FIGURE 1.16 – Graphe planaire

1.4.6 Arbre

On dit qu'un graphe est un arbre si et seulement s'il est connexe et n'admet pas un cycle.
Étant donné un graphe non orienté $G = (X, A)$ d'ordre n, les affirmations suivantes sont équivalentes :

1. G est un arbre.
2. G est sans cycles et connexe.
3. G est sans cycles et comporte n-1 arêtes.
4. G est connexe et comporte n-1 arêtes.
5. G est sans cycle, et quand on ajoute une arête il devient cycline élémentaire.
6. G est connexe, et la suppression d'une arête le déconnecte.

La Figure 1.17 représente un arbre.

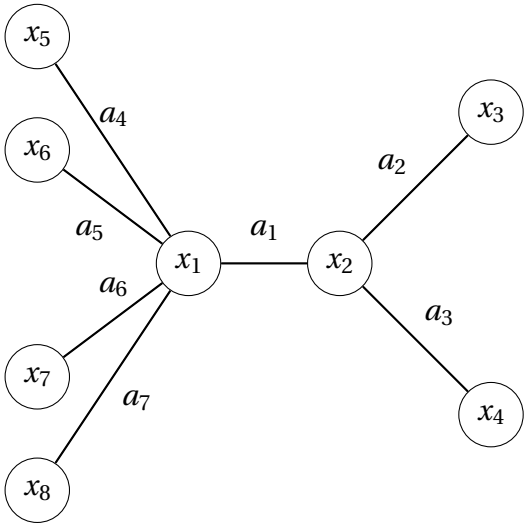


FIGURE 1.17 – Arbre

Remarque :

Un graphe sans cycle et non connexe est appelé une forêt.

1.4.7 Arborescence :

C'est un graphe orienté ou chaque sommet possède un seul précédent sauf un qui n'en a pas la racine.$\forall\ x \in X$, $\exists$ un chemin unique de la racine à x. On considère un nœud x d'une arborescence T, de racine r. Un nœud y quelconque sur le chemin unique de r à x est appelé ancêtre de x; x est un descendant de y.

Si le dernier arc sur le chemin de r vers x est (y, x), alors y est le père de x, x est un fils de y. Si deux nœuds ont le même père, ils sont frères. Un nœud sans fils est une feuille. La longueur du chemin entre r et x est la profondeur de x dans T. La plus grande profondeur de T est la hauteur de T. Si chaque nœud a au maximum deux fils, on parle d'arborescence binaire. La Figure 1.18 représente une Arborescence .

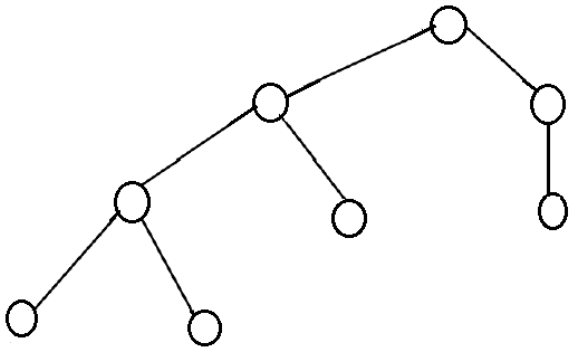


FIGURE 1.18 – Arborescence

1.5 Représentation matricielle des graphes

1.5.1 Matrice d'adjacence

Utilisation des outils d'algèbre linéaire.

Définition 1.5.1. Considérons un graphe $G = (X, A)$ comportant n sommets. La matrice d'adjacence de G est égale à la matrice $U = (u_{ij})$ de dimension $n \times n$ telle que :

$$u_{ij} = \begin{cases} 1, si (i,j) \in A \text{ (c'est-à-dire } (i,j) \text{ est une aréte)} ; \\ 0, \text{sinon.} \end{cases}$$

Une telle matrice, ne contenant que des $\ll 0 \gg$ et des $\ll 1 \gg$ est appelée matrice booléenne.

On considère le graphe de la Figure 1.19 :

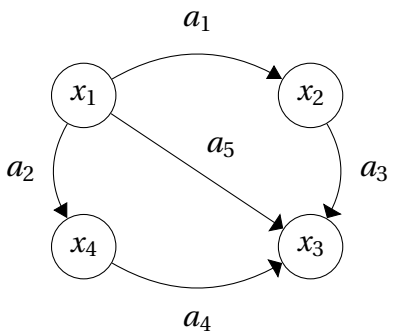


FIGURE 1.19 – Graphe orienté

La matrice d'adjacence associer au graphe de la Figure 1.19, est la suivante :

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

• **Propriétés**

- La somme des éléments de la $i^{éme}$ ligne de U est égale au degré sortant $d_s(x_i)$ du sommet x_i de G.
- La somme des éléments de la $j^{éme}$ colonne de U est égale entrant $d_e(x_j)$ du sommet x_j de G.
U est symétrique si, et seulement si, le graphe G est symétrique.

1.5.2 Matrice d'incidence :

Incidence entre arêtes et sommets

Définition 1.5.2. Considérons un graphe orienté sans boucle $G=(X,A)$ comportant n sommets $x_1,...,x_n$ et m arêtes $a_1,...,a_m$. On appelle matrice d'incidence(aux arcs) de G la matrice $M=(m_{ij}$ de dimension $n\times m$ telle que :

$$m_{ij} = \begin{cases} 1 \ \textit{si} \ x_i \text{ est l'extrémité initiale de } a_j \\ \text{-1 } \textit{si} \ x_i \text{ est l'extrémité terminale de } a_j \\ 0 \ \textit{si} \ x_i \text{ n'est pas une extrémité de } a_j \end{cases}$$

• Pour un graphe non orienté sans boucles, la matrice d'adjacence (aux arêtes) est définie par :

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

La matrice d'incidence du graphe de la Figure 1.19, est la suivante :

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

La Figure 1.20 montre un graphe à quatre sommets

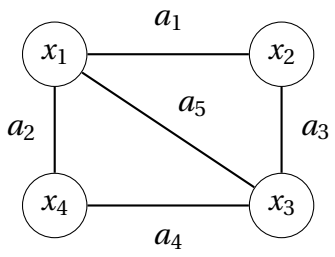


FIGURE 1.20 – Graphe non-orienté

La matrice d'adjacence du graphe de la Figure 1.20, est la suivante :

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

1.6 Opérations sur les graphes

1.6.1 Subdivision de graphe

En remplaçant les arêtes d'un graphe $G = (X, A)$ par des chaînes disjointes intérieurement , on obtient une subdivision de G.
Une subdivision d'une arête $u \rightarrow v$ s'obtient en ajoutant au graphe de départ de nouveaux points $(x_1, x_2, ..., x_n)$ et en remplaçant l'arête $u \rightarrow v$ par un chemin $(u \rightarrow x_1 \rightarrow ... \rightarrow x_n \rightarrow v)$ (la subdivision triviale obtenue pour $n = 0$ consiste à ne rien changer). Un graphe G' est une subdivision d'un graphe G si on l'obtient par subdivision distinctes de ses arêtes.
La Figure 1.21 représente une subdivision d'un graphe .

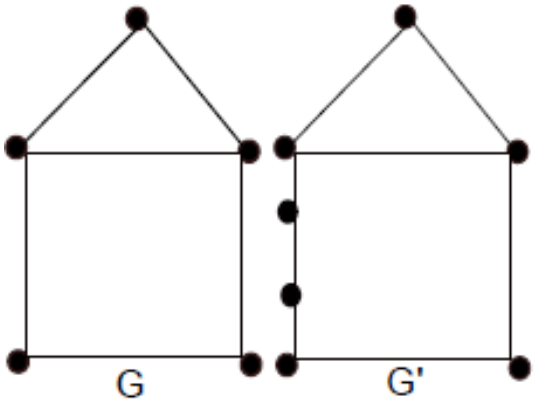


FIGURE 1.21 – Subdivision de graphe

1.6.2 Somme Cartésienne de deux graphes

On appelle somme cartésienne de deux graphes $G = (X(G), A(G))$ et $H = (X(H), A(H))$, notée $G \bigoplus H$, le graphe dont l'ensemble des sommets est le produit cartésien $X(G) \times X(H)$ et où deux sommets (x, x') et (y, y') sont adjacents si et seulement si l'une des propriétés suivantes est vérifiée :

- $x = y$ et $x'y' \in A(H)$
- $xy \in A(G)$ et $x' = y'$.

La Figure 1.22 montre la somme cartésienne de deux graphes :

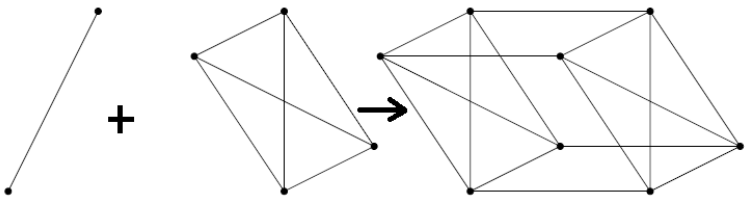


FIGURE 1.22 – Somme cartésienne de deux graphes

On note que le nombre de sommets dans $G \times H$ est $|X(G)|.|X(H)|$, et que le nombre d'arêtes est :
$|X(G)|.|A(H)| + |X(H)|.|A(G)|$.

1.6.3 Produit Cartésien de deux graphes

Le produit cartésien de deux graphes $G = (X(G), A(G))$ et $H = (X(H), A(H))$ est le graphe noté $G \times H$ où $X(G \times H) = X(G) * X(H)$. Deux sommets (x, y) et (x', y') sont adjacents si et seulement si $xx' \in A(G)$ et $yy' \in A(H)$.

La Figure 1.23 montre le produit cartésien de deux graphes.

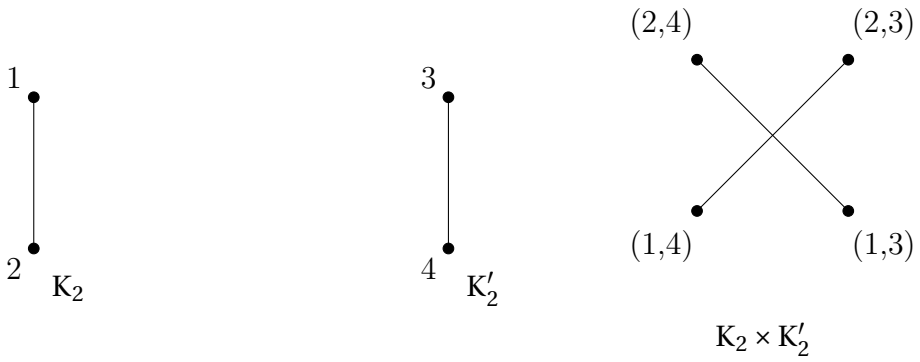


FIGURE 1.23 – Produit cartésien de deux graphes

1.6.4 Morphisme de graphe

• **Graphes orientés**

Soit G = (X, A) et H = (T, B) deux graphes orientés. On dit qu'une Application ϕ de X dans T réalise un morphisme du graphe G vers le graphe H si elle vérifie la propriété suivante :

$((x, y) \in A) \Rightarrow (\phi(x), \phi(y)) \in B$ Si ϕ est une application injective, on dit que le graphe G s'injecte dans le graphe H. On dit aussi parfois que le graphe H contient le graphe G.

• **Graphes non-orientés**

Soit $G = (X, A)$ et $H = (T, B)$ deux graphes non-orientés. On dit qu'une Application ϕ de X dans T réalise un morphisme du graphe G vers le graphe H si elle vérifie la propriété suivante : $(y \in A) \Rightarrow \phi(y) \in B$

Si ϕ est une application injective, on dit que le graphe G s'injecte dans le graphe H. On dit aussi parfois que le graphe H contient le graphe G [5].

1.6.5 Homomorphisme de graphe

Soient $G = (X, A)$ et $G' = (X', A')$ deux graphes. Une application $f : X \to X'$ tel que : est un homomorphisme de G dans G' Si : $(x, y) \in A \Rightarrow (f(x), f(y)) \in A'$

On considère la Figure 1.24 , avec les graphes G et G' on voit facilement qu'on a un homomorphisme de G dans G' mais pas de G' dans G.

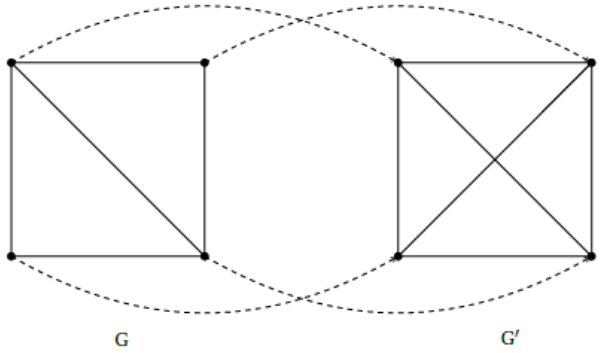


FIGURE 1.24 – Homomorphisme de graphe

1.6.6 Isomorphisme de graphe

Soient $G=(X,A)$ et $G^{'}=(X,A^{'})$ deux graphes. S'il existe une relation univoque $m: X \rightarrow X^{'}$ tel que $(x_1,x_2) \in A \Leftrightarrow (m(x_1),m(x_2)) \in A^{'}$, alors G et $G^{'}$ sont isomorphes [7]. La Figure 1.25 montre que G et $G^{'}$ sont isomorphes.

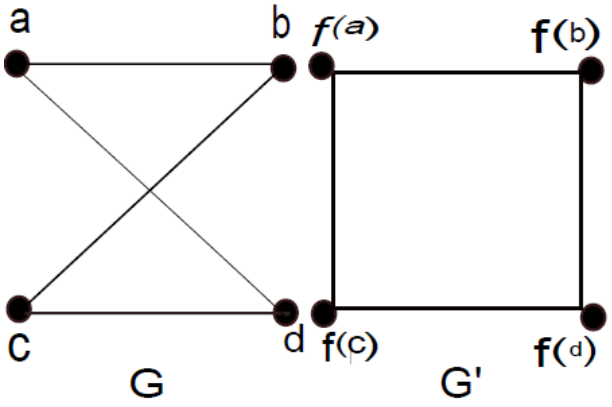


FIGURE 1.25 – Isomorphisme de graphe

Conclusion :

Dans ce chapitre, on s'est intéressé aux quelques notions de la théorie des graphes en donnant quelques définitions et concepts de base sur les graphes qu'on va utiliser dans le chapitre suivant.

Chapitre 2

Quelques algorithmes d'optimisation

La théorie des graphes a été largement étudiée et de nombreux problèmes de la vie réelle ont été résolus grâce à elle, beaucoup de ces problèmes peuvent être modéliser en utilisant des graphes valués parmi eux les problèmes d'optimisation dans les réseaux qu'on va étudier dans ce chapitre.

2.1 Définition d'un réseau :

Un réseau est un graphe valué, fortement connexe, sans boucle et ayant plus d'un sommet, noté $R = (X, A, C)$ dans lequel il y a un unique sommet S appelé la source tel que $d^-(s) = 0$ et un unique sommet P appelé puit tel que $d^+(s) = 0$, pour $a \in A$ le nombre $C(a)$ est appelé capacité de l'arc a.
On appelle noeud d'un réseau un sommet qui a plus de deux arcs incidents. Les autres sommets sont appelés antinoeuds.
On appelle branche tout chemin pour lequel seuls les premiers et derniers sommets sont des noeuds.

La valuation des arêtes peut être :
a) Une lettre, un mot, un nombre, un symbole (la valuation est donné par une étiquète) on parle d'un graphe étiqueté.

2.1.1 Graphe étiqueté :

Un graphe étiqueté est un graphe où chacune des arêtes est affectée d'une lettre, d'un mot, d'un nombre ou d'un symbole. Ces symboles sont appelés étiquettes [4]. La Figure 2.1 représente un graphe étiquetté.

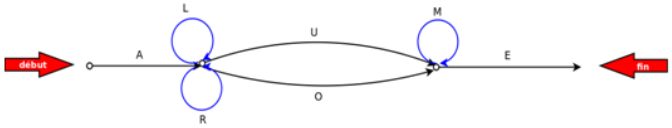


FIGURE 2.1 – Graphe étiqueté

b) La valuation est positive dans ce cas on parle d'un graphe pondéré (valué).

2.1.2 Graphe pondéré :

On dit un graphe est pondéré si on affecte à chaque arête un nombre positif (quelque soit sa signification).

- Ce nombre positif est alors appelé poids de l'arête.
- Le poids d'une chaîne est la somme des poids des arêtes qui la compose.
- La plus courte chaîne entre deux sommets est la chaîne de poids minimal entre ces deux sommets [4].

La Figure 2.2 représente un graphe pondéré.

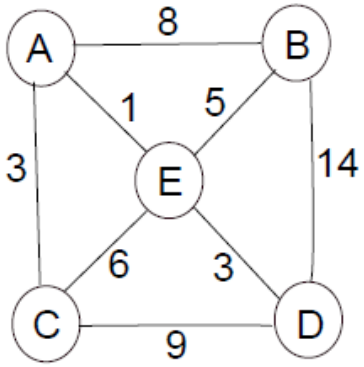


FIGURE 2.2 – Graphe pondéré

Remarque : Un graphe pondéré est donc un cas particulier de graphe étiqueté.

c) lorsque la valuation est un nombre entre 0 et 1 on parle d'un graphe probabiliste.

2.1.3 Graphe probabiliste :

Un graphe probabiliste est un graphe orienté et pondéré dans lequel :

- Il y a au plus un arc d'un sommet à l'autre.
- La somme des poids des arcs issus d'un même sommet est égale à 1 [4].

