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

Université A. Mira de Bejaia
Faculté des Sciences Exactes
Département de Recherche Opérationnelle

MÉMOIRE DE FIN DE CYCLE

En
Recherche Opérationnelle

Option
Modélisation Mathématique et Techniques de la Décision

Thème

Quelques Algorithmes d'optimisation et Application dans les graphes

Présenté par : M. DJALIL Mourad

Devant le jury composé de :

Présidente	**Mme L. YOUNSI**	**Maître Assistant. A**	**U. A/Mira Bejaia.**
Examinateur	**M. S. TAOUINET**	**Maître Assistant. A**	**U. A/Mira Bejaia.**
Encadreur	**M. K. KABYL**	**Maître de conf. B**	**U. A/Mira Bejaia.**

Bejaia, 2018-2019

⁕ *Remerciements* ⁕

Je remercie avant tout, Dieu tout-puissant qui m'a donné la force, le courage et la volonté pour réaliser ce travail.

Un grand merci à ma famille pour leur présence, leur préoccupation et le souci qu'ils se sont fait pour moi, leur encouragement et leur suivi avec patience le long de la réalisation de notre projet.

Je suis très reconnaissant envers M. K. KABYL mon promoteur pour l'honneur qu'il nous a fait en assurant la direction du présent mémoire. Nous le remercions pour ses précieux conseils et orientations.

Je tiens également à remercier les membres de jury qui m'ont fait l'honneur de juger ce travail.

Enfin, je remercie de tous cœur tous ceux qui ont contribué de près ou de loin à la réalisation de ce mémoire.

⁕ *Dédicaces* ⁕

Je dédié ce modeste travail :

A mes chers parents ;

A mes frères et ma sœur ;

A toute la famille ;

A mes amis et collègues et, tous ceux qui m'a aidé ;

A toute la famille RO.

M. ***DJALIL Mourad***

Table des matières

Table des figures

Introduction générale

La théorie des graphes constitue aujourd'hui un corpus de connaissances très important, historiquement elle est née en 1736 avec la communication d'Euler (1707-1783) dans laquelle il proposait une solution au célèbre problème des ponts de Königsberg (Euler, 1736). " Les habitants de Königsberg se demandaient s'il était possible, en partant d'un quartier quelconque de la ville, de traverser tous les ponts sans passer deux fois par le même et de revenir à leur point de départ. " Euler démontra que ce problème n'a pas de solution. Le problème des ponts de Königsberg est identique à celui consistant à tracer une figure géométrique sans lever le crayon et sans repasser plusieurs fois sur un même trait.

Des années plus tard des recherches ont été faites dans ce domaine notamment par Kirchhoff (1824-1887) ou il développa la théorie des arbres pour l'appliquer à l'analyse de circuits électriques. De nombreux problèmes intéressent la théorie des graphes tels que la conjecture des quatre couleurs et le casse-tête de Hamilton 1859 (recherche d'un chemin Hamiltonien).

A partir de 1946, la théorie des graphes a connu un développement intense sous l'impulsion de chercheurs motivés par la résolution de problèmes concrets. Parmi ceux-ci, citons de manière privilégiée Kuhn (1955), Ford et Fulkerson (1956) et Roy (1959). Parallèlement, un important effort de synthèse a été opéré en particulier par Claude Berge. Son ouvrage "Théorie des graphes et ses applications" publié en 1958 (Berge, 1958) marque sans doute l'avènement de l'ère moderne de la Liste des figures théorie des graphes par l'introduction d'une théorie des graphes unifiée et abstraite rassemblant de nombreux résultats épars dans la littérature. Parmi ses problèmes il y a le problème de réseaux, réseaux de transport et problèmes de flots, problème de cheminement, problème d'ordonnancement...etc

Nous nous intéressons dans notre travail à la définition ainsi que "la programmation de quelques méthodes d'optimisation dans les graphes ", dont le but est de résoudre un problème réel en utilisant la théorie des graphes et plus particulièrement l'optimisation . Et pour cela, on a réparti notre mémoire en quatre chapitres, une conclusion et une introduction :

Dans le premier chapitre, on a donné les éléments de la théorie des graphes en présentant quelques définitions et concepts de base sur les graphes.

Le deuxième chapitre consiste à la présentation de quelques problème de la théorie des graphes.

Puis le chapitre trois consiste à déterminer les méthodes adéquates pour la résolution des problèmes posés au deuxième chapitre illustrées par des applications appropriées.