Remarques :

1. Les poids des arcs sont alors des probabilités (nombres réels compris entre 0 et 1).
2. Un graphe probabiliste indique les différents états possibles d'un système (sommets du graphe) et les probabilités de passage d'un état à l'autre (poids des arcs).

La Figure 2.3 représente un graphe probabiliste.

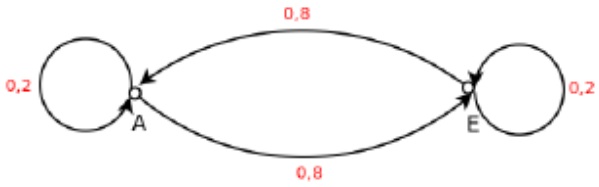


FIGURE 2.3 – Graphe probabiliste

2.2 Quelques problèmes d'optimisation dans les réseaux :

Dans cette section nous allons présenter quelques problèmes connus dans les réseaux pour lesquelles nous allons présenter les techniques d'optimisations en citant le problème de plus court chemin, le problème de l'arbre couvrant de poids minimum, recherche des points d'articulation, problème d'affectation, problème d'ordonnancement, problème de transport.

2.2.1 Problème du plus court chemin

Les problèmes de cheminement sont des problèmes classiques de la théorie des graphes. L'objectif est de calculer une route entre des sommets d'un graphe qui minimise ou maximise une certaine fonction économique.

Le problème le plus classique consiste à chercher le chemin qui minimise la somme des valuations des arêtes traversées. Il existe des algorithmes de complexité en temps polynomiale pour le résoudre, comme l'algorithme de Dijkstra, Bellman, Ford...

On peut s'intéresser à la recherche d'un plus court chemin dans un graphe :

P1 - d'un sommet à tous les autres.

P2 - entre tous les couples de sommets.

P3 - entre deux sommets donnés.

Notons qu'il n'y a pas de solution spécifique au problème consistant à chercher un plus court chemin entre deux sommets x et y particuliers d'un graphe, il se résout en résolvant (P1) : partant de x on s'arrêtant en y.

Les algorithmes étudiés ici sont ceux de DIJKSTRA et de BELLMAN − FORD qui résolvent (P1), et l'algorithme de DANTZIG et Maria Hasse qui résolvent (P2).
L'algorithme de DIJKSTRA est sans doute le plus utilisé car il est aisé à mettre en oeuvre, efficace en temps d'exécution et bien adapté aux situations courantes, c'est pourquoi nous en donnerons une écriture détaillée.
Nous détaillerons aussi l'algorithme de DANTZIG et MARIA HASSE car il sont matriciels, donc facile à mettre en oeuvre, et plus efficaces que l'algorithme de BELLMAN . FORD appliqué à tous les sommets lorsque v est quelconque.

(a) Plus court chemin (PCC) d'un sommet à tous les autres :

Soit R=(X, A, C) un réseau et s un sommet particulier du graphe , où $\| X = n \|$ et $\| A = m \|$ et C une application définie comme suit :

$C : A \rightarrow R^+$

$(i, j) \rightarrow C_{ij}$

(1) Algorithme de DIJKSTRA

L'algorithme de DIJKSTRA (1959) résout (P1) lorsque $C_{ij} \geq 0$ dans ce cas il nous donne une solution optimale. on note :

λ_j : la longeur du plus court chemin entre les sommets s et j .

S : l'ensemble des sommets marqués .

$\bar{S}$: l'ensemble des sommets non marqués tel que $\bar{S} = X/\{S\}$

(1) Algorithme de DIJKSTRA

Étape1 : Initialisation

S={s} ,$\lambda_s = 0$, $\lambda_j = \infty$, $j \in \bar{S}$

Étape2 : Calculer le pcc entre s et les autres sommets comme suit :

$$\begin{cases} \lambda_j = C_{sj}, & si \quad j \in \Gamma^+(s) \\ \lambda_j = \infty, & sinon \end{cases}$$

Étape3 : Sélectionner un sommet $k \in \bar{S}$ vérifiant :

$\lambda_k = min \lambda_j$

si $\bar{s} = \{\emptyset\}$ alors

arrêter l'algorithme

sinon

aller à l'étape 4

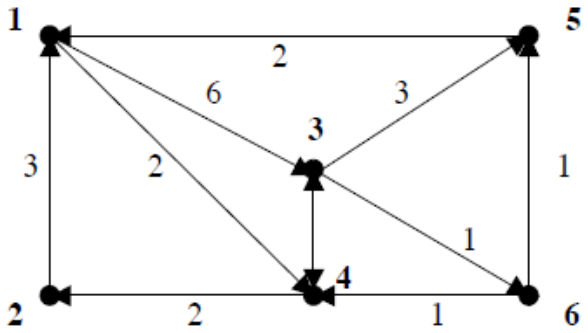
Étape4 : Mettre à jour les distances pour tout $j \in \Gamma^+(k)$ avec $j \in \bar{s}$ faire

$$\lambda_j = min\{\lambda_j, \lambda_k + C_{kj}\}$$

retourner à l'étape 3.

Exemple :

Soit un réseau $R = (X, A, C)$ suivant :



Initialisation :

j	1	2	3	4	5	6
λ_j	0	∞	∞	∞	∞	∞

Itération 1 :$S = \{1\}$,$\bar{S} = \{2,3,4,5,6\}, k = 1, \Gamma^+(1) = \{3,4\}$

j	1	2	3	4	5	6
λ_j	0	∞	6	2	∞	∞

Itération 2 :$S = \{1,4\}, \bar{S} = \{2,3,5,6\}, k = 4$,$\Gamma^+(4) = \{2,3\}$

j	1	2	3	4	5	6
λ_j	0	4	6	2	∞	∞

Itération 3 : $S = \{1,4,2\}$,$\bar{S} = \{3,5,6\}, k = 2, \Gamma^+(2) = \emptyset$

Itération 4 : $S = \{1,4,2,3\}, \bar{S} = \{5,6\}, k = 3, \Gamma^+(3) = \{5,6\}$

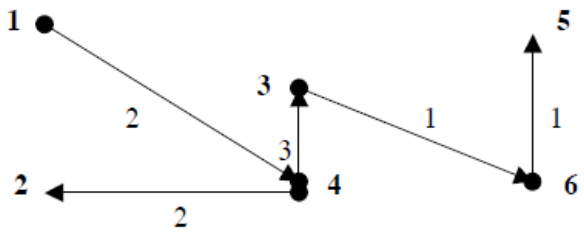
j	1	2	3	4	5	6
λ_j	1	4	6	2	9	7

Itération 5 : $S = \{1,4,2,3,6\}, \bar{S} = \{5\}, k = 6$,$\Gamma^+(6) = \{5\}$

j	1	2	3	4	5	6
λ_j	1	4	6	2	8	7

Itération 6 :$S = \{1,4,2,3,6,5\}, \bar{S} = \{\emptyset\}$ alors l'algorithme s'arrête.

le PCC est : $1-4-2-3-6-5$



Remarques :

Si l'on veut uniquement chercher un PCC de 1 à un sommet particulier x_0 il suffit d'arrêter l'algorithme dès que x_0 devient visité.

La seule différence entre la version orientée vue ici et la version non orientée consiste pour cette dernière à remplacer la recherche des successeurs par celles des voisins.

(2) Algorithme de BELLMAN-FORD :

L'algorithme de BELLMAN-FORD (1956) permet de résoudre (P1) pour des valuations quelconques.

Structures des données :

Nous supposerons ici que le graphe $G = (X, A, V)$ où $X = 1, 2, ..., n$ est donné par sa matrice d'adjacence M, de taille n.n.

Pour le calcul des PCC de 1 aux autres sommets il faut déterminer le tableau $d(1), d(2), ..., d(n)$ des distances ($d(i) = \infty$ si un tel chemin n'existe pas) et le tableau $p(1), p(2), ..., p(n)$ des prédécesseurs défini par : $p(1) = 1$ et p(i) = j $\Leftrightarrow$ j est le prédécesseur de i sur un PCC de 1 à i, s'il existe, 0 sinon. Nous aurons ainsi au plus un PCC par sommet. Le tableau p permet, à la demande et s'il existe, de construire effectivement un PCC de 1 à un sommet donné ainsi que l'arborescence des PCC.

Pour le calcul des PCC entre tous les couples de sommets il faut déterminer la matrice D des distances, $D(i, j)$ = longueur d'un PCC de i à j s'il existe, ∞ sinon , bien sur $D(i, i) = 0$, et la matrice P des prédécesseurs, $P(i, j)$ = prédécesseur immédiat de j sur un PCC de i à j s'il existe, 0 sinon.

Là aussi il y aura au plus un PCC pour chaque couple de sommets comme dans le cas précèdent P permet de construire effectivement un PCC entre tout couple de sommets ainsi que le sous graphe des PCC, si nécessaire.

Remarquons enfin que l'étude faite dans le cadre général des graphes orientés reste bien sur valable pour les graphes non orientés.

Principe :

A partir de $d(1) = 0$ et $d(x) = +\infty, \forall x \neq 1$, on parcourt séquentiellement tous les arcs du graphe :

pour chaque arc xy on teste si $d(y) \leq d(x) + \mathrm{M}(x, y)$ si ce n'est pas le cas d(y) prend la valeur $d(x) + \mathrm{M}(x, y)$, puis on teste l'arc suivant. Si, a l'issue d'un deuxième parcours, d n'est pas modifié, il représente les distances cherchées sinon un parcours supplémentaire est nécessaire et cela jusqú'à stabilisation de d.

Squelette de l'algorithme :

d(1) = 0
p(1) = 1
pour $i = 2$ à n faire
    $d(i) = +\infty$
    $p(i) = 0$
tant que $\exists xy \in \mathrm{A}$ tel que $d(y) > d(x) + \mathrm{M}(x, y)$ faire
    $d(y) = d(x) + \mathrm{M}(x, y)$
    $p(y) = x$

Exemple :

Appliquons l'algorithme sur l'exemple précédent :

On initialise le tableau des distances à : $d(1) = 0$ et $d(2) = ... = d(6) = \infty$ et le tableau des prédécesseurs à $p(1) = 1$ et $p(2) = ... = p(6) = 0$ (0 signifiant ici non déterminé).

L'algorithme consiste à faire plusieurs parcours (faire un parcours du graphe c'est parcourir un à un tous ses arcs) et si, dans un parcours, on a un arc xy vérifiant $d(y) > d(x) + \mathrm{M}(x, y)$ (ce qui montre que le chemin allant de 1 à y en passant par x est plus court le chemin déjà connu de longueur d(y)) alors d(y) devient $d(y) = d(x) + \mathrm{M}(x, y)$ et $p(y) = x$.

L'algorithme ne s'arrête que si, dans un parcours, aucune modification n'est faite.

Pour chaque parcours considérons les arcs dans l'ordre lexicographique (ce n'est pas obligatoire, tout ordre convient, mais c'est plus simple à gérer) c'est-à-dire 13, 14, 21, 34, 35,36, 42, 43, 51, 64 et 65.

Nous ne détaillerons pas toutes les étapes de l'algorithme, examinons tout de même le premier parcours en détail.

13 : $d(3) = \infty > d(1) + \mathrm{M}(1,3) = 0 + 6 = 6$, d'où les modifications $d(3) = 6 \text{et} p(3) = 1$

14 : $d(4) = \infty > d(1) + \mathrm{M}(1,4) = 0 + 2 = 2$, d'où les modifications $d(4) = 2 \text{et} p(4) = 1$

21 : aucune modification puisque $d(1) = 0 \leq d(2) + \mathrm{M}(2,1) = \infty + 3 = \infty$

34 : aucune modification puisque $d(4) = 2 \leq d(3) + \mathrm{M}(3,4) = 6 + 1 = 7$

35 : $d(5) = \infty > d(3) + \mathrm{M}(3,5) = 6 + 3 = 9$, d'où les modifications $d(5) = 9 et p(5) = 3$

36 : $d(6) = \infty > d(3) + \mathrm{M}(3,6) = 6 + 1 = 7$, d'où les modifications $d(6) = 7 et p(6) = 3$

42 : $d((2) = \infty > d(4) + \mathrm{M}(4,2) = 2 + 2 = 4$, d'où les modifications $d(2) = 4 et p(2) = 4$

43 : $d(3) = 6 > d(4) + \mathrm{M}(4,3) = 2 + 3 = 5$, d'où les modifications $d(3) = 5 et p(3) = 4$

51 : aucune modification puisque $d(1) = 0 \leq d(5) + \mathrm{M}(5,1) = 9 + 2 = 11$

64 : aucune modification puisque $d(4) = 2 \leq d(6) + \mathrm{M}(6,4) = 7 + 1 = 8$

65 : $d(5) = 9 > d(6) + \mathrm{M}(6,5) = 7 + 1 = 8$, d'où les modifications $d(5) = 8 et p(5) = 6$
Dans le deuxième parcours (tous les arcs du graphe sont à nouveau passés en revue, toujours dans le même ordre : 13, 14, . . . , 65) les deux seules modifications sont (et dans cet ordre) :
36 : $d((6) = 7 > d(3) + \mathrm{M}(3,6) = 5 + 1 = 6$, d'où la modification $d(6) = 6 \text{et} p(6) = 3$
65 : $d(5) = 8 > d(6) + \mathrm{M}(6,5) = 6 + 1 = 7$, d'où la modification $d(5) = 7 \text{et} p(5) = 6$
Un troisième parcours est donc nécessaire, celui-ci n'apporte rien, l'algorithme est donc terminé.
Les tableaux définitifs des distances de 1 à tous les autres sommets et des prédécesseurs sont donc respectivement :
$d(2) = 4, d(3) = 5,\ d(4) = 2,\ d(5) = 7,\ d(6) = 6$ et $p(2) = 4,\ p(3) = 4,\ p(4) = 1,\ p(5) = 6, p(6) = 3$

On retrouve le graphe partiel des PCC obtenu avec l'algorithme de DIJKSTRA.

Remarque :
Cet exemple met bien en évidence la différence d'efficacité en temps de ces deux algorithmes : l'algorithme de DIJKSTRA est bien plus rapide et clair que celui de BELLMAN - FORD.

(b) Plus court chemin entre tous les sommets

(1) Algorithme de DANTZIG

L'algorithmes de DANTZIG (1966), essentiellement matriciel, permet de résoudre directement (P2) pour une valuation quelconque.

Principe et squelette de l'algorithme :

L'idée est de calculer de proche en proche, les matrices D et P sur des sous graphes de plus en plus gros allant du sous graphe réduit au sommet 1 jusqú'au graphe tout entier.
Pour $k \in \{1, 2, \ldots, n\}$ notons G_k le sous-graphe engendré par les sommets $1, 2, \ldots, k$ et D_k la matrice des distances des PCC entre tous les couples de sommets de G_k.
Il faut donc calculer $\mathrm{D} = \mathrm{D}_n$ tout en actualisant à chaque étape la matrice des prédécesseurs P.

On a $\mathrm{D}_1(1,1) = 0$
Pour k variant de 1 à $n-1$ la construction de la matrice D_{k+1} à partir de la matrice D_k se fait de la façon suivante :
- Pour tout i variant de 1 à k

$$\mathrm{D}_{k+1}(i, k+1) = min \mathrm{D}_k(i,j) + \mathrm{M}(j, k+1) \qquad 1 \le j \le k$$

$\mathrm{P}(i, k+1) = j$, pour " le j " réalisant le minimum, ci celui-ci est fini.
- Pour tout i variant de 1 à k

$$\mathrm{D}_{k+1}(k+1, i) = min \mathrm{M}(k+1, j) + \mathrm{D}_k(j, i) 1 \le j \le k$$

P(k + 1, i) = soit k + 1 soit P(j, i) selon que le minimum est réalisé par j = i ou non

- Pour tous les i et j variant de 1 à $k (i \neq j)$

$D_{k+1}(i,j) = min\{D_k(i,j), D_{k+1}(i,k+1) + D_{k+1}(k+1,j)\}$

P(i, j) = P(k + 1, j) si le minimum change [5].

L'algorithmes de DANTZIG :

/* Initialisations */

pour $i = 1$ à n faire

pour $j = 1$ a n faire

si $i \neq j$ alors $P(i,j) = 0$

$D(i,j) = \infty$

sinon $P(i,j) = i$

$D(i,j) = 0$

/* Corps de l'algorithme */

Pour $k = 1$ à n–1 faire

pour$i = 1$ à k faire

pour $j = 1$ à k faire

$d = D(i,j) + M(j,k+1)$

si $d < D(i,k+1)$ alors$D(i,k+1) = d; P(i,k+1) = j$

$d = M(k+1,j) + D(j,i)$

si $d < D(k+1,i)$ alors $D(k+1,i) = d$

si $i = j$ alors $P(k+1,i) = k+1$

sinon$P(k+1,i) = P(j,i)$

pour $i = 1$ à k faire

pour$j = 1$ à k faire

$d = D(i,k+1) + D(k+1,j)$

si $d < D(i,j)$ alors $D(i,j) = d$;$P(i,j) = P(k+1,j)$ [5].

Exemple :

Appliquons l'algorithme sur l'exemple précédent :

Initialement $D(i,i) = 0$ et $P(i,i) = i, D(i,j) = \infty$ et $P(i,j) = 0$ pour tout $i \neq j$

Il y a 5 étapes, k variant de 1 à 5 , examinons-les successivement (pour ne pas trop surcharger le déroulement de l'algorithme, seules certaines étapes sont détaillées)

- **pour** $k = 1$ $D(1,2) = D(1,1) + M(1,2) = 0 + \infty = \infty$ et $D(2,1) = M(2,1) + D(1,1) = 3 + 0 = 3$, donc $P(2,1) = 2$

- **pour** $k = 2$: $D(1,3) = min(D(1,1) + M(1,3), D(1,2) + M(2,3))$ = min $(0 + 6, \infty + \infty) = 6$, donc

$P(1,3) = 1$

$D(2,3) = \min(D(2,1)+M(1,3), D(2,2)+M(2,3)) = \min(3+6, 0+\infty) = 9$, donc $P(2,3) = 1$

$D(3,1) = \min(M(3,1)+D(1,1), M(3,2)+D(2,1)) = \min(\infty+0, \infty+3) = \infty$

$D(3,2) = min(M(3,1)+D(1,2), M(3,2)+D(2,2)) = min(\infty+3, \infty+0) = \infty$

$D(1,2) = min(D(1,2), D(1,3)+D(3,1)) = min(\infty, 6+\infty) = \infty$

$D(2,1) = min(D(2,1), D(2,3)+D(3,1)) = min(3, 9+\infty) = 3$

- pour k = 3 : il faut calculer D(1, 4), D(2, 4) et D(3, 4), par exemple :

$D(1,4) = min(D(1,1)+M(1,4), D(1,2)+M(2,4), D(1,3)+M(3,4)) = min(0+2, \infty+\infty, 6+1) = 2$, donc $P(1,4) = 1$

Il faut ensuite calculer D(4, 1), D(4, 2) et D(4, 3) puis finalement D(1, 2), D(1, 3), D(2, 1), D(2, 3), D(3, 1) et D(3, 2).

Nous obtenons :

$D(2,4) = 5(P(2,4) = 1), D(3,4) = 1(P(3,4) = 3); D(4,1) = 5(P(4,1) = 2), D(4,2) = 2(P(4,2) = 2), D(4,3) = 3(P(4,3) = 4); D(1,2) = 4(P(1,2) = 4), D(1,3) = 5(P(1,3) = 4), D(2,3) = 8(P(2,3) = 4), D(3,1) = 6(P(3,1) = 2) et D(3,2) = 3(P(3,2) = 4)$

- pour k = 4 : nous obtenons $D(1,5) = 8(P(1,5) = 3), D(2,5) = 11(P(2,5) = 3), D(3,5) = 3(P(3,5) = 3), D(4,5) = 6(P(4,5) = 3); D(5,1) = 2(P(5,1) = 5), D(5,2) = 6(P(5,2) = 4), D(5,3) = 7(P(5,3) = 4), D(5,4) = 4(P(5,4) = 1)$

Enfin, le seul changement :

$D(3,1) = 5(P(3,1) = 5)$

- pour k = 5 : nous obtenons $D(1,6) = 6(P(1,6) = 3, D(2,6) = 9(P(2,6) = 3), D(3,6) = 1(P(3,6) = 3), D(4,6) = 4(P(4,6) = 3), D(5,6) = 8(P(5,6) = 3); D(6,1) = 3(P(6,1) = 5), D(6,2) = 2(P(6,2) = 4), D(6,3) = 4(P(6,3) = 4), D(6,4) = 1(P(6,4) = 6), D(6,5) = 1(P(6,5) = 6)$

Enfin, les derniers changements sont :

$D(1,5) = 7(P(1,5) = 6), D(2,5) = 10(P(2,5) = 6), D(3,1) = 4(P(3,1) = 5), D(3,5) = 2(P(3,5) = 6), D(4,5) = 5(P(4,5) = 6)$

Il est facile d'établir.

(2) Méthode de Maria Hasse

La méthode de Maria Hasse (1961) permet de calculer la matrice M des coûts minimums dans un graphe valué $G = (X, A, C.)$, où $x = (x_1, x_2, x_3 \ldots .. x_n)$ et C est la matrice des coûts.

Considérons les opérations $\oplus$ et $\otimes$ définies sur l'ensemble R : $u\{O, \infty\} = [0, \infty]$, par $a \oplus b = min\{a, b\}$ et $a \otimes b = a + b$.

Calculons les puissances successives de la matrice $C = (C_{ij})$, en remplaçant les opérations habituelles d'addition de multiplication par les opérations $\oplus$ et $\otimes$ respectivement.

Par exemple, on aura C_{ij}^2=$(C_{ik} \otimes C_{kj})$ =min$\{C_{ik}, C_{kj}\}$ [8].

Théorème

Si k est tel que $C^k = C^{ktl}$, alors C^k est la matrice des coûts minimums [8].

Preuve :

On montre facilement, par induction sur k, que $(C_k)_{ij}$ est le minimum parmi les coûts des chemins de longeur inférieur à k de x_i à x_j.

En effet, ceci découle de la formule :

$(C^k)_{ij} = (C^{k-1} \times C)_{ij} = min\{C_{il}^{k-1} + C_{lj}\}$.

si $C^k = C^{k+1}$ alors

$\forall i, j, (C^k)_{ij}$ est le coût minimum pour tous les chemins de x_i à x_j, c-à-d $C_{ij}^k = m_{ij}$.

Remarque :

Comme un chemin minimum est toujours de longueur $<$ n $=$ ord G, on a toujours $C^{n-1} = C^n$ [8].

Exemple :

Considérons le graphe valué décrit par la Figure suivante :

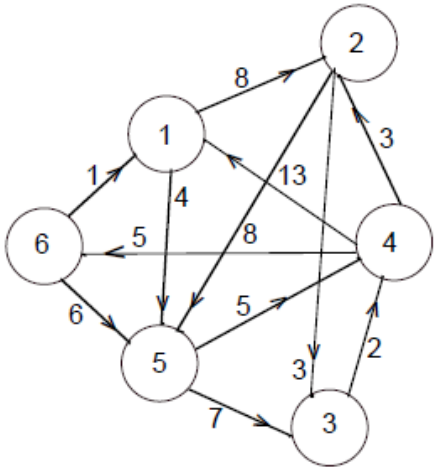


FIGURE 2.4 – Graphe valué

$$C = \begin{pmatrix} 0 & 8 & \infty & \infty & 4 & \infty \\ \infty & 0 & 3 & \infty & 8 & \infty \\ \infty & \infty & 0 & 2 & \infty & \infty \\ 13 & 3 & \infty & 0 & \infty & 5 \\ \infty & \infty & \infty & 5 & 0 & \infty \\ 1 & \infty & \infty & \infty & 6 & 0 \end{pmatrix}.$$

$$C^2=\begin{pmatrix} 0 & 8 & 11 & 9 & 4 & \infty \\ \infty & 0 & 3 & 5 & 8 & \infty \\ 15 & 5 & 0 & 2 & \infty & 7 \\ 6 & 3 & 6 & 0 & 10 & 5 \\ 18 & 8 & \infty & 5 & 0 & 10 \\ 1 & 9 & \infty & 11 & 5 & 0 \end{pmatrix}.$$

$$C^3=\begin{pmatrix} 0 & 8 & 11 & 9 & 4 & 14 \\ 18 & 0 & 3 & 5 & 8 & 10 \\ 8 & 5 & 0 & 2 & 13 & 7 \\ 6 & 3 & 6 & 0 & 10 & 5 \\ 11 & 8 & 11 & 5 & 0 & 10 \\ 1 & 9 & 12 & 10 & 5 & 0 \end{pmatrix}.$$

$$M=C^4\begin{pmatrix} 0 & 8 & 11 & 9 & 4 & 14 \\ 11 & 0 & 3 & 5 & 8 & 10 \\ 8 & 5 & 0 & 2 & 12 & 7 \\ 6 & 3 & 6 & 0 & 10 & 5 \\ 11 & 8 & 11 & 5 & 0 & 10 \\ 1 & 9 & 12 & 10 & 5 & 0 \end{pmatrix}.$$

Comme $c^{(3)}_{1,2} = 18 \neq c^{(4)}_{2,1} = 11$, le coût du chemin minimal de x_2 à x_1 est égal à 11.
Comme $c^3_{2,i}+c_{i,1}=11$ pour $i = 6$, le chemin se termine par l'arc (x_6, x_1) , de même $c^{(3)}_{2,6}=10=c^{(2)}_{2,i}+c_{i,6}$ pour $i = 4$, les deux derniers arcs sont (x_4, x_6),(x_6, x_1).
Enfin, $c^{(2)}_{2,4} =5= c_{2,3} + c_{3,4}$, le deuxième arc (x_3, x_4).
Finalement le chemin cherché est : x_2 , x_3 , x_4 , x_6 , x_1.

2.2.2 Problème de l'arbre couvrant de poids minimum :

Pour faire un réseau, on doit connecter des noeuds (sommets) par des lignes de communication (arêtes). L'établissement d'une ligne entre deux noeuds donnés a un certain coût (éventuellement infini).
Le problème est de choisir quelles lignes construite de façon que tous les noeuds soient interconnectés au coût total minimum.
Clairement, un tel réseau de coût minimal ne contient pas de cycle (sinon on pourrait enlever une des arêtes du cycle sans déconnecter).
Un tel réseau est donc un arbre libre (c'est-à-dire un graphe connexe sans cycle). Un tel arbre libre est en fait un arbre (avec racine) dont on "oublie" la racine.
Cet arbre est appelé arbre recouvrant de poids minimum (ARPM).

Deux algorithmes différents permettent de résoudre le problème, le premier dû à Prim, le second à Kruskal.

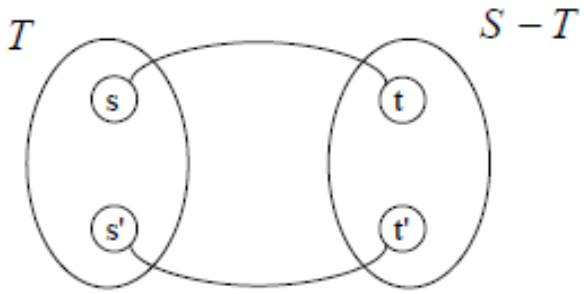
Propriété à la base des algorithmes :

Si les sommets S du graphe sont divisés en 2 sous ensembles T et $S-T$, alors tout ARPM contiendra exactement une arête de poids minimum reliant T à $S-T$, inversement, pour toute arête de ce type, il existe un ARPM qui la contient.

Supposons que l'ARPM ne contient aucune arête de poids minimum entre T et $S-T$. Soit st est une telle arête.

Puisqu'on a un arbre recouvrant, il existe un chemin entre s et t dans cet arbre. Quand on ajoute st, on fait donc apparaître un cycle et un seul. Il existe donc au moins une autre arête entre S et T. Si on enlève celle-ci, on a toujours un arbre (pas de cycle) de poids inférieur.

Le même argument montre qu'on peut toujours trouver un ARPM contenant cette arête st supposée de poids minimum.



(1) Algorithme de Prim

Principe de l'algorithme :

L'ensemble T est initialisé avec un sommet, puis, à chaque étape, on ajoute à cet ensemble le sommet de $S-T$ qui y est relié par l'arête de poids minimum, de cette façon on ne provoque pas de cycle, et comme le graphe est connexe, tous les sommets sont atteints.

À chaque étape de l'itération, on cherche si le sommet qu'on vient de rajouter est relié par une arête de poids plus faible aux sommets non encore ajoutés.

L'algorithme ressemble à celui de Dijkstra. La différence est dans la fonction d'actualisation. $P[i, j]$ est le poids de l'arête de ij et ∞ si i et j ne sont pas reliés.

On initialise S avec le sommet 1 et on note D[i] le poids minimum d'une arête reliant S au sommet i.

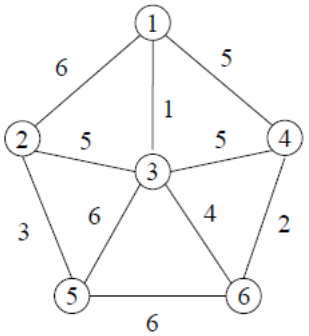
On choisit le sommet t qui a le D[t] minimum, puis on réévalue les D[i] des successeurs de t [6].

Algorithme de Prim :

fonction Prim

```
{
        s = 1 ;
        Pour chaque sommet i ≠ 1 faire          /* initialisation */
    {
        C[i] = 1,D[i] = P[1, i] ;
    {
        Pour k = 1 à n − 1 faire
    {                                   /Iterations /
      t = sommet∉ S tel que D[t] est minimun ;/*          */
      S = S ∪ {t} ;                     /sommet ajouté a E */
       pour chaque i ∉ E adjacent à t faire
      Si P[t, i] ≤ D[i]alors            /*Actualisation de D* /
      {
          D[i] = P[t, i] ;
          C[i] = t ;                        /*Actualisation de C* /
      }
    }
}
```

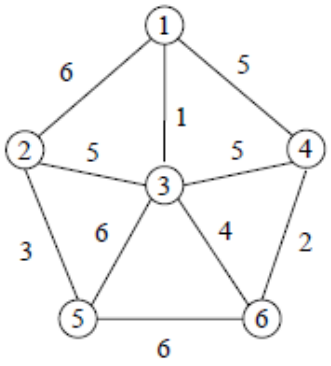
Exemple :



Itér.	S	t	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$
0	{1}	—	6	1	5	∞	∞
1	{1,3}	3	5	1	5	6	4
2	{1,3,6}	6	5	1	2	6	4
3	{1,3,6,4}	4	5	1	2	6	4
4	{1,3,6,4,2}	2	5	1	2	3	4
5	{1,3,6,4,2,5}	5	5	1	2	3	4

(2) Algorithme de Kruskal

Un autre algorithme, l'algorithme de Kruskal, construit un arbre, initialement vide, en choisissant à chaque étape l'arête de poids minimum reliant deux sommets non déjà reliés.
Au départ, tous les sommets sont dans des composantes différentes.
Si l'arête libre de poids minimum relie deux sommets dans des composantes distinctes, on la choisit et on fusionne les deux composantes auxquelles appartiennent ses extrémités.
Sinon, on l'écarte et on examine l'arête de poids immédiatement supérieur.
Exemple :



Composantes	Arête	Choix
1 2 3 4 5 6	13	oui
13 2 4 5 6	46	oui
13 2 46 5	25	oui
13 25 46	36	oui
1346 25	14	non
	23	oui

On obtient l'arbre couvrant de poids minimum de la Figure 2.5 :

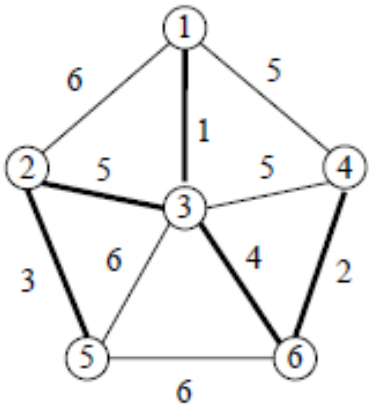


FIGURE 2.5 – Arbre couvrant de poids minimum

2.2.3 Recherche des points d'articulation

Un réseau de communications peut être représenté par un graphe non orienté. Pour assurer que deux noeuds quelconques puissent communiquer, il faut évidemment que le graphe soit connexe.
Il peut être utile de détecter d'éventuelles faiblesses d'un tel réseau. Si un noeud cesse d'être opérationnel, il ne peut plus retransmettre les informations qui lui arrivent. Si elles peuvent prendre un autre chemin tout va bien. Mais si certaines informations doivent nécessairement passer par ce noeud, il est à surveiller tout particulièrement.
Un sommet x d'un graphe est un point d'articulation s'il existe deux sommets s et t, différents de x, tels que tout chemin entre s et t passe par x.
La recherche des points d'articulation d'un graphe peut se faire assez simplement par une exploration en profondeur.
Lorsqú'une exploration en profondeur est faite sur un graphe non orienté, il ne peut pas y avoir d'arête transversé. Les seules arêtes autres que celles de l'arbre sont donc des arêtes de retour.
Si la racine de l'arborescence a deux fils au moins, elle est donc un point d'articulation.
La seule possibilité pour qu'un sommet x autre ne soit pas un point d'articulation est que chacun de ses fils ait un descendant connecté par une arête de retour à un ancêtre de x.
La Figure 2.6 montre que les sommets 1 et 3 sont des points d'articulation.

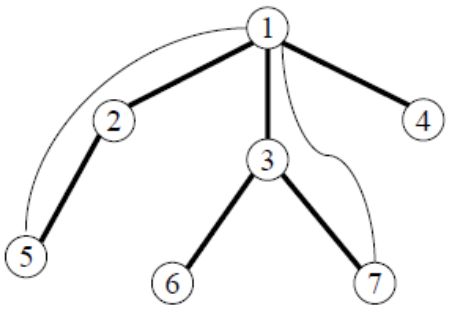
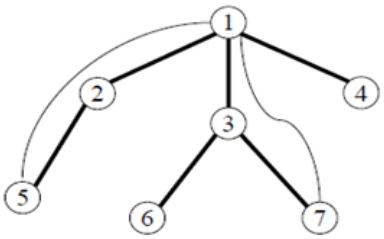


FIGURE 2.6 – Les sommets 1 et 3 sont des points d'articulation

Pour repérer les ancêtres, on calcule le numéro d'ordre préfixe, $ord(x)$ dans le courant d'une exploration en profondeur. Ce numéro classe les sommets dans l'ordre dans lequel la procédure récursive d'exploration est appelée.

Exemple :



	1	2	3	4	5	6	7
ord	1	2	4	7	3	5	6
mini	-	1	1	1	1	4	1

La fonction $mini(x)$ est la plus petite valeur de $ord(y)$ pour un sommet y relié à x ou à un descendant de x.

Si x a un fils s tel que $mini(s) = ord(x)$, cela signifie qu'aucune arête ne relie un descendant de s à un ancêtre de x. Donc x est un point d'articulation (x est différent de la racine).