Dans le dernier chapitre, nous abordons l'application de trois problèmes concrets qui résument le but de notre mémoire à savoir les techniques d'optimisation dans les graphes programmés sous Matlab. Nous présentons le premier c'est l'ordonnancement des tâches d'un projet de construction d'un maison, puis le réseau routier en Algérie, En finira par le problème de canalisation d'eau dans trois villes.

Chapitre 1

Définitions de base et notations

1.1 Introduction

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

1.2 Concept des graphes :

Définition 1.2.1. Un graphe $G = (X, E)$ est constitué d'un ensemble fini $X = \{x_1, x_2, \ldots, x_n\}$ appelés sommets avec $|X| = n$ et d'une famille $E = \{e_1, e_2, \ldots, e_m\}$ de paires distincts de X appelés arêtes (arcs) avec $|E| = m$.
Le nombre de sommets du graphe G est appelé ordre de G qu'on note $O(G)$.[16]

1.2.1 Graphe orienté :

Un graphe orienté est un graphe dont chaque arc (e) tel que $e = (x_i, x_j) \in E$ possède une extrémité initiale $I(e) = x_i$ et une extrémité terminal $T(e) = x_j$. Si $I(e) = T(e)$ on dit que e est une boucle.[16]

Le graphe de la Figure 1.1 possède six sommets et sept arcs qui sont : $X = \{1, 2, 3, 4, 5, 6\}$, $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$, avec $I(e_1) = 2, T(e_1) = 1$.

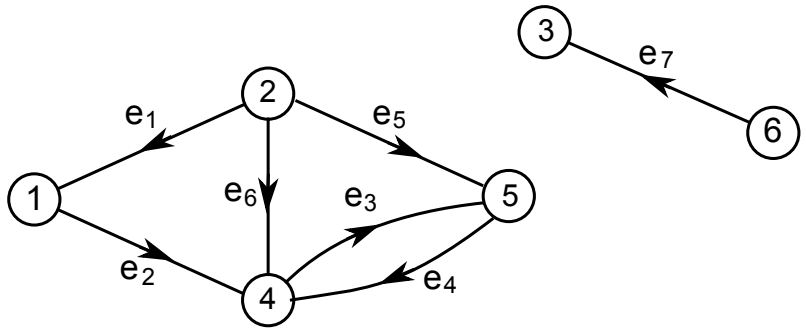


FIGURE 1.1 – Graphe orienté.

1.2.2 Graphe non orienté :

Est un graphe dont les arêtes ne sont pas orientés et chaque arête (e) est définie comme suit : $e = (x_i, x_j) = x_i x_j$. Une arête de type $x_i x_i$ est appelée boucle Dans ce cas on dit que :

- x_i et x_j sont adjacents ;
- (e) est incidente â x_i et x_j ;
- Deux arêtes (arcs) e_i et e_j sont adjacents s'ils ont au moins une extrémité en commun.

Le graphe de la Figure 1.2 représente un graphe non orienté.[16]

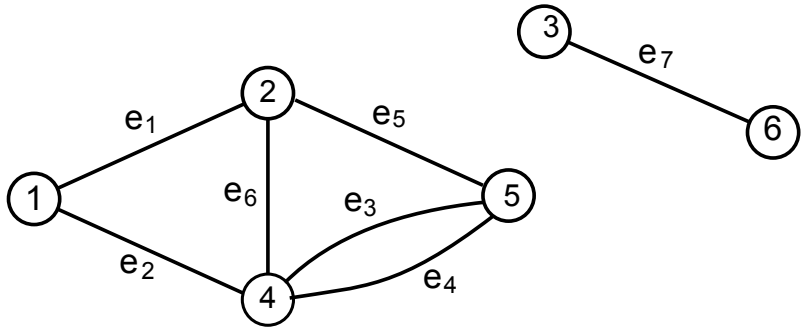


FIGURE 1.2 – Graphe non orienté.