Algorithme de recherche des points d'articulation :

```
fonction INITIALISE() {
        compteur = 0 ;
        pour chaque sommet t faire
        {
            explore[t] = 0; ord[t] = 0 ;
        }
}
fonction ENTIER ARTICULATION(sommet s) /* retourne mini[s] */
{
        ord[s] = compteur++ ; /* calcule ord(s) */
        mini_s = MAXINT ;
        pour chaque successeur t de s faire
        si (explore[t] == 1) alors mini_s = min(mini_s, ord[t])
                /* t vaut s ou un ancetre de s */
        sinon
                 /* t sera un fils de s */
            explore[t] = 1; mini_t = ARTICULATION(t) ;
            mini_s = min(mini_s, mini_t)) ;
            si (mini_t == ord[s]) alors s est point d'articulation ;
                /* le fils t de s verifie mini[t] = ord[s] */
```

```
}
        retour mini_s ;
}

                    /* On suppose que la racine est 1 */
fonction RECHERCHE POINTS ARTICULATION()
{
        INITIALISE(); explore[1] = 1; ord[1] = compteur++; nbre_fils = 0;
        pourchaquesuccesseursde1faire
        si(explore[s] == 0)alors
{
            explore[s] = 1; nbre_fils++; ARTICULATION(s);
                /*lavaleurdeminipourlesfilsdelaracinen'ontpasd'importance*
/
}
        sinbre_fils > 1alors1estpointd'articulation;
}
```

On suppose que la structure du graphe comporte toutes les arêtes dans les deux sens [8].

2.2.4 Problème d'affectation

Plusieurs chercheurs se sont intéressés au problème d'affectation, travaux ont permet de donner des algorithmes de résolutions parmi eux l'algorithme hongroise qui est définie comme suit :

(1)Méthode d'hongroise

L'algorithme d'hongrois appelé aussi algorithme de Kuhn est un algorithme d'optimisation combinatoire, qui résout le problème d'affectation en temps polynomial. Il a été proposé en 1955 par le mathématicien américain Harold Kuhn, qui l'a baptisé ; méthode hongroise . Dans notre étude nous nous intéressons au cas où le nombre de machines est égales aux nombres de tâches.

1-1 Algorithme d'hongroise :

Étape 1 : Réduction de la matrice des coûts C : on soustrait le plus petit élément de chaque ligne aux éléments de cette ligne puis, on soustrait le plus petit élément de chaque colonne aux éléments de cette colonne.

Étape 2 : Encadrer un zéro par ligne et par colonne en commençant par la ligne ayant le moins de zéro possible puis barrer les autres zéros se situant sur la même ligne et la même colonne. Si l'affectation est optimal alors terminer sinon aller à l'étape 3.

Étape 3 : Éliminer les lignes et les colonnes contenant des zéros de sorte que le nombre de telles ligne et colonne soit minimum. On opère de la manière suivante :

a- Repérer par une étoile les lignes qui n'ont pas d'affectation.

b- Repérer par une étoile les colonnes qui ont des zéros barrés dans la ligne repéré.

c- Repérer parmi les autres lignes celles qui ont une affectation dans la colonne repéré.

d- Répéter les étapes b et c jusqú'au moment où il n'est plus possible d'ajouter un repère.

e- Éliminer, tracer un trait sur les lignes non repérées et les colonnes repérées.

Étape 4 : Distinguer la valeur la plus petite des données restantes (qui n'est pas couverte par un trait). Soit α cette valeur la matrice des coûts de la prochaine itération $C(k+1)$ se calcule de la manière suivante :

- $C_{ij}^{(k+1)}=C_{ij}^{(k)}$, si la cellule (i,j) est couverte soit par un trait horizontal ou un trait vertical.

- $C_{ij}^{(k+1)}=C_{ij}^{(k)}-\alpha$, si la cellule (i,j) n'est pas couverte par un trait horizontal ni par un trait vertical.

- $C_{ij}^{(k+1)}=C_{ij}^{(k)}+\alpha$; si la cellule (i,j) est couverte par un trait horizontal et un trait vertical.

Aller à l'étape 2 [8].

Exemple :

Étape 1 :Réduction de la matrice :

	1	2	3	4	5
A	7	3	5	7	10
B	6	∞	∞	8	7
C	6	5	1	5	∞
D	11	4	∞	11	15
E	∞	4	5	2	10

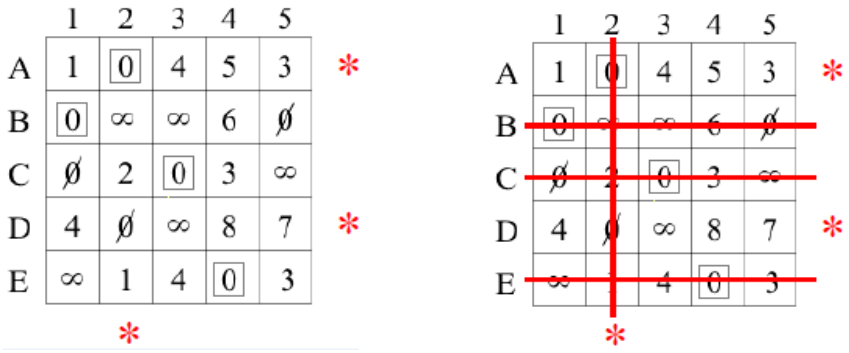
	1	2	3	4	5
A	1	0	4	5	3
B	0	∞	∞	6	0
C	0	2	0	3	∞
D	5	1	∞	9	8
E	∞	1	4	0	3

	1	2	3	4	5
A	1	0	4	5	3
B	0	∞	∞	6	0
C	0	2	0	3	∞
D	4	0	∞	8	7
E	∞	1	4	0	3

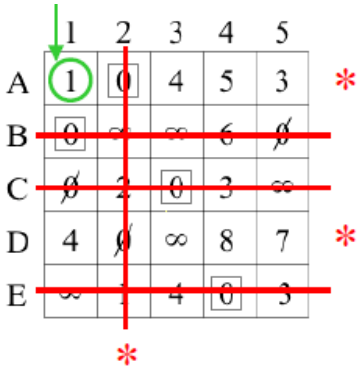
Étape 2 : Encadrement des zéros :

	1	2	3	4	5
A	1	[0]	4	5	3
B	[0]	∞	∞	6	~~0~~
C	~~0~~	2	[0]	3	∞
D	4	~~0~~	∞	8	7
E	∞	1	4	[0]	3

Étape 3 : Elimination des lignes et des colonnes :



Étape 4 : Construction de la nouvelle matrice



	1	2	3	4	5
A	0	0	3	4	2
B	0	∞	∞	6	0
C	0	3	0	3	∞
D	3	0	∞	7	6
E	∞	2	4	0	3

Étape 2 : Encadrement des zéros de la nouvelle matrice :

	1	2	3	4	5
A	0	~~0~~	3	4	2
B	~~0~~	∞	∞	6	0
C	~~0~~	3	0	3	∞
D	3	0	∞	7	6
E	∞	2	4	0	3

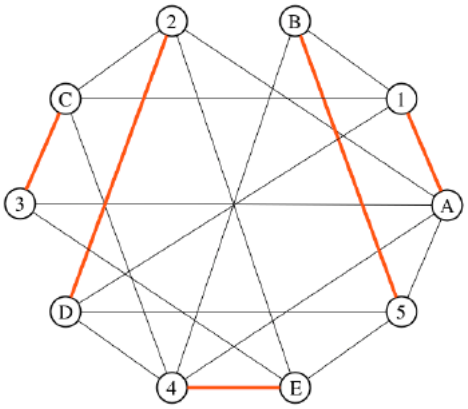
Solution optimale

	1	2	3	4	5
A	0	~~0~~	3	4	2
B	~~0~~	∞	∞	6	0
C	~~0~~	3	0	3	∞
D	3	0	∞	7	6
E	∞	2	4	0	3

La solution est :

$C_{A1} + C_{B5} + C_{C3} + C_{D2} + C_{E4} = 7 + 7 + 1 + 4 + 2 = 21$

	1	2	3	4	5
A	7	3	5	7	10
B	6	∞	∞	8	7
C	6	5	1	5	∞
D	11	4	∞	11	15
E	∞	4	5	2	10



2.2.5 Problème d'ordonnancement

On a affaire à un problème d'ordonnancement lorsque l'on est confronté à un problème d'organisation. Il faut accomplir de multiples taches qui demandent un certain temps d'exécution et qui doivent être exécutées dans un certain ordre.
II est donc nécessaire d'identifier les taches prioritaires en fonction de l'objectif à atteindre et, lors de leur exécution, il faut détecter les retards et les dépassements de moyens.
Un problème d'ordonnancement consiste alors à organiser dans le temps la réalisation de tâches d'un projet, compte tenu de contraintes temporelles (délais, contraintes d'enchaînement) et de contraintes portant sur la disponibilité des ressources requises afin de minimiser la durée globale du projet.

Contexte

Un projet consiste en un ensemble de n tâches liées par des contraintes de succession ou de précédence, l'objectif est de :

1. Calculer la durée minimale du projet.
2. Déterminer les dates de début au plus tôt et au plus tard des tâches.
3. Déterminer les tâches critiques.

Il faut donc identifier les tâches, leurs durées et les contraintes de succession ou de précédence entre ces différentes tâches. Nous ne considérons pas les contraintes liées aux ressources dans ce cours (les ressources sont supposées illimitées).

Les tâches :

Une tâches est l'activité élémentaire caractérisée par :

- Sa durée d_i.
- Sa date de début t_i et sa date de fin f_i. La date de fin est égale a $t_i + d_i$.
- Eventuellement les ressources nécessaires a son accomplissement.

Les tâches sont supposées non-préemptives : une fois que l'exécution a commencé, elles se poursuit sans interruption jusqú'a la fin.
Il existe entre les tâches d'un projet des relations de précédence, qui signifient par exemple qu'une tâche ne peut débuter que si certaines autres tâches sont finies [9].

Diagramme de Gantt :

On peut représenter sur le graphique l'avancement des travaux. En effet on indique par une barre le travail effectivement accompli depuis que les tâches ont été commencées. Si on procède ainsi pour toutes les tâches on saura à tout moment quel est l'état réel d'avancement du projet.

Considérons le problème d'ordonnancement suivant :

Tâches	Durée	Contraintes
a	6	-
b	3	-
c	6	-
d	2	b achevée
e	4	b achevée
f	3	a et b achevées

Le diagramme de Gantt correspondant au tableau ci dessus est donné dans la Figure 2.7 :

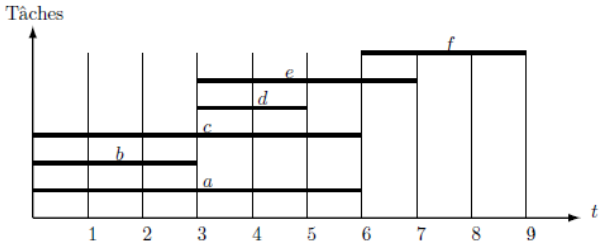


FIGURE 2.7 – Diagramme de Gantt

Représentation par un graphe potentiels-tâches :

On peut représenter un problème d'ordonnancement simple par un graphe oriente $G = (X, A)$:

- Chacun des sommets de X représente une tâche.
- Les arcs A représente les contraintes de potentiels (de précédence en général) :

$$A = (i, j) \in X \times X, \exists \text{ une contrainte } t_j - t_i \geq a_{ij}$$

Le poids associe a un arc (i, j) est $p_{ij} = a_{ij}$.

- On ajoute a cela deux sommets 0 et * représentant les tâches fictives début et fin de projet.

Exemple :

Le graphe potentiels-tâches du problèmes précédent est donné ci-dessous :

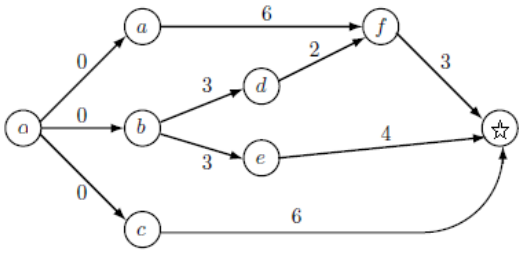


FIGURE 2.8 – Le graphe Potentiels−Tâches

Recherche d'un ordonnancement réalisable :

Théorème Il existe un ordonnancement réalisable si ∃ de circuit de valeur strictement positive dans G.

Si la condition d'existence est vérifiée alors, il existe en général plusieurs ordonnancements réalisables de durée minimale. On distinguera deux cas particuliers :

1. Ordonnancement au plus tôt.
2. Ordonnancement au plus tard (avec une date limite d'achèvement du projet imposée) [10].

Dates de début au plus tôt :

Définition : La date de début au plus tôt $\underline{t_i}$ d'une tâche i est égale à la longueur du plus long chemin de la tâche début (ou 0) à i.

Théorème Dans le cas d'un graphe sans circuit, on a :

$$\underline{t_0} = 0$$
$$\underline{t_i} = \max_{j \in \Gamma^-(i)} (\underline{t_j} + v_{ji})$$

$C_{j,i}$ est la capacité de l'arc (j, i) (par exemple la durée de j) $\Gamma^-(i)$ est l'ensemble des prédécesseurs de i [10].

Dates de début au plus tard :

On souhaite terminer le projet au plus tard à une date $D \geq \underline{t_{n+1}}$

Définition

La date de début au plus tard $\overline{t_i}$ de i est la date maximum à laquelle on peut exécuter i sans retarder le chantier (date D).

Théorème $\overline{t_{n+1}} = D$

$\overline{t_i} = \overline{t_{n+1}}-$ valeur d'un plus long chemin entre i et $n+1$ [10].

Théorème Dans le cas d'un graphe sans circuit, on a :

$$\overline{\overline{t_*}} = D$$
$$\overline{t_i} = \min_{j \in \Gamma^+(i)} \{\overline{t_j} - v_{ji}\}$$

$C_{i,j}$ est la capacité de l'arc (i, j) (par exemple la durée de i), $\Gamma^+(i)$ est l'ensemble des successeurs de i [10].

Marges totales et chemin critique :

Définition :

La marge totale d'une tâche i est le retard total qu'on peut se permettre sur i sans remettre en cause la date de fin du projet.

$$MT_i = \overline{t_i} - \underline{t_i}$$

Définition :

Les tâches critiques ont une marge nulle par extension si $MT_i = MT_{n+1}$.

Tout retard sur leur exécution entraîne un retard global sur le projet.

Un chemin est critique s'il relie Début à Fin et s'il ne contient que des tâches critiques.

2.2.6 Problème de transport

(1)Présentation du problème :

C'est en 1941 que Frank Lauren Hitchcock à formulé pour la première fois le problème de transport, et en suite 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.

Un problème de transport peut être défini comme l'action de transporter des marchandises où des produits fabriqués 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.

(2)Formulation du problème :

Avant de commencer la formulation du problème, considérant la notation suivante :

a_i = la quantité disponible du produit à l'origine i.

b_j = la quantité requise à la destination j.

X_{ij} = quantité transportée des unités de l'origine i vers la destination j.
C_{ij} = le coût de transport d'une unité de l'origine i vers la destination j.
On peut d'écrire un problème de transport de la façon suivante :
Une quantité donnée d'un produit uniforme est disponible à chacune des origines (par exemple des dépôts ou usines). Il s'agit d'envoyer des quantités spécifiées à chacune des destinations (par exemple des points de vente). On connaît le coût de transport d'une unité de l'une des origines vers l'une des destinations.
En supposant qu'il est possible d'expédier des produits depuis n'importe quelle origine vers n'importe quelle destination, il s'agit de déterminer le coût de transport minimum des origines et n destination.
Nous supposerons qu'il y a m origines et n destinations. La variable X_{ij} représentera le nombre d'unités expédiées de l'origine i vers la destination j .$X_{ij} \geq 0$ pour tout i, j.
Pour chaque origine i donnée, il y a n valeurs de j possibles, cela implique qu'il y a $(m \times n)$ X_{ij} différents.
Donc on peut modéliser le problème par le graphe simple biparti suivant :

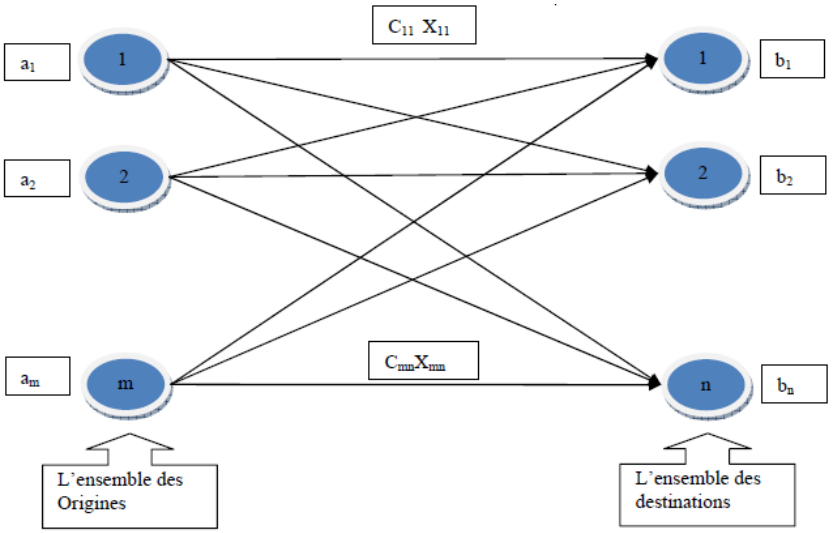


FIGURE 2.9 – Représentation du problème sous forme d'un graphe

Donc on peut modéliser le problème de transport de la manière suivante :

$$\begin{cases} Z = Z(x) = \sum_{i=1}^{m}\sum_{j=1}^{n} C_{ij}X_{ij}, (1) ; \\ \sum_{j=1}^{n} X_{ij} = a_i, i = \overline{1,m}, (2) ; \\ \sum_{i=1}^{m} X_{ij} = b_j, j = \overline{1,n}, (3) ; \\ X_{ij} \geq 0, i = \overline{1,m}, j = \overline{1,n}, (4). \end{cases}$$

L'ensemble $x = \{x_{ij}, i = \overline{1,m}, j = \overline{1,n}\}$ est dit plan de transport du problème $(1)-(4)$ s'il satisfait les contraintes (2),(3),(4).

Il est dit plan optimal du transport si de plus il réalise le minimum de la fonction objectif (1).

(3)Condition d'existence d'un plan optimal de transport :

Dans le problème (1)-(4), on suppose que les quantités a_i et b_j sont toutes positives ou nulles, de plus, pour que le problème de transport soit réalisable, il faut que les quantités produites soit supérieur ou égal aux quantités demandées, c'est-à-dire :

$\sum_{i=1}^{m} a_i \geq \sum_{j=1}^{n} b_j$ **Remarque :**

- Dans le cas où on aurait la condition (5), on crée un dépôt fictif (supplémentaire), où on transforme la quantité produite restante pour avoir l'égalité :
  $\sum_{i=1}^{m} a_i \doteq \sum_{j=1}^{n} b_j = K$ (6)
  Par la suite on supposera quand a toujours la condition (6) appelé condition de balance c'est-à-dire l'offre est égale à la demande.
- Le problème $(1)-(4)$ possède un plan optimal de transport si et seulement si la condition de balance (6) est vérifiée.

(4)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 de la solution de base.

[11] **4.1 Recherche d'une solution de base initiale :**

Pour le problème $(1)-(4)$ il existe plusieurs méthodes d'obtention d'un plan basique initial, les deux méthodes les plus utilisées sont les suivantes :

4.1.1 Méthode du coin nord-ouest de l'angule :

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 à cette raisons là que généralement la solution obtenue par cette méthode est loin de la solution optimale [11].

Règle du coin Nord-Ouest :

On choisit la case (1,1) située au coin nord-ouest du tableau de transport, et on lui affecte la

quantité $x_{11} = \min\{a_1, b_1\}$.

Deux cas peuvent alors se présenter :

- i) Si $x_{11}=a_1$, alors la quantité de A_1 est entièrement transporté et ceci sature la première ligne du tableau.

  Dans le tableau réduit, on remplacera b_1 par $(b_1 - x_{11})$ et on répétera la même procédure que précédemment.
- ii) Si $x_{11}=b_1$, alors la demande du point de distribution B_1 est entièrement satisfaite par A_1 et ceci sature la première colonne.

  Dans le tableau réduit, on remplacera a_1 par $(a_1 - x_{11})$ et on fera la même procédure.

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

Exemple :

Dans le tableau suivant, trouver un plan basique initial par la méthode du coin nord-ouest :

	B_1	B_2	B_3	B_4	a_i
A_1	1	4	2	3	20
A_2	5	1	3	4	21
A_3	2	5	1	2	35
b_j	12	23	28	13	76

En appliquant la méthode du coin nord-ouest, on trouve

$x_{11}= \min\{20,12\} = 12$; $x_{12}= \min\{20-12,23\} = 8$;

$x_{22}= \min\{21,23-8\}=15$; $x_{23}= \min\{21-15,28\} = 6$;

$x_{33}= \min\{35,28-6\} = 22$; $x_{34}= \min\{35-22,13\} = 13$.

Les composantes basiques du plan x sont représentées en gras dans le tableau de transport ci-dessous et $x_{ij}=0$ pour les cases restantes :

4.1.2 Méthode de l'élément minimal :

Cette méthode donne en général un plan basique initial plus proche du plan basique optimal que celui obtenu par la méthode du coin nord-ouest.

	B_1	B_2	B_3	B_4	a_i
A_1	1 **(12)**	4 **(8)**	2	3	20
A_2	5	1 **(15)**	3 **(6)**	4	21
A_3	2	5	1 **(22)**	2 **(13)**	35
b_j	12	23	28	13	76

Le principe consiste à choisir au début une case (i_1, j_1) qui correspond à l'élément $c_{i_1 j_1}$ tel que : $c_{i_1 j_1}$=min $\{c_{ij}, 1 \leq i \leq m, 1 \leq j \leq n\}$, puis on posera x_{i_1,j_1}=min $\{a_{i_1}, b_{j_1}\}$ dans la case (i_1, j_1).
Si $x_{i_1 j_1}=a_{i_1}$, on exclut la ligne i_1 et on remplace le nombre b_{j_1} par $(b_{j_1}\text{-}x_{i_1 j_1})$.
Dans le cas où $x_{i_1 j_1}=b_{i_1}$, on exclut la ligne j_1 et on remplace le nombre a_{i_1} par $(a_{i_1}\text{-}x_{i_1 j_1})$.ensuite on refait la même procédure avec le tableau réduit.
Ce processus sera répété $(m+n-1)$ fois et permettra de trouver les $(m+n-1)$ variables basiques du plan initial recherché.

Exemple :

Par la méthode de l'élément minimal, trouver un plan basique initial dans le même tableau de transport précèdent :

	B_1	B_2	B_3	B_4	a_i
A_1	1	4	2	3	20
A_2	5	1	3	4	21
A_3	2	5	1	2	35
b_j	12	23	28	13	76

On a :

$c_{11} = \min i, j c_{ij} = 1, x_{11} = min\{20, 12\} = 12;$

$c_{22} = \min j \neq i c_{ij} = 1, x_{22} = min\{22, 23\} = 21;$

$c_{33} = \min j \neq 1, i \neq 2 c_{ij} = 1, x_{35} = min\{35, 28\} = 28;$

$c_{34} = \min j \neq 1,3, i \neq 2 c_{ij}, x_{34} = min\{35-28,13\} = 7;$

$c_{14} = \min j \neq 1,3, i \neq 2,3 c_{ij}, x_{14} = min\{20-12,13-7\} = 6;$

$c_{12} = \min j \neq 1,3,4, i \neq 2,3 c_{ij}, x_{12} = min\{20-18,23-21\} = 2;$

Les composantes basiques du plan basique initial obtenu sont représentées dans le tableau de transport suivant et $x_{ij} = 0$ pour les cases restantes :

	B_1	B_2	B_3	B_4	a_i
A_1	1 **(12)**	4 **(2)**	2	3 **(6)**	20
A_2	5	1 **(21)**	3	4	21
A_3	2	5	1 **(28)**	2 **(7)**	35
b_j	12	23	28	13	76

Soient x et x' les plans basiques obtenus respectivement par la méthode du coin nord-ouest et par l'élément minimal, alors on aura :

$Z(x) = (1*12) + (4*8) + (1*15) + (3*6) + (1*22) + (2*13) = 125,$

$Z(x') = (1*12) + (4*2) + (3*6) + (1*21) + (1*28) + (2*7) = 101.$

On remarque bien que $Z(x') < Z(x)$

4.1.3 Méthode des potentiels :

Soit x un plan basique de transport de départ, auquel correspond U_B.

Si le critère d'optimalité : $\Delta_{ij} \leq 0 (i,j) \in U_H$ n'est pas vérifié, alors on cherche une case $(i_0, j_0) \in U_H$; $\Delta_{i_0 j_0} = \max \Delta_{i_0 j_0}, (i,j) \in U_H$.

A l'aide de la case (i_0, j_0) et des cases de U_B, on construit un cycle qui est d'ailleurs unique. Puis on affecte des signes successivement (+) et (-) aux sommets de ce cycle, en commençant par le sommet (i_0, j_0) affecté du signe (+) et en se mouvant dans le sens des aiguilles d'une montre ou dans le sens contraire.

Parmi les sommets du cycle affectés du signe (-), on choisit celui où la variable x_{ij} est minimale et on pose : $\theta^n = \min x_{ij} = x_{i_1 j_1}$.

Pour les sommets affectés du signe (+), on ajoute aux variables x_{ij} la quantité $\theta * ^0$ et on soustrait la même quantité des variables x_{ij}, correspondantes aux sommets affectés de signe (-).

Toutes les autres variables x_{ij} resteront inchangées.

On obtient ainsi un nouveau plan basique de transport $\overline{x}$, avec un nouveau ensemble basique $\overline{U}_B=\{U_B \setminus (i_1,j_1)\} \cup (i_0,j_0)$.

Cette itération sera répétée jusqú'à ce que le critère d'optimalité soit vérifié [11].

Exemple :

Résoudre par la méthode des potentiels le problème de transport, présenté dans l'exemple 2 :

a) En commençant par le plan basique x trouvé dans l'exemple 2.

b) En commençant par le plan basique x' trouvé dans l'exemple 3.

Dressons le tableau de transport avec les variables basiques x trouvées dans l'exemple 2 en utilisant la méthode du coin nord-ouest :

	B_1	B_2	B_3	B_4	a_i
A_1	1 **(12)**	4 **(8)**	2	3 **(6)**	20
A_2	5	1 **(15)**	3 **(6)**	4	21
A_3	2	5	1 **(22)**	2 **(13)**	35
b_j	12	23	28	13	76

En posant $u_1 = 0$, on calcule les autres potentiels par la formule : $u_i + v_j - c_{ij} = 0, (i,j) \in U_B$. On placera les u_i sur une colonne à droite des a_i et les v_j sur une ligne au dessous des b_j. En suite en utilisant la formule $\Delta_{ij} \leq 0$ $(i,j) \in U_H$, on trouvera les estimations Δ_{ij} qu'on va placer en bas et à droite des cases non basiques.

Le nouveau tableau obtenu possède alors la forme suivante :

Le plan basique initial n'est pas optimal, car $\Delta_{i_0 j_0} = \max \Delta_{ij} = \Delta_{13} = 4 > 0$ A l'aide de la case

	B_1	B_2	B_3	B_4	a_i	u_i
A_1	1 **(12)**	4 - **(8)**	2 +	3 4	20	0
A_2	5 -7	1 + **(15)**	3 - **(6)**	4 0	21	-3
A_3	2 -6	5 -6	1 **(22)**	2 **(13)**	35	-5
b_j	12	23	28	13	76	
v_j	1	4	6	7		

$(1,3)$, on construit le cycle $(1,3) \rightarrow (2,3) \rightarrow (1,2) \rightarrow (1,3)$, on aura alors :

$\theta^0 = x_{i_1 j_1} = \min\{8,6\} = 6$.

Pour le nouveau plan basique $\overline{x}$, on obtient :

$\overline{x_{13}} = x_{13} + \theta^0 = 6$; $\overline{x_{22}} = x_{22} + \theta^0 = 21$; $\overline{x_{12}} = x_{12} - \theta^0 = 2$; $\overline{x_{23}} = x_{23} - \theta^0 = 0$ Les autres variables $\overline{x_{ij}}$ resteront inchangées.

On commencera donc une nouvelle itération avec le tableau suivant :

	B_1	B_2	B_3	B_4	a_i	u_i
A_1	1 **(12)**	4 **(2)**	2 (6)	3 0	20	0
A_2	5 -7	1 **(21)**	3 -4	4 -4	21	-3
A_3	2 -2	5 -2	1 **(22)**	2 **(13)**	35	-1
b_j	12	23	28	13	76	
v_j	1	4	2	3		

Le critère d'optimalité est vérifié dans ce tableau, donc $x^0 = \{x_{ij}^0, 1 \le i \le 3, 1 \le i \le 4\}$ avec :

$x_{11}^0 = 12, x_{12}^0 = 2, x_{13}^0 = 6, x_{14}^0 = 0, x_{21}^0 = 0, x_{22}^0 = 21, x_{23}^0 = 0,\ x_{24}^0 = 0, x_{31}^0 = 0, x_{32}^0 = 0, x_{33}^0 = 22, x_{34}^0 = 13, est optimale et Z^0 = min Z(x) = Z(x^0) = 101.$ Dressons le tableau de transport avec les variables

basiques x' trouvées dans l'exemple 3 en utilisant la méthode de l'élément minimal :

En posant $u_1 = 0$, on calcule les autres potentiels par la formule(3) :

	B_1	B_2	B_3	B_4	a_i
A_1	1 **(12)**	4 **(2)**	2	3 (6)	20
A_2	5	1 **(21)**	3	4	21
A_3	2	5	1 **(28)**	2 **(7)**	35
b_j	12	23	28	13	76

$v_1 = c_{11} - u_1 = 1 - 0 = 1, v_2 = c_{12} - u_1 = 4 - 0 = 4$, $v_4 = c_{14} - u_1 = 3 - 0 = 3$

$u_2 = c_{22} - v_2 = 1 - 4 = -3$, $u_3 = c_{34} - v_4 = 2 - 3 = -1, v_3 = c_{33} - u_3 = 1 - (-1) = 2$

calculons les estimations Δ_j :

$\Delta_{ij} = 0,\ (i,j) \in U_B$

$\Delta_{ij} = u_i + v_j - c_{ij}$, $(i,j) \in U_H$

$\Delta_{13} = u_1 + v_3 - c_{13} = 0$, $\Delta_{21} = u_2 + v_1 - c_{21} = -7$, $\Delta_{23} = u_2 + v_3 - c_{23} = -4$

$\Delta_{24} = u_2 + v_4 - c_{24} = -4,\ \Delta_{31} = u_3 + v_1 - c_{31} = -2$, $\Delta_{32} = u_3 + v_2 - c_{32} = -2$

On obtient alors le tableau suivant :

	B_1	B_2	B_3	B_4	a_i	u_i
A_1	1 **(12)**	4 **(2)**	2	3 **(6)**	20	0
A_2	5	1 **(21)**	3	4	21	-3
A_3	2	5	1 **(28)**	2 **(7)**	35	-1
b_j	12	23	28	13	76	
v_j	1	4	2	3		

Le critère d'optimalité est vérifié, donc le plan de transport basique $\acute{x}$ est **optimal** , avec $\mathbf{Z}(\acute{x})$**=101**

Conclusion

Ce chapitre a été consacré aux méthodes de résolution de différents problèmes d'optimisation dans les réseaux ainsi à quelques exemples pour mieux comprendre leurs résolution.

Chapitre 3

Résolution de quelques problèmes concrets

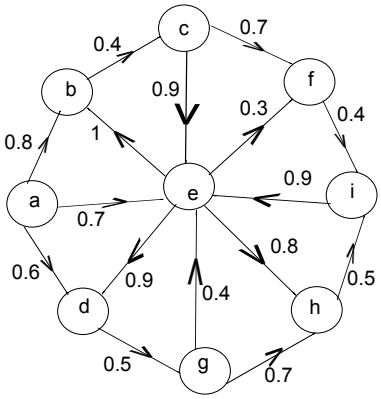
Nous allons présenter quelques situations concrètes pour lesquelles nous allons modéliser sous forme de graphe afin d'optimiser certains paramètres en utilisant les algorithmes cités dans le chapitre précédent.

3.1 Résolution d'un problème de Télécommunication :

Problème :

On désire relier par un réseau un serveur central a à divers utilisateurs avec des intermédiaires éventuels pour chacune des lignes possibles i et j, on a une vitesse de transmission C_{ij} en Mb/s. Une vitesse de transmission d'une chaîne a à k (k est un utilisateur) est égale au plus petit C_{ij} rencontrés sur la chaîne.

Quel est le chemin le plus fiable pour relier le serveur a à un utilisateur i ?



On cherche le chemin pour lequel le produit des probabilités soit maximal. Pour cela, on peut adapter l'algorithme de Dijkstra de la façon suivante :

Étape1 : Initialisation

S={s} ,$\lambda_s = 1$,$\lambda_j = 0$,$j \in \bar{S}$

Étape2 : Calculer le cpf entre s et les autres sommets comme suit :

$$\begin{cases} \lambda_j = C_{sj}, & si \quad j \in \Gamma^+(s) \\ \lambda_j = 0, & sinon \end{cases}$$

Étape3 : Sélectionner un sommet $k \in \bar{S}$ vérifiant :

$\lambda_k = max\lambda_j$

si $\bar{s} = \{\emptyset\}$ alors

arrêter l'algorithme

sinon

aller à l'étape 4

Étape4 : Mettre à jour les plus longues distances pour tout $j \in \Gamma^+(k)$ avec $j \in \bar{s}$ faire

$$\lambda_j = max\{\lambda_j, \lambda_k \times P_{kj}\}$$

retourner à l'étape 3.