1.2.3 Successeurs, prédécesseurs d'un sommet :

a) L'ensemble des successeurs d'un sommet x_i est l'ensemble de tous les arcs sortants du sommet x_i noté par $\Gamma^+(x_i)$ tel que :
$\Gamma^+(x_i) = \{x_j \in X/(x_j, x_i \in E)\}$,

b) L'ensemble des prédécesseurs d'un sommet x_i est l'ensemble de tous les arcs entrants au sommet x_i noté par $\Gamma^-(x_i)$ tel que :
$\Gamma^-(x_i) = \{x_j \in X/(x_j, x_i \in E)\}$,

– x est un voisin de y si x est un prédécesseur ou x est un successeur qu'on note $\Gamma(x)$ avec :$\Gamma(x) = \Gamma^-(x) \cup \Gamma^+(x)$.[14]

Dans le graphe orienté de la Figure 1.3 on a : $\Gamma^-(4) = \{1, 2\} et \Gamma^+(4) = \{3\}$.

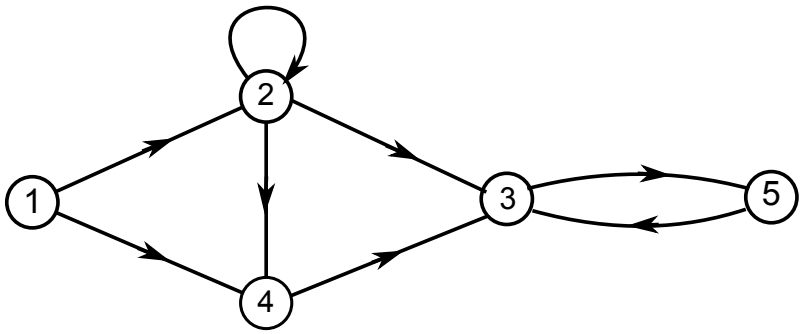


FIGURE 1.3 – Succésseurs, prédecesseurs d'un sommet.

1.2.4 Degré d'un sommet :

Soit $= (X, E)$ un graphe orienté :

– Le demi degré extérieur d'un sommet x : Est égale au nombre d'arcs ayant x comme extrémité initiale (c-à-d les arcs qui sort de x) on le note $d_G^+(x)$ avec : $d_G^+(x) = |\{e \in E/I(e) = x\}|$.

– Le demi degré intérieur d'un sommet x : Est égale au nombre d'arcs ayant x comme extrémité terminal (c-à-d les arcs entrant au x) on le note $d_G^-(x)$ avec : $d_G^-(x) = |\{e \in E/T(e) = x\}|$.

– Le degré d'un sommet x : C'est le nombre d'arcs ayant x comme extrémité initiale ou terminal, on le note $d_G(x)$ avec : $d_G(x) = d_G^-(x) + d_G^+(x)$.[14]

Proposition 1.2.1. *Pour un graphe non orienté $G = (X, E)$ [14]*

$$\sum_{x \in X} d_G(x) = 2|E|$$

Proposition 1.2.2. *Pour un graphe orienté $G = (X, E)$ [14]*

$$\sum_{x \in X} d_G^+(x) = \sum_{x \in X} d_G^-(x) = |E|$$

1.2.5 Graphe simple :

Est un graphe qui ne possède ni boucles ni arcs (arêtes) multiples.

Le graphe de la Figure 1.4 représente un graphe simple à quatre sommets et cinq arêtes.[7]

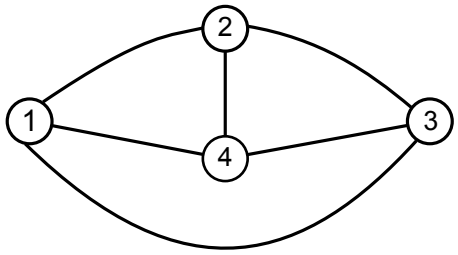


FIGURE 1.4 – Graphe simple.

1.2.6 Graphe complémentaire :

Soit $G = (X, E)$ un graphe simple. Le graphe $\bar{G}= (X, \bar{E})$ est le graphe complémentaire de G si : $\forall e \in E \Leftrightarrow e \notin \bar{E}$.[14]

La Figure 1.5 représente le graphe G ainsi que son graphe complémentaire $\bar{G}$.[14]

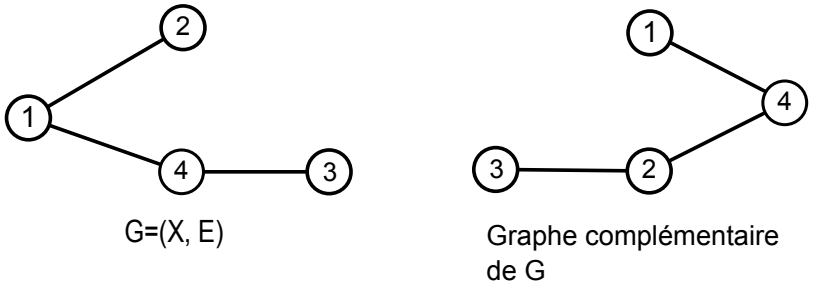


FIGURE 1.5 – Graphe G et son complémentaire.