Résolution de notre problème en appliquant l'algorithme de dijkstra modifié :

Itération 1 : Initialisation

$S = \{a\}$, $\overline{S} = \{b, c, d, e, f, g, h, i\}$

j	a	b	c	d	e	f	g	h	i
λ_j	1	0	0	0	0	0	0	0	0

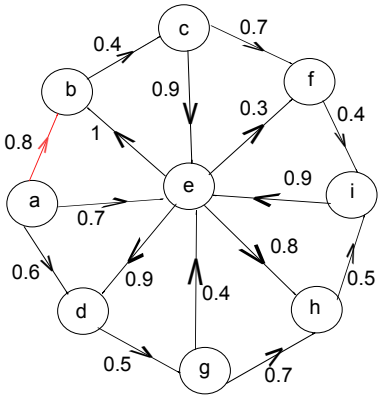
$\Gamma^+(a) = \{b, d, e\}$

Itération2 : Mettre à jour les fiabilités des chemins partant de a

j	a	b	c	d	e	f	g	h	i
λ_j	1	0.8	0	0.6	0.7	0	0	0	0

$\Gamma^+(b) = \{c\}$

$k = \{b\}$, $S = \{a, b\}$,$\overline{S} = V/S = \{c, d, e, f, g, h, i\}$

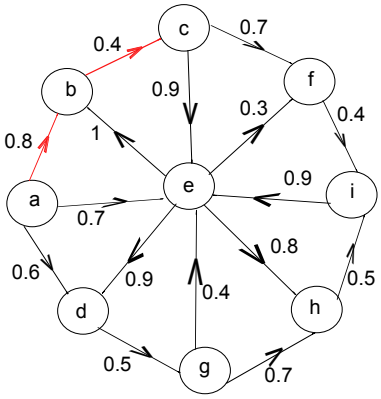


Itération 3 :

j	a	b	c	d	e	f	g	h	i
λ_j	1	0.8	0.32	0.6	0.7	0	0	0	0

$\Gamma^+(c) = \{f, e\}$

$k = \{c\}$, $S = \{a, b, c\}$, $\overline{S} = \{d, e, f, g, h, i\}$ et $\lambda_c = 0.8 * 0.4 = 0.32$

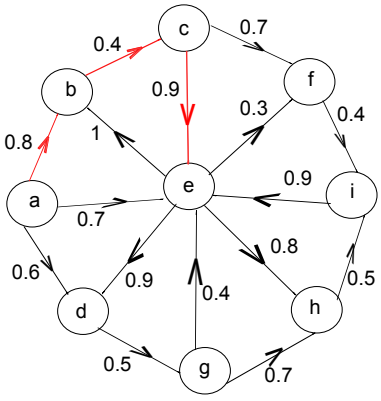


Itération 4 :

j	a	b	c	d	e	f	g	h	i
λ_j	1	0.8	0.32	0.6	0.7	0.224	0	0	0

$\Gamma^+(e) = \{f, h, d\}$

$k = \{e\}$, $S = \{a, b, c, e\}$, $\overline{S} = \{d, f, g, h, i\}$ et $\lambda_e = 0.32 * 0.9 = 0.288$

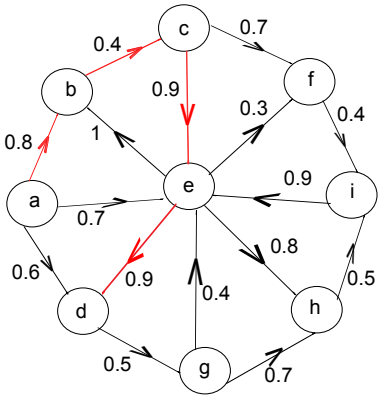


Itération 5 :

j	a	b	c	d	e	f	g	h	i
λ_j	1	0.8	0.32	0.26	0.29	0.09	0	0.23	0

$\Gamma^+(e) = \{f, h, d\}$

$k = \{d\}$, $S = \{a, b, c, e, d\}$, $\overline{S} = \{f, g, h, i\}$ et $\lambda_d = 0.29 * 0.9 = 0.2592$

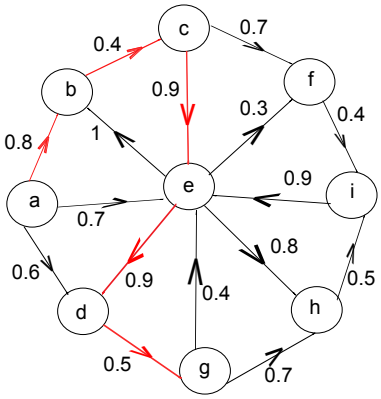


Itération 6 :

j	a	b	c	d	e	f	g	h	i
λ_j	1	0.8	0.32	0.26	0.29	0.09	0.13	0.23	0

$\Gamma^+(g) = \{h\}$

$k = \{g\}$, $S = \{a, b, c, e, d, g\}$, $\overline{S} = \{f, h, i\}$ et $\lambda_g = 0.26 * 0.5 = 0.13$

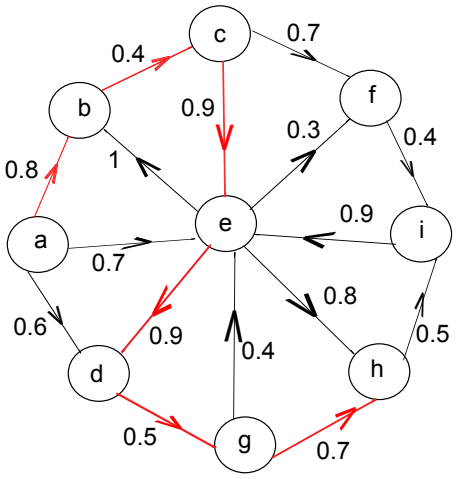


Itération 7 :

j	a	b	c	d	e	f	g	h	i
λ_j	1	0.8	0.32	0.26	0.29	0.09	0.13	0.09	0

$\Gamma^+(h) = \{i\}$

$k = \{h\}$, $S = \{a, b, c, e, d, g, h\}$, $\overline{S} = \{f, i\}$ et $\lambda_h = 0.13 * 0.7 = 0.09$

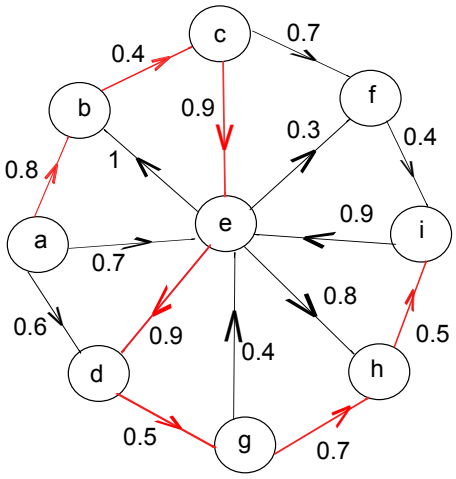


Itération 8 :

j	a	b	c	d	e	f	g	h	i
λ_j	1	0.8	0.32	0.26	0.29	0.224	0.13	0.09	0.045

$\Gamma^+(i) = \{\emptyset\}$

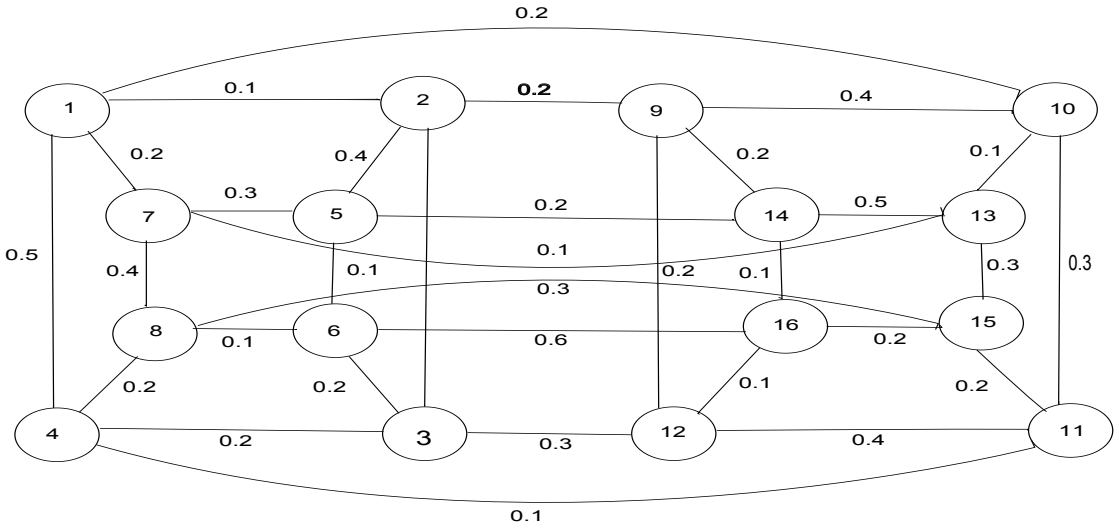
$k = \{i\}$, $S = \{a, b, c, e, d, g, i\}$, $\overline{S} = \{f\}$ et $\lambda_i = 0.09 * 0.5 = 0.045$



Le chemin le plus fiable pour relier le serveur a à un utilisateur i est : $S = \{a, b, c, e, d, g, h, i\}$

3.2 Résolution d'un problème de l'arbre couvrant probabiliste de poids minimum :

Soit G = (X ;A) un réseau de communication de 16 personnes, tel que P_{ij} représente la probabilité d'interception des messages confidentiel entre i et j par une personne étrangère.
Cherchons un arbre couvrant qui minimise la probabilité de l'interception, en utilisant l'algorithme de PRIM.
La situation est représentée par le graphe suivant :



Initialisation

$S = \{\emptyset\}$.

$\overline{S} = \{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16\}$.

1^{re} itération :

On prend le sommet 1 comme un sommet de départ.

$Min\{P(1,2), P(1,4), P(1,7), P(1,10)\} = \{P(1,2)\}$

Donc on ajoute l'arrête $a_1 = (1,2)$ au graphe

Soit : $S = S \cup \{2\} = \{(1,2)\}$

$\overline{S} = \{3,4,5,6,7,8,9,10,11,12,13,14,15,16\}$.

$2^{éme}$ itération :

$Min\{P(1,4), P(1,7), P(1,10), p(2,3), p(2,5), p(2,9)\} = \{P(2,9)\}$

Donc on ajoute l'arrête $a_2 = (2,9)$ au graphe

Soit : $S = S \cup \{9\} = \{(1,2,9)\}$

$\overline{S} = \{3,4,5,6,7,8,10,11,12,13,14,15,16\}$.

$3^{éme}$ itération :

$Min\{P(1,4), P(1,7), P(1,10), p(2,3), p(2,5), P(9,10), P(9,12), P(9,14),\} = \{P(9,14)\}$

Donc on ajoute l'arrête $a_3 = (9,14)$ au graphe

Soit : $S = S \cup \{14\} = \{(1,2,9,14)\}$

$\overline{S} = \{3,4,5,6,7,8,10,11,12,13,15,16\}$.

$4^{éme}$ itération :

$Min\{P(1,4), P(1,7), P(1,10), p(2,3), p(2,5), P(9,10), P(9,12), p(14,5), p(14,13), p(14,16)\} = \{P(14,16)\}$

Donc on ajoute l'arrête $a_5 = (14,16)$ au graphe

Soit : $S = S \cup \{16\} = \{(1,2,9,14,16)\}$

$\overline{S} = \{3,4,5,6,7,8,10,11,12,13,15\}$.

$5^{éme}$ itération :

$Min\{P(1,4), P(1,7), P(1,10), p(2,3), p(2,5), P(9,10), P(9,12), p(14,5), p(14,13), (16,6), (16,12)$
$, (16,15)\} = \{P(16,12)\}$

Donc on ajoute l'arrête $a_6 = (16,12)$ au graphe

Soit : $S = S \cup \{12\} = \{(1,2,9,14,16,12)\}$

$\overline{S} = \{3,4,5,6,7,8,10,11,13,15\}$.

$6^{éme}$ itération :

$Min\{P(1,4), P(1,7), P(1,10), p(2,3), p(2,5), P(9,10), P(9,12), p(14,5), p(14,13), (16,6), (16,15),$
$p(12,3), p(12,9), p(12,11)\} = \{P(1,7)\}$

Donc on ajoute l'arrête $a_7 = (1,7)$ au graphe

Soit : $S = S \cup \{7\} = \{(1,2,9,14,16,12,7)\}$

$\overline{S} = \{3,4,5,6,8,10,11,13,15\}$.

$7^{éme}$ itération :

$Min\{P(1,4), P(1,10), P(2,3), P(2,5), P(9,10), P(9,12), P(14,5), P(14,13), P(16,6), P(16,15)$
$, P(12,3), P(12,11), P(7,5), P(7,8), P(7,13)\} = \{P(7,13)\}$

Donc on ajoute l'arrête $a_8 = (7,13)$ au graphe

Soit : $S = S \cup \{13\} = \{(1,2,9,14,16,12,7,13)\}$

$\overline{S} = \{3,4,5,6,8,10,11,15\}$.

$8^{éme}$ itération :

$Min\{P(1,4), P(1,10), P(2,3), P(2,5), P(9,10), P(9,12), P(14,5), P(14,13), P(16,6), P(16,15),$
$P(12,3), P(12,11), P(7,5), P(7,8), P(13,10), P(13,15)\} = \{P(13,10)\}$

Donc on ajoute l'arrête $a_9 = (13,10)$ au graphe

Soit : $S = S \cup \{10\} = \{(1,2,9,14,16,12,7,13,10)\}$

$\overline{S} = \{3,4,5,6,8,11,15\}$.

$9^{éme}$ itération :

$Min\{P(1,4),P(1,10),P(2,3),P(2,5),P(9,10),P(9,12),P(14,5),P(14,13),P(16,6),P(16,15),$
$P(12,3),P(12,11),P(7,5),P(7,8),P(13,14),P(13,15),P(10,11)\} = \{P(10,11)\}$
Donc on ajoute l'arrête $a_9 = (10,11)$ au graphe
Soit : $S = S \cup \{11\} = \{(1,2,9,14,16,12,7,13,10,11)\}$
$\overline{S} = \{3,4,5,6,8,15\}$.

$10^{éme}$ itération :

$Min\{P(1,4),P(1,10),P(2,3),P(2,5),P(9,10),P(9,12),P(14,5),P(14,13),P(16,6),P(16,15),$
$P(12,3),P(12,11),P(7,5),P(7,8),P(13,14),P(13,15),P(10,9),P(11,4),P(11,12),P(11,15)\} = \{P(11,4)\}$
Donc on ajoute l'arrête $a_{10} = (11,4)$ au graphe
Soit : $S = S \cup \{4\} = \{(1,2,9,14,16,12,7,13,10,11,4)\}$
$\overline{S} = \{3,5,6,8,15\}$.

$11^{éme}$ itération :

$Min\{P(1,4),P(1,10),P(2,3),P(2,5),P(9,10),P(9,12),P(14,5),P(14,13),P(16,6),P(16,15),$
$P(12,3),P(12,11),P(7,5),P(7,8),P(13,14),P(13,15),P(11,15),P(4,8),P(4,3)\} = \{P(4,8)\}$
Donc on ajoute l'arrête $a_{11} = (4,8)$ au graphe
Soit : $S = S \cup \{8\} = \{(1,2,9,14,16,12,7,13,10,11,4,8)\}$
$\overline{S} = \{3,5,6,15\}$.

$12^{éme}$ itération :

$Min\{P(1,4),P(1,10),P(2,3),P(2,5),P(9,10),P(9,12),P(14,5),P(14,13),P(16,6),P(16,15),$
$P(12,3),P(12,11),P(7,5),P(7,8),P(13,14),P(13,15),P(11,15),P(4,3),P(8,6),P(8,15)\} =$
$\{P(8,6)\}$
Donc on ajoute l'arrête $a_{12} = (8,6)$ au graphe
Soit : $S = S \cup \{6\} = \{(1,2,9,14,16,12,7,13,10,11,4,8,6)\}$
$\overline{S} = \{3,5,15\}$.

$13^{éme}$ itération :

$Min\{(P(1,4),P(1,10),P(2,3),P(2,5),P(9,10),P(9,12),P(14,5),P(14,13),P(16,6),P(16,15),$
$P(12,3),P(12,11),P(7,5),P(7,8),P(13,14),P(13,15),P(11,15),P(4,3),P(8,15),P(6,3),P(6,5)\} =$
$\{P(6,5)\}$
Donc on ajoute l'arrête $a_{13} = (6,5)$ au graphe
Soit : $S = S \cup \{6\} = \{(1,2,9,14,16,12,7,13,10,11,4,8,6,5)\}$

$\overline{S} = \{3, 15\}$.

$14^{éme}$ itération :

$Min\{P(1,4), P(1,10), P(2,3), P(2,5), P(9,10), P(9,12), P(14,5), P(14,13), P(16,6), P(16,15),$
$P(12,3), P(12,11), P(7,5), P(7,8), P(13,14), P(13,15), P(11,15), P(4,3), P(8,15), P(6,3), P(6,16)\} = \{P(6,3)\}$

Donc on ajoute l'arrête $a_{14} = (6,3)$ au graphe

Soit : $S = S \cup \{6\} = \{(1,2,9,14,16,12,7,13,10,11,4,8,6,5,3)\}$

$\overline{S} = \{15\}$.

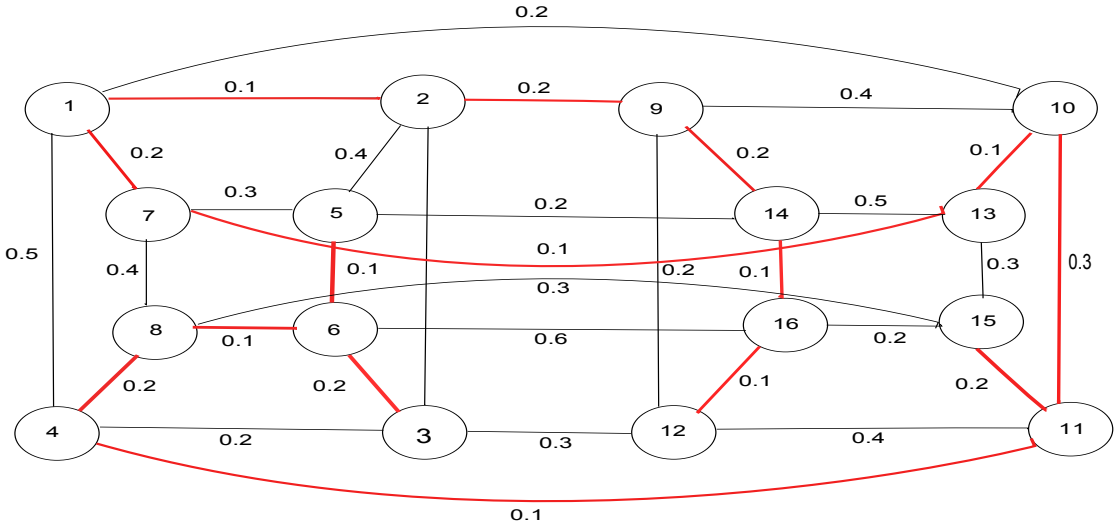
$15^{éme}$ itération :

$Min\{P(1,4), P(1,10), P(2,3), P(2,5), P(9,10), P(9,12), P(14,5), P(14,13), P(16,6), P(16,15),$
$P(12,3), P(12,11), P(7,5), P(7,8), P(13,15), P(11,12), P(11,15), P(4,3), P(8,15)\} = \{P(11,15)\}$

Donc on ajoute l'arrête $a_{15} = (11,15)$ au graphe

Soit : $S = S \cup \{6\} = \{(1,2,9,14,16,12,7,13,10,11,4,8,6,5,3,15)\}$

$\overline{S} = \{\emptyset\}$.



3.3 Résolution d'un problème d'ordonnancement :

La rénovation du séjour d'un appartement se décompose en plusieurs tâches décrites dans le tableau ci dessous. Ce dernier donne également les précédences à respecter lors de la planification des travaux ainsi qu'une estimation de la durée de chacune des tâches.

Rubrique	Tâches	Description	Durée	Tâches préc
	Début	Lancement du projet	0	-
	A	Dépose ancien carrelage	6	J
	B	Pose carrelage	4	A
	C	Joints carrelage	2	B
Murs	D	Décollage ancien papier	8	J
	E	Pose faïence	6	D
	F	Pose nouveau papier ancien papier	4	E
Plomberie	G	Dépose ancien évier	1	Début
	H	Déplacement arriv et évac	6	G
	I	Pose et raccordement evier	2	M
Mobilier	J	Dépose ancien éléments	4	G
	K	Assemb. caissons et tiroirs	8	Début
	L	Pose éléments bas	6	B,J,H,K
	M	Pose plan de travail	4	L
	N	Etanchéité plan de travail	1	E, M
	O	Pose éléments hauts	1	E, J
	Fin	Fin du projet	0	C,F,I,N,O

Le diagramme de Gantt associé à ce tableau est donné dans la figure suivante :

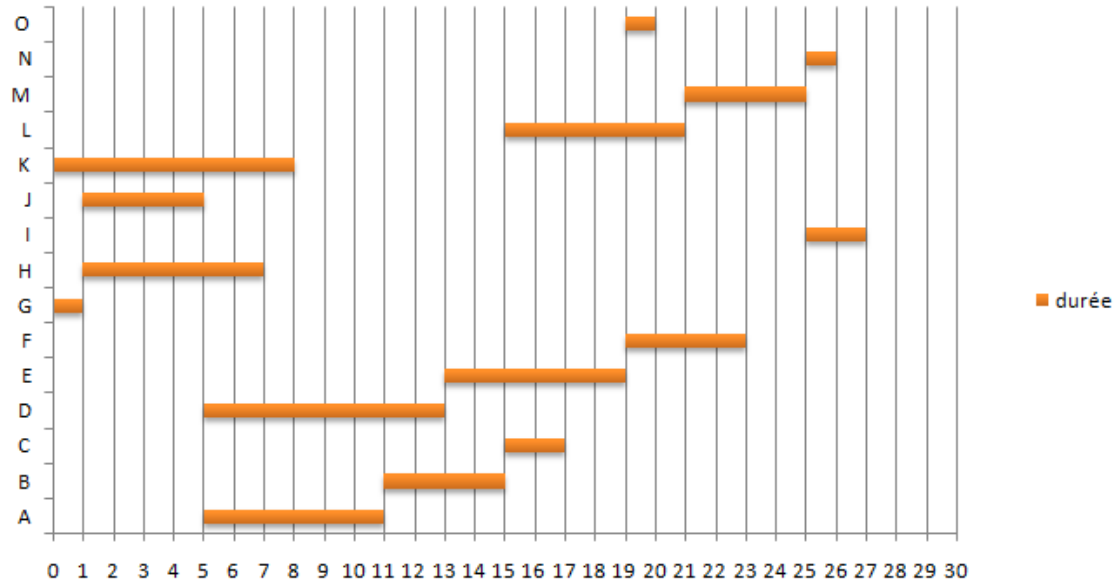


FIGURE 3.1 – Diagramme de Gantt

On peut modéliser le tableau des tâches sous forme d'un réseau pert potentiels tâches, tel que les tâches sont représentées par des sommets et les contraintes de potentiels sont représentées par des arcs. Le graphe potentiels-tâches correspondant est donné dans la figure qui suit :

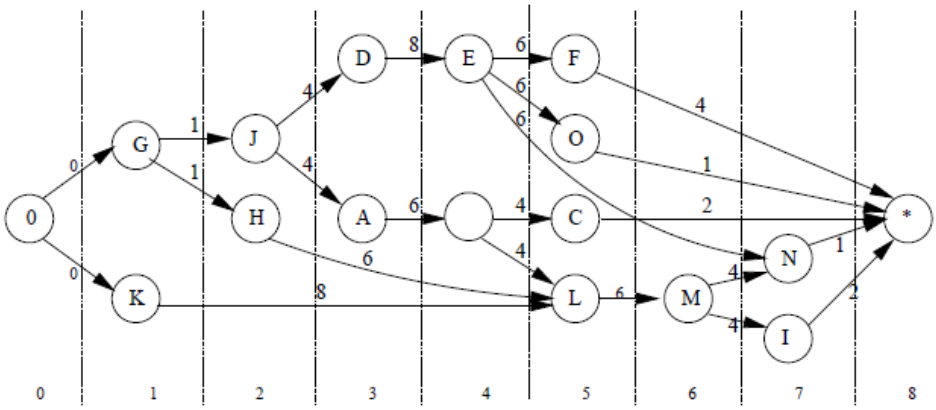


FIGURE 3.2 – Réseaux Pert

Dates de début au plus tôt :

On les calcule de la manière suivante :

$$\underline{t_i} = \max_{j \in \Gamma^-(i)} (\underline{t_j} + C_{ji})$$

$C_{j,i}$ est la Capacité de l'arc (j, i) (par exemple la durée de j), $\Gamma^-(i)$ est l'ensemble des prédécesseurs de i.

donc :

$\underline{t_G} = max(\underline{t_0} + C_{0,G}) = 0 + 0 = 0$

$\underline{t_k} = max(\underline{t_0} + C_{0,k}) = 0 + 0 = 0$

$\underline{t_j} = max(\underline{t_G} + C_{G,J}) = 0 + 1 = 1$

$\underline{t_H} = max(\underline{t_G} + C_{G,H}) = 0 + 1 = 1$

$\underline{t_D} = max(\underline{t_J} + C_{J,D}) = 1 + 4 = 5$

$\underline{t_A} = max(\underline{t_J} + C_{J,A}) = 1 + 4 = 5$

$\underline{t_E} = max(\underline{t_D} + C_{D,E}) = 5 + 8 = 13$

$\underline{t_B} = max(\underline{t_A} + C_{A,B}) = 5 + 6 = 11$

$\underline{t_F} = max(\underline{t_E} + C_{E,F}) = 13 + 6 = 19$

$\underline{t_O} = max(\underline{t_E} + C_{E,O}) = 13 + 6 = 19$

$\underline{t_C} = max(\underline{t_B} + C_{B,C}) = 11 + 4 = 15$

$\underline{t_L} = max(\underline{t_B} + C_{B,L}, \underline{t_H} + v_{H,L}, \underline{t_K} + v_{K,L}) = max(11 + 4, 1 + 6, 0 + 8) = max(15, 7, 8) = 15$

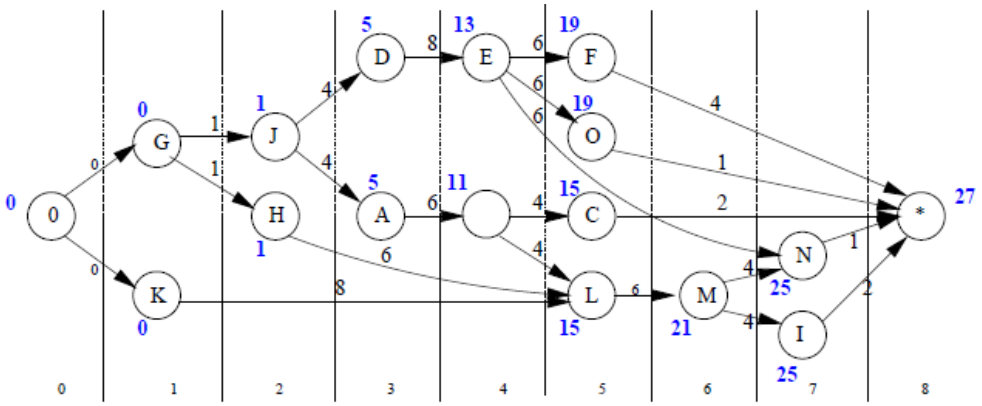
$\underline{t_M} = max(\underline{t_L} + C_{L,M}) = 15 + 6 = 21$

$\underline{t_N} = max(\underline{t_E} + C_{E,N}, \underline{t_M} + C_{M,N}) = max(13 + 6, 21 + 4) = max(19, 25) = 25$

$\underline{t_I} = max(\underline{t_M} + C_{M,I}) = 21 + 4 = 25$

$\underline{t_*} = max(\underline{t_F} + C_{F,*}, \underline{t_O} + C_{O,*}, \underline{t_C} + C_{C,*}, \underline{t_N} + C_{N,*}, \underline{t_I} + C_{I,*},) = max(19 + 4, 19 + 1, 15 + 2, 25 + 1, 25 + 2) =$
$max(23, 20, 17, 26, 27,) = 27$

Les dates de début au plus tôt sont représentées dans le graphe suivant :



Dates de début au plus tard :

On les calcule de la manière suivante :

$$\overline{t_i} = \min_{j \in \Gamma^+(i)} (\overline{t_j} - C_{i,j})$$

$C_{j,i}$ est la capacité de l'arc (j, i) (par exemple la durée de j) $\Gamma^+(i)$ est l'ensemble des successeurs de i.

donc :

$\overline{t_I} = min(\overline{t_*} - C_{I,*}) = 27 - 2 = 25$

$\overline{t_N} = min(\overline{t_*} - C_{N,*}) = 27 - 1 = 26$

$\overline{t_M} = min(\overline{t_M} - C_{N,M}, \overline{t_I} - C_{M,I}) = min(26-4, 25-4) = min(22-21) = 21$

$\overline{t_L} = min(\overline{t_L} - C_{M,M}) = 21 - 6 = 15$

$\overline{t_C} = min(\overline{t_*} - C_{C,*}) = 27 - 2 = 25$

$\overline{t_O} = min(\overline{t_*} - C_{O,*}) = 27 - 1 = 26$

$\overline{t_F} = min(\overline{t_*} - C_{F,*}) = 27 - 4 = 23$

$\overline{t_B} = min(\overline{t_L} - C_{L,B}, \overline{t_C} - C_{B,C}) = min(15-4, 25-4) = 11$

$\overline{t_E} = min(\overline{t_N} - v_{E,N}, \overline{t_O} - C_{E,O}, \overline{t_F} - C_{E,F}) = min(26-6, 26-6, 23-6) = min(20, 20, 17) = 17$

$\overline{t_A} = min(\overline{t_B} - C_{A,B}) = 11 - 6 = 5$

$\overline{t_D} = min(\overline{t_E} - C_{D,E}) = 17 - 8 = 9$

$\overline{t_H} = min(\overline{t_L} - C_{H,L}) = 15 - 6 = 9$

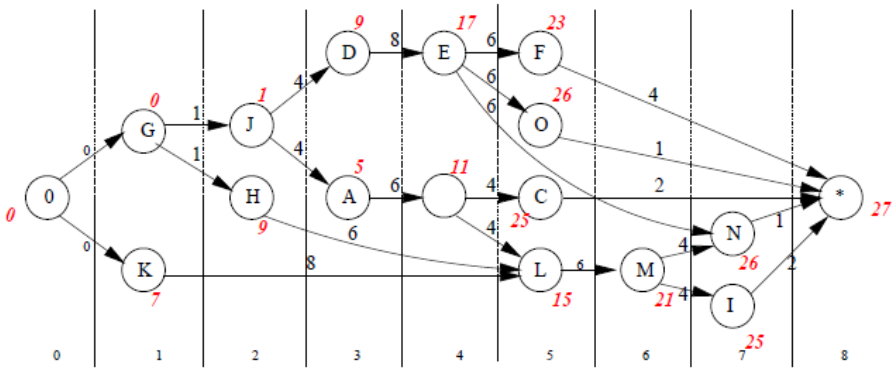
$\overline{t_J} = min(\overline{t_A} - C_{J,A}, \overline{t_D} - C_{J,D}) = min(5-4, 9-4) = min(1, 5) = 1$

$\overline{t_K} = min(\overline{t_L} - C_{K,L}) = 15 - 8 = 7$

$\overline{t_G} = min(\overline{t_H} - C_{G,H}, \overline{t_J} - C_{G,J}) = min(9-1, 1-1) = min(8, 0) = 0$

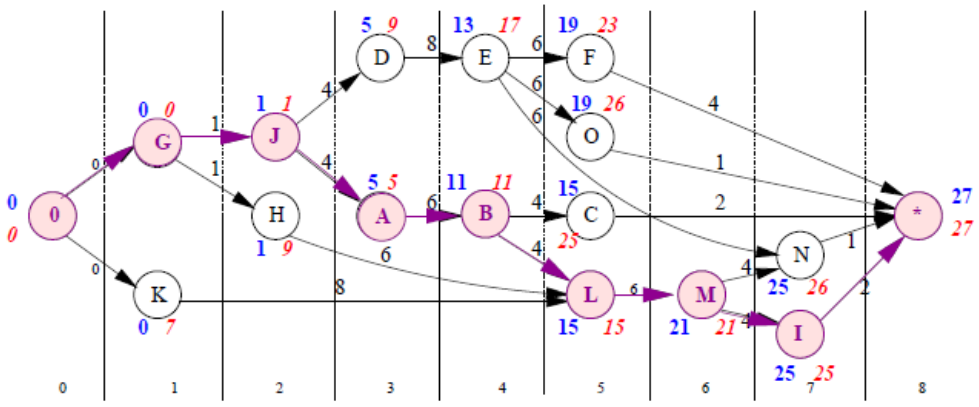
$\overline{t_o} = min(\overline{t_k} - C_{o,k}, \overline{t_g} - C_{o,g}) = min(7-0, 0-0) = min(7, 0) = 0$

Les dates de début au plus tard sont représentées dans le graphe suivant :



Marges totales et chemin critique :

On calcule la marge totale d'une tâche i de la manière suivante : $MT_i = \overline{t_i} - \underline{t_i}$, le chemin critique est l'ensemble des tâches critiques (les tâches ayant une marge nulle) sont données dans la figure qui suit :



Le temps minimum de réalisation de l'ensemble est lisible sur le dernier sommet (*) :**27 jours** .

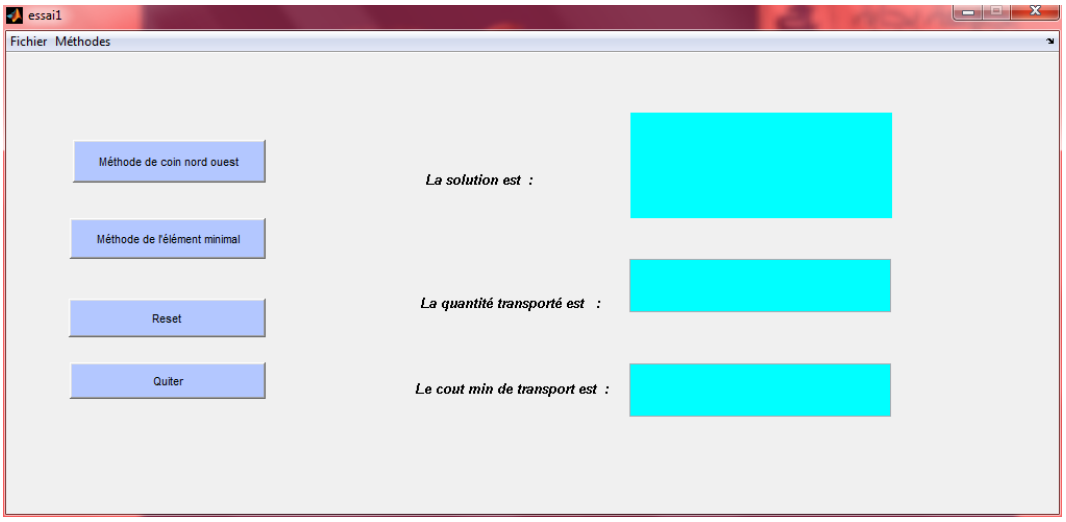
3.4 Résolution d'un problème de transport :

Nous aboutissons maintenant à l'étape finale à savoir l'élaboration d'une application aussi convivial que possible, sans perdre de vue son aspect pratique et encore moins sa performance, muni d'une interface claire et accessible, facilitant ainsi son utilisation.