1.2.7 Graphe multiple :

C'est un graphe qui possède des boucles ou bien des arêtes (arcs) multiples.[14]

Le graphe de la Figure (1.6) représente un graphe multiple.

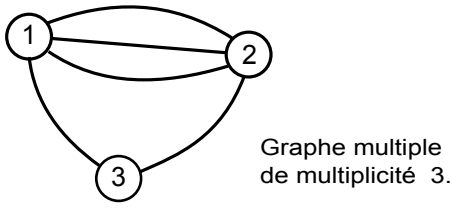


FIGURE 1.6 – Graphe multiple.

1.2.8 Graphe partiel :

Soit $G = (X, E)$ un graphe. Le graphe $G' = (X, E')$ est un graphe partiel de G si E' est inclus dans E. Autrement dit, on obtient G' en enlevant une ou plusieurs arêtes au graphe G.[13]
Le graphe de la Figure (1.7) représente un graphe G et son graphe partiel.

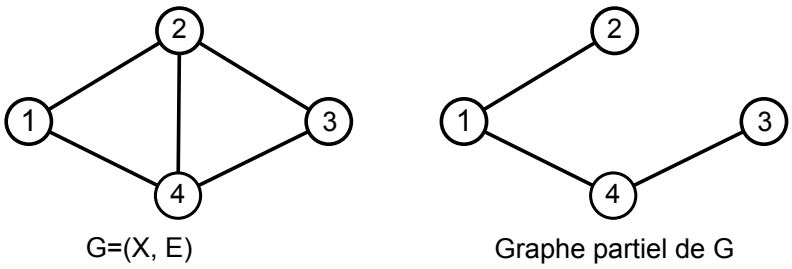


FIGURE 1.7 – Graphe G et l'un de ces graphes partiels.

1.2.9 Sous-graphe :

Soit $G = (X, E)$ un graphe. Soit A un sous ensemble de sommets inclus dans X.

Le sous-graphe $G_A = (X_A, E(A))$ est un graphe qu'est formé de toutes les arêtes de G ayant les deux extrémités dans A.[13]

Soit $G = (X, E)$ le graphe de la Figure 1.8, on prend $A = \{1, 2, 4\}$ et $E(A) = \{e_1, e_2, e_6\}$ le sous-graphe de G.

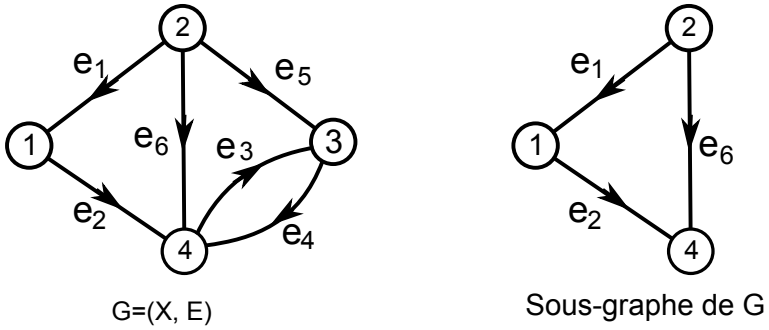


FIGURE 1.8 –

Un graphe partiel d'un sous-graphe est dit sous-graphe partiel.

1.2.10 Une clique :

Soit $G = (X, E)$ un graphe simple tel que X un ensemble de sommets de G deux â deux adjacents. Cet ensemble est dite une clique de n sommets est noté K_n. Donc un graphe G est une clique si $d_G(x) = n - 1$ pour tout ses sommets telle que $|X| = n$.[1]
La Figure (1.9) représente deux cliques de taille 3 et 5 respectivement.

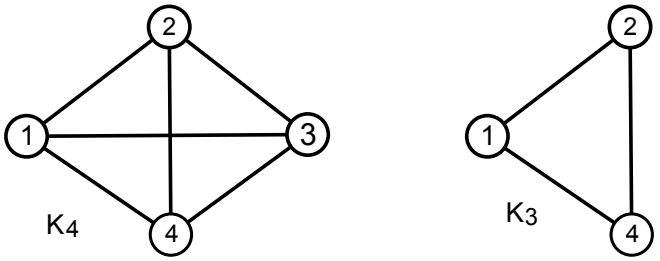


FIGURE 1.9 – Cliques.

1.2.11 Un stable :

Un stable est un ensemble de sommets de G deux â deux non-adjacents.[1]

Le graphe de la Figure 1.10 admit deux stables tel que : $S_1 = \{1, 3, 6, 8\}$ et $S_2 = \{2, 4, 5, 7\}$.

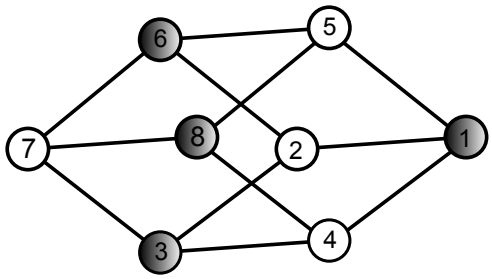


FIGURE 1.10 – Stable.

1.2.12 Chaines, chemins, cycles et circuits :

a) Chaine : Une chaine joignant deux sommets x_0 et x_k dans un graphe G est une suite de sommets relie par des arêtes tel que deux sommets successifs ont une arête commune. On la note $(x_0, x_1, \ldots, x_k)$.

- Le premier et le dernier sommet sont appelés extrémités de la chaîne.
- La longueur de la chaîne est égale au nombre d'arêtes qui la composent.

- Une chaine qui passe une seule fois par ses arêtes est dite *chaîne simple* et celle qui passe une fois par chaque sommet est dite *chaine élémentaire.*[4]

b) Chemin : Un chemin de x_0 â x_k dans un graphe est une suite de sommets relie successivement par des arcs orientée dans le même sens, on le note $(x_0, x_1, \ldots, x_k)$.

- Un chemin est simple s'il passe une et une seul fois par ses arcs et il dit chemin élémentaire s'il passe une et une seul fois par ses sommets.
- On appelle *distance* de sommet x_i â un autre sommet x_j , la longueur du plus court chemin de x_i â x_j ;
- Le diamètre de graphe G est alors la plus grand distance entre deux sommet de G.[4]

c) Cycle : C'est une chaine simple qui se ferme sur elle même.[4]

d) Circuit : Est un chemin simple qui se ferme sur lui même.[4]

1.2.13 Connexité et forte connexité :

Un graphe G est connexe ssi : $\forall x, y \in X$, il existe une chaine reliant x et y.

Si G n'est pas connexe, alors il va avoir des sous-graphes connexes, ces derniers sont appelés composantes connexes de G.

Le graphe G_1 de la Figure 1.11 est composé d'une seule composante connexe, tandis que le graphe G_2 est composé de 2 composantes connexes.

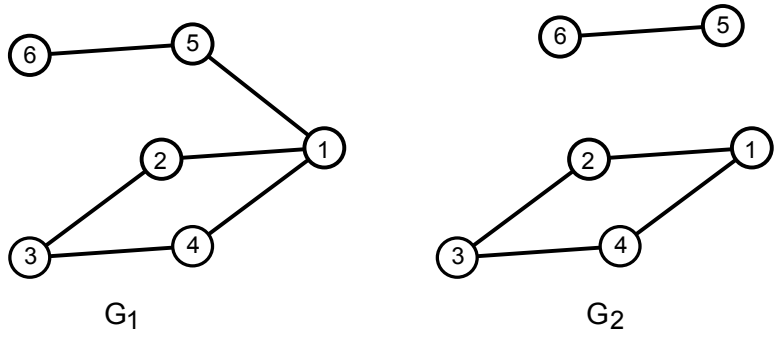


FIGURE 1.11 – Composantes connexes.

Définition 1.2.2. Un graphe G est dit **fortement connexe** si :$\forall x, y \in X^2$, il existe un chemin de x â y et un autre chemin de y à x.[13]

Définition 1.2.3. Un graphe G est dit fortement connexe s'il possède une seule composante fortement connexe.[13]

Le graphe de la Figure 1.12 contient 2 composantes fortement connexes : la première est le sous-graphe défini par les sommets $\{a, b, c, d\}$ et la seconde est le sous-graphe défini par les sommets $\{e, f, g\}$.

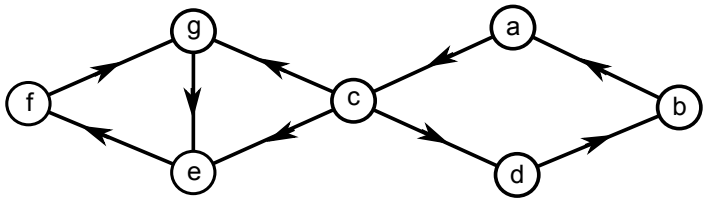


FIGURE 1.12 – Composantes fortement connexes.

On appelle le graphe réduit du graphe $G = (X, E)$ le graphe $G_r = (X_r, E_r)$ tel que : les sommets de G_r sont les composantes fortement connexes de G (on la note : c.f.c). S'il existe un arc (x, y) dans le graphe G avec : $x \in c.f.c_i$ et $y \in c.f.c_j$ alors il existera un arc (c_i, c_j) dans le graphe G_r.[13]

Le graphe de la Figure 1.13 représente le graphe réduit de la Figure 1.12.

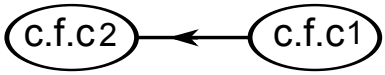


FIGURE 1.13 – Graphe réduit.

1.2.14 Algorithme de recherche des composantes connexes :

Soit $G = (X, E)$ un graphe donné.

(1) Initiation, $K = 0, W = X$ avec $X = \{$ ensemble de sommets de graphe $G \}$;

(2) Choisir un sommet et marquer le d'un signe (+), puis marquer tous ses voisins d'un signe (+) (direct et indirect), continuer cette procédure jusqu'on ne puisse marquer tout les sommets ;

(3) Poser $k = k + 1$, et C_k l'ensemble des sommets marqués ;

(3) Retirer de W les sommets C_k et poser $W = W - C_k$;

(4) Tester si $W = \emptyset$;

i) si oui terminer ;

ii) sinon aller en (1).

Le nombre de composantes connexes est k et C_k sont les composantes connexes de G.[8]

On applique l'algorithme précédent sur le graphe de la Figure 1.14 :

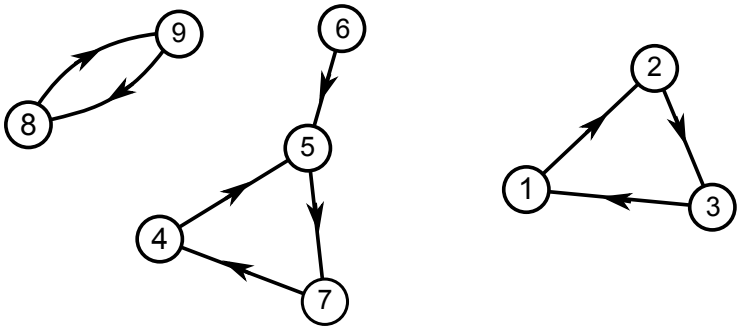


FIGURE 1.14 – Recherche des composantes connexes.

- $k = 0$, $W = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- $k = k + 1 = 1$, $C_1 = \{1, 2, 3\}$, $W = W - C_1 = \{4, 5, 6, 7, 8, 9\}$
- $k = k + 1 = 2$, $C_2 = \{4, 5, 6, 7\}$, $W = W - C_2 = \{8, 9\}$
- $k = k + 1 = 3$, $C_3 = \{8, 9\}$, $W = W - C_3 = \emptyset$; terminer.

Alors le nombre de composantes connexes de G est 3 et les composantes connexes sont : $C_1 = \{1, 2, 3\}$, $C_2 = \{4, 5, 6, 7\}, C_3 = \{8, 9\}$.

1.2.15 Algorithme de recherche des composantes fortement connexes :

Soit $G = (X, E)$ un graphe orienté.

(1) Initiation, $k = 0, W = X$ avec $X = \{$ ensemble de sommets de graphe G $\}$; Choisir un sommet et le marquer d'un signe (+) ou (-).

(2) Marquer tous ses successeurs direct et indirect d'un signe (+) ;

(3) Marquer tous ses prédécesseurs direct et indirect d'un signe (-) ;

(4) Poser $k = k + 1, C_k$ l'ensemble des sommets marqués d'un signe (+) et (-) ;

(5) Poser $W = W - C_k$, et effacer toutes les marques ;

(6) Tester si $W = \emptyset$

i) si oui terminer ;

ii) si non aller en (1).

Le nombre de composantes fortement connexes est k et C_k sont les composantes fortement connexes de G.[8]

On applique l'algorithme ci-dessus sur le graphe de la Figure 1.15, On obtient :

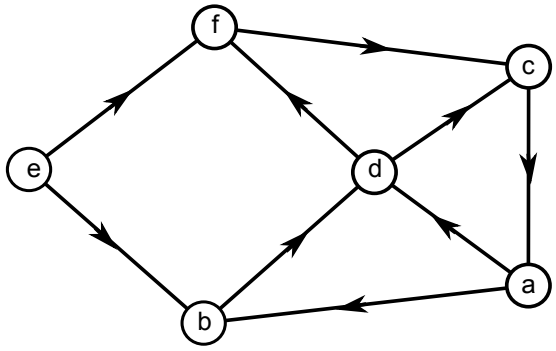


FIGURE 1.15 – Recherche des composantes fortement connexes.

- $k = 0$, $W = \{e, b, c, d, e, f\}$
- $k = k + 1 = 1$, $C_1 = \{1, 2, 3\}$, $W = W - C_1 = \{4, 5, 6, 7, 8, 9\}$
- $k = k + 1 = 2$, $C_2 = \{4, 5, 6, 7\}$, $W = W - C_2 = \{8, 9\}$
- $k = k + 1 = 3$, $C_3 = \{8, 9\}$, $W = W - C_3 = \emptyset$;, terminer.

Alors le nombre de composantes connexes de G est 3 et les composantes connexes sont : $C_1 = \{1, 2, 3\}$, $C_2 = \{4, 5, 6, 7\}$, $C_3 = \{8, 9\}$.

1.2.16 La mise en ordre d'un graphe ou la recherche d'un circuit :

Soit $G = (X, E)$ un graphe orienté.

(1) Déterminer le dictionnaire des prédécesseurs (direct) de G formé par le couple $(W, \Gamma_G^-(x))$;
(2) Repérer dans le dictionnaire des prédécesseurs les sommets n'ayant pas des prédécesseurs $\Gamma_G^-(x) = \emptyset$;
(3) Poser N_0 (niveau nul) l'ensemble des sommets n'ayant pas des prédécesseurs ;
(4) Barrer dans la colonne de $\Gamma_G^-(x)$ tous les sommets de niveau nul N_0 ;
(5) vous obtiendrez donc un nouveau tableau avec une nouvelle colonne de $\Gamma_G^-(x)$, terminer lorsque vous couvrez tous les sommets du G ;
(6) Présenter par la suite le graphe ordonné.[8]

x	$\Gamma^-(x)$
a	$\emptyset$
b	~~a~~
c	~~a~~-b
d	b-c
e	d

$N_0 = \{a\}$

x	$\Gamma^-(x)$
a	-
b	$\emptyset$
c	~~b~~
d	~~b~~-c
e	d

$N_1 = \{b\}$

x	$\Gamma^-(x)$
a	-
b	-
c	$\emptyset$
d	~~c~~
e	d

$N_2 = \{c\}$

x	$\Gamma^-(x)$
a	-
b	-
c	-
d	$\emptyset$
e	~~d~~

$N_3 = \{d\}$

x	$\Gamma^-(x)$
a	-
b	-
c	-
d	-
e	$\emptyset$

$N_4 = \{e\}$

FIGURE 1.16 – Tables des prédécesseurs du graphe G.

Pour le graphe ordonné du graphe G associé aux tables de la Figure 1.16 est donné par la Figure 1.17 :

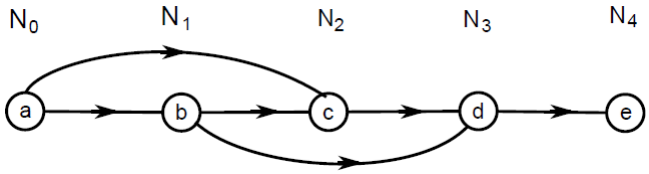
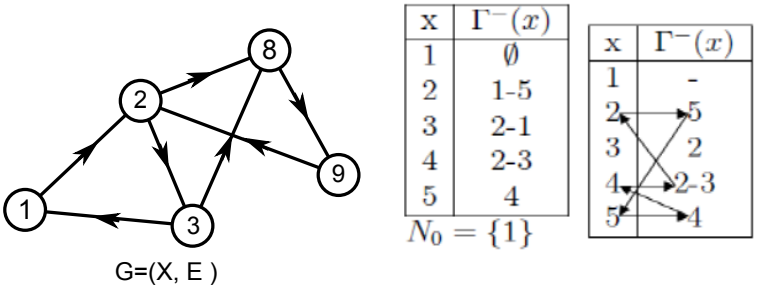


FIGURE 1.17 – Graphe ordonné.

A une étape donnée de l'ordonnancement d'un graphe la définition des niveaux se bloque (il n'existe pas de sommet), donc la mise en ordre du graphe est impossible, on dit que G possède un circuit.

On applique la procédure précédente sur la Figure 1.18. On obtient alors le dictionnaire des prédécesseurs donné.



x	$\Gamma^-(x)$
1	$\emptyset$
2	1-5
3	2-1
4	2-3
5	4

$N_0 = \{1\}$

x	$\Gamma^-(x)$
1	-
2	5
3	2
4	2-3
5	4

FIGURE 1.18 – Mise en ordre d'un graphe.

Il n'y a pas un sommet n'ayant pas de prédécesseur comme le montre le deuxième tableau, alors le graphe n'est pas ordonné, donc il contient un circuit.

Comment déterminer le circuit :

Dans la figure 1.18, le sommet 5 nous renvoi à sa ligne dans la colonne x, on choisit un sommet dans la ligne 5 soit 4, le sommet 4 nous renvoi à sa ligne dans la colonne x, on choisit un sommet dans la ligne 4, soit 2 , le sommet 2 nous renvoi à sa ligne dans la colonne x, on trouve par la suite que le sommet 5 se répète, donc on arrête la procédure.

La suite trouvé est 5,4,2,5 (la lecture est de gauche à droite) $\{5, 4, 2, 5\}$ est le circuit de graphe qui donné par la Figure 1.19.

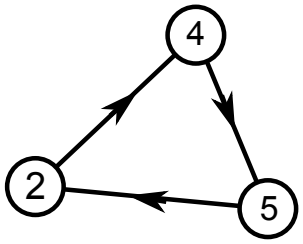


FIGURE 1.19 – Cercuit extraite à partir du graphe G.

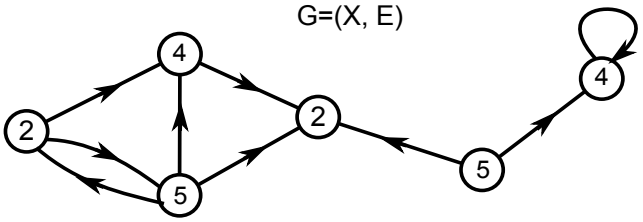
1.3 Représentation matricielle d'un graphe :

1.3.1 Matrice d'adjacence

La matrice d'adjacence est une matrice $(n * n)$ tel que chaque ligne et chaque colonne correspond à un sommet de tel sorte :

$$a_{ij} = \begin{cases} 1 & \text{s'il éxiste un arc } (x_i, x_j) \text{ ou } x_i = I(e) \text{ et } x_j = T(e), \\ 0 & \text{sinon} \end{cases}$$

La Figure 1.20 représente Un graphe G et sa matrice d'adjacence.[10]



	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	1	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

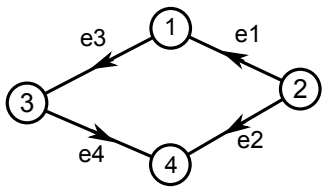
FIGURE 1.20 – Graphe G et Sa matrice d'adjacence.

1.3.2 Matrice d'incidence aux arcs :

La matrice d'incidence aux arcs d'un graphe sans boucles est une matrice $(n*m)$ tel que chaque ligne correspond à un sommet et chaque colonne à un arc et qui prennent comme valeurs.[10]

$$a_{ij} = \begin{cases} 1 & \text{si } x_i \text{ est l'extrémité initial de } e_j, \\ -1 & x_i \text{ est l'extrémité terminal de } e_j \\ 0 & \text{sinon} \end{cases}$$

La Figure 1.21 représente la matrice d'incidence du graphe G.



	e_1	e_2	e_3	e_4
1	-1	0	1	0
2	1	1	0	0
3	0	0	-1	1
4	0	-1	0	-1

FIGURE 1.21 – Matrice d'incidence.

Remarque 1.3.1. $d_G^+(x)$= la somme des valeurs (1) d'une ligne