(a) Présentation de l'application :

L'application nommée **"Optimisation dans les réseaux de transports"** est présentée sous forme d'une interface principale comme le montre la figure :

L'interface contient un bouton nommé **" transport "** qui permet d'accéder à l'application, en cliquant sur le bouton l'interface s'affiche.



Cette interface est composée de :

1.Barre de menus comportant les menus suivants :

- Menu Fichier
- Menu Méthodes

a-Menu Fichier :

Ce menu comporte trois sous-menus :

- **Ouvrir nouveau :** créer un nouveau réseau de transport
- **Enregistrer :** enregistrer les résultats trouvés dans un fichier texte
- **Quitter :** pour fermer l'application

b- Menu Méthodes.

Ce menu comporte deux sous-menus :

- Méthode de coin nord ouest
- Méthode de l'élément minimal

2. Quatre patches boutons :

- **Méthode de coin nord ouest :** pour résoudre le problème avec la méthode coin nord ouest
- **Méthode de l'élément minimal :** pour résoudre le problème avec la méthode de l'élément minimal
- **Reset :** pour actualiser l'interface
- **Quitter :** pour quitter l'application

3. Trois édits boutons

- **La solution est :** pour afficher la matrice des coûts optimaux.
- **La quantité transporté est :**pour afficher la quantité optimal du transport.
- **Le coût minimum de transport est :** pour afficher le coût minimum de transport.

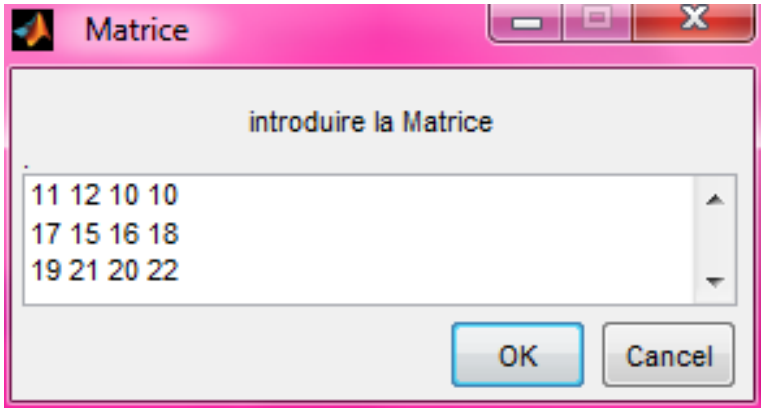
(b) Exemple d'application :

On applique notre programme sur l'exemple suivant :

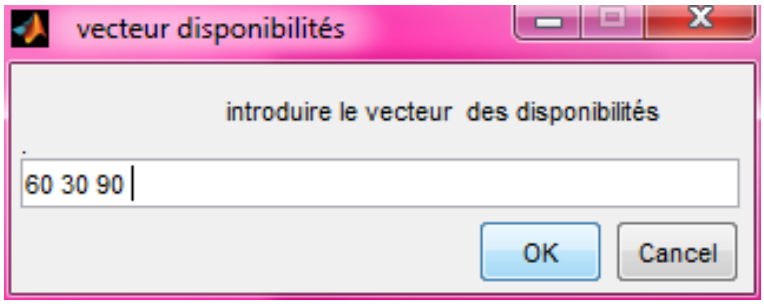
	Client 1	Client 2	Client 3	Client 4	Disponibilité
Usine 1	11	12	10	10	60
Usine 2	17	16	15	18	30
Usine 3	19	21	20	22	90
Demande	50	75	30	25	180

- D'abord on choisit la méthode avec laquelle on veut résoudre notre problème en cliquant sur l'un des boutons(Méthode de coin nord ouest ou Méthode de l'élément minimal)

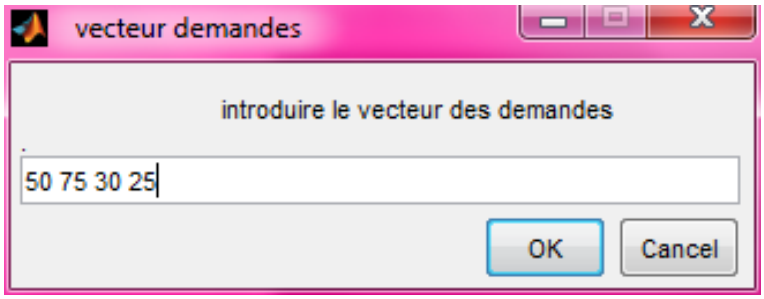
- Lorsque l'utilisateur clique sur l'un des boutons, une nouvelle interface s'affiche où il introduit la matrice des coûts de transport



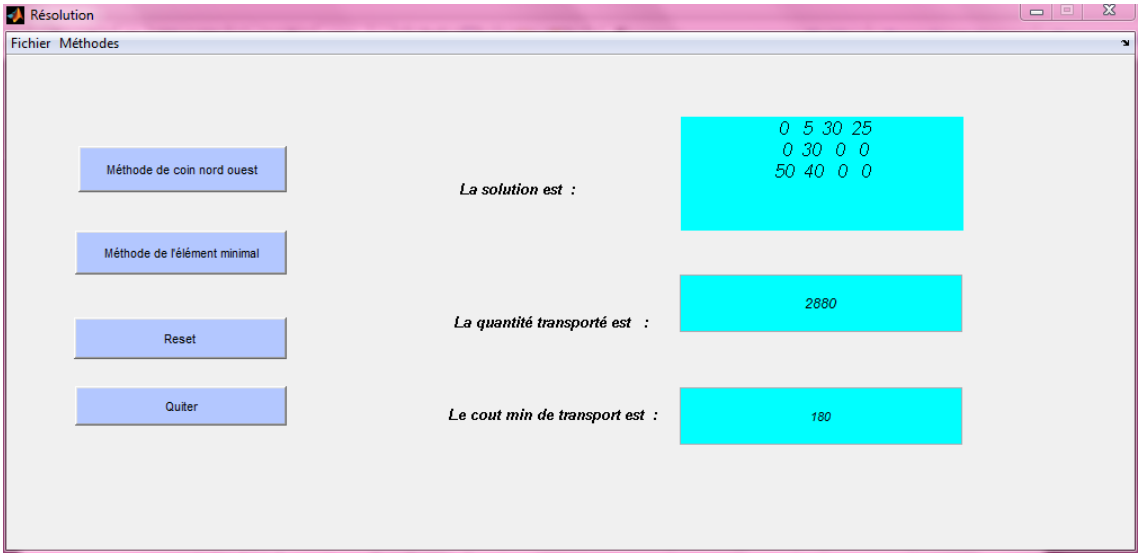
- En cliquant sur le bouton " OK ", une autre interface s'affiche ou il introduit le vecteur des disponibilité



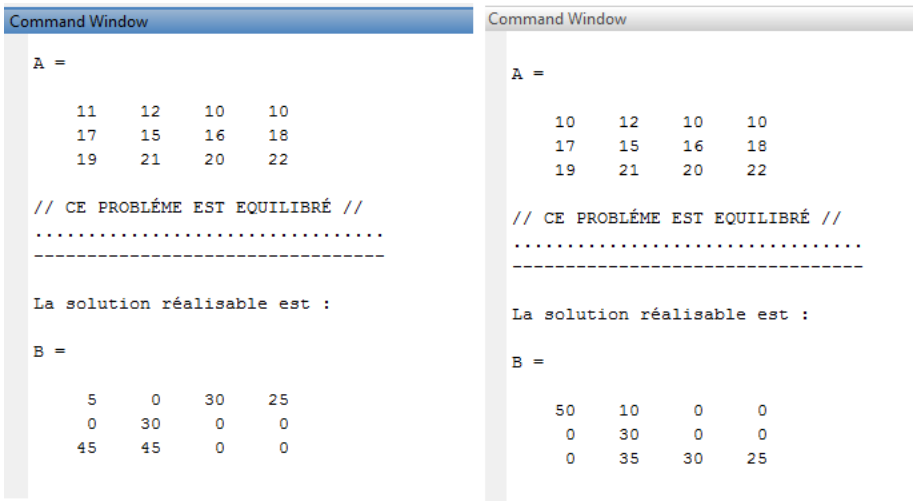
– En cliquant sur le bouton ” OK ”, une nouvelle interface s’affiche ou il introduit le vecteur des demandes.



– En cliquant sur le bouton ” OK ”, une nouvelle interface s’affiche dans laquelle la solution optimal est affiché.



– En allant vers la commande MATLAB on aperçoit que les différents résultats trouvés dans chaque itération sont affichés, on remarque aussi que la solution de basse réalisable diffère dans les deux méthodes, ce qui implique la différence de nombres d’itération par contre on aura toujours une unique solution optimal.



Conclusion

Dans ce chapitre on a présenté trois problèmes concrets d'optimisation dans les réseaux, dans les deux premiers on a résolu un problème de télécommunication et un problème d'ordonnancement et dans le dernier on a réalisé une application qui permet de résoudre un problème de transport avec deux méthodes différentes : méthode coin nord ouest et méthode de l'élément minimal en utilisant **MATLAB** et le constructeur d'interface graphique **GUIDE**.

Conclusion générale

Dans ce travail, nous nous sommes intéressés au problème d'optimisation dans les réseaux en utilisant les graphes valués.
On a pu constater que les graphes constituent une méthode de pensée qui permet de modéliser une grande variété de problèmes concrets en se ramenant à l'étude de sommets et d'arcs.
La théorie des graphes permet de générer des circuits optimisés et de gérer des réseaux (routiers, de communication, de transport, d'eau de gaz, ...), d'ordonnancer des tâches et de gérer des plannings. Elle est la clé de l'intelligence artificielle avec la notion du " plus court chemin ".
Ces nombreuses applications font de la théorie des graphes un outil appréciable d'aide à la décision (en recherche opérationnelle), en apparence, sa mise en œuvre est simple et ludique, voire enfantine.
Notre travail consiste à donner aux lecteurs un certain nombre de méthode et d'algorithme pour résoudre des problèmes d'optimisation qu'ils peuvent rencontré dans la vie de tous les jours.
Pour cela on a résolu quatre problème concrets, le premier consiste à résoudre un problème de télécommunication en prenant compte de la fiabilité de communication en utilisant l'algorithme de Dijkstra pour un chemin fiable.
Pour le deuxième, on a considéré un réseau de communication avec une probabilité d'interception des messages confidentiels. Pour déterminer un arbre couvrant de poids minimum, pour cela on a modélisé ce problème à l'aide d'un graphe probabiliste, et on l'a résolu avec l'algorithme de Prim.
Le troisième problème consiste à inaugurer un appartement en minimisant le délai des travaux, pour cela on a modélisé le problème sous forme d'un réseau PERT.

Enfin, le dernier problème consiste à élaborer une application pour la résolution d'un problème de transport de telle sorte qu'elle soit aussi convivial que possible, sans perdre de vue son aspect pratique et encore moins sa performance, muni d'une interface claire et accessible, facilitant ainsi son utilisation, la résolution de ce problème s'est fait par deux méthodes différentes : méthode coin nord ouest et méthode de l'élément minimal en utilisant **MATLAB** et le constructeur d'interface graphique **GUIDE**.

Annexe

Environnement de programmation :
Une description de l'environnement de programmation utilisé s'avère nécessaire.

MATLAB : est un logiciel qui permet de façon interactive et graphique de modéliser et de simuler des systèmes. Matlab s'impose dans les mondes universitaires et industriels comme un outil puissant de simulation et de visualisation de problèmes numériques [12].

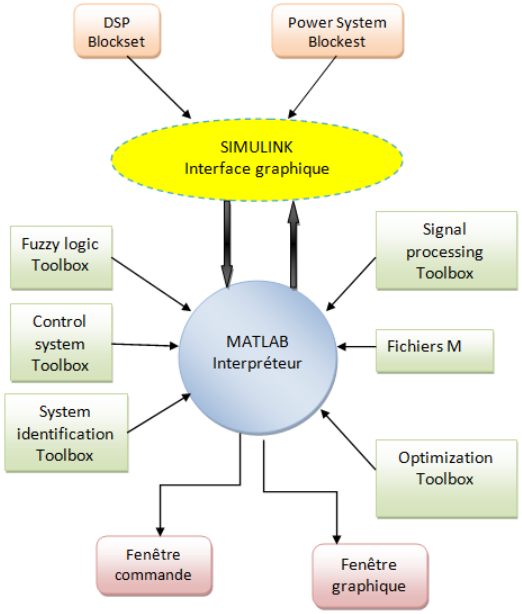


FIGURE 3.3 – Environnement de Matlab. [13]

Fenêtre commande : Dans cette fenêtre, l'usager donne les instructions et MATLAB retourne les résultats.
Fenêtre graphique : MATLAB trace les graphiques dans ces fenêtres.
Fichiers M : Ce sont des programmes en language matlab (écrit par l'usager).
Toolboxes : Ce sont des collections de fichiers M développés pour des domaines d'application spécifiques (Signal Processing Toolbox, System identification Toolbox, Control system Toolbox, u-Synthesis and Analysis Toolbox, etc.)
Simulink : C'est l'extension graphique de MATLAB permettant de travailler avec des diagrammes en blocs. **Blocksets :** Ce sont des collections de blocs Simulink développés pour des domaines d'application spécifiques (DSP Blockset, Power System Blockset, etc.)

En effet MATLAB possède le **GUIDE** (pour Graphical User Interface) c'est un constructeur d'interface graphique qui regroupe tous les outils dont le programmeur a besoin pour créer une

interface graphique. Il s'ouvre soit en cliquant sur l'icône approprier soit en tapant guide dans la commande Windows de MATLAB.
Le **GUIDE** permet à l'utilisateur d'interagir avec un programme informatique, grâce à différents objets graphiques (boutons, menus, cases à cocher...). Ces objets sont généralement actionnés à l'aide de la souris ou du clavier.

Malgré le fait que les interfaces graphiques semblent secondaires par rapport au développement du coeur d'une application, elles doivent néanmoins être conçues et développées avec soin et rigueur.
Leur efficacité et leur ergonomie sont essentielles dans l'acceptation et l'utilisation de ces outils par les utilisateurs finaux.
Une bonne conception et un développement maîtrisé permettent également d'en assurer une meilleure maintenabilité.

Bibliographie

[1] Claude. Berge, ≪*Graphes et hypergraphes*≫, Dunod Editions, Paris 1970.

[2] Didier. Muller, ≪*Introduction à la théorie des graphes*≫, City Editions, France 2007.

[3] Philippe. Compoint, ≪*Les Graphes en recherche opérationnelle*≫, Dunod Editions, Paris 1972.

[4] Christophe. Rossignol, ≪*Graphes pondéré, étiquetés, probabilistes*≫, Grenoble 1989.

[5] Michel. Gandron et Michel Minoux, ≪*Graphes et algorithmes*≫, Editions Eyrolles, Paris VI 1986 .

[6] Wajdi. Tekaya, ≪*Probléme d'arbre couvrant de poinds minimum*≫, these de doctorat, Université Paris dauphine.

[7] Michel. Sakarovitch, ≪*Graphes et programmation linéaire*≫, Editions Hermann, France, 08/09/1988.

[8] Jacques. Labelle, ≪*Théorie des graphe*≫, Modulo Éditeur, Québec 1981.

[9] G. Fontan, ≪*Combinatoire et Ordonnancement*≫, Edition Masson, Toulouse 1987.

[10] Mohamed Ali. Aloulou, ≪*Introduction aux problèmes d'ordonnancement*≫, Dunod Editions, Paris Dauphine,28 novembre 2005.

[11] Philippe. Mahey, ≪*La programmation linéaire*≫, Editions Technip, France 1967.

[12] Hoang. Le-Huy, ≪*Introduction à MATLAB et Simulink*≫, Université Laval, Québec, Canada 1998.

[13] Jérôme. Briot, ≪*Introduction à la programmation des interfaces graphique*≫, matlab.developpez.com/tutoriels/, 11/06/2007.

Résumé

L'objectif de ce travail consiste à donner les différentes méthodes et outils pour résoudre des problèmes d'optimisation relevant de la théorie des graphes rencontrés dans le quotidien de chacun. Les deux premiers problèmes ont comme champs d'application les télécommunication dont l'une des applications concerne le problème de la fiabilité des communications qu'on a pu résoudre grâce à l'utilisation de l'algorithme de Dijskra. Quand à l'autre, elle pose le problème de la fiabilité d'interception des messages confidentiels dont la résolution a pu se faire grâce à l'algorithme de PRIM. Le troisième problème consiste à minimiser le délai des travaux d'aménagement d'un appartement qu'on a modéliser sous forme d'un réseau de PERT. Pour finir, le dernier problème concerne l'optimisation des coûts de transport tout en maximisant la quantité transportée à été résolu grâce à deux méthodes différentes : méthode du coin nord ouest et la méthode de l'élément minimal en utilisant MATLAB et le constructeur d'interface graphique GUIDE.

Mots-clés : Graphe, optimisation, algorithme de Dijkstra, problème de Maria Hasse,....

Abstract

The aim of this work is to give different methods and tools to solve optimization problems concerned with the graph theory met in the daily life of each one. Both of the two first problems have as fields of application the telecommunication, the first one relates to the problem of the reliability of communication that has been solved by the use of Dijskra algorithm. The second one raises the issue of the reliability of the interception of confidential messages which its resolution could be done thanks to the PRIM algorithm. The third problem is to minimize the delay in the set up of an apartment which have been modeled as a PERT network. Finally, the last problem relates to the optimization of transport costs while maximizing the amount transported was solved by two different methods : Coin nord ouest method and the minimal element method by using MATLAB and the graphical interface builder GUIDE.

Keywords : Graph, optimization, algorithm of Dijkstra, problem of Maria Hasse,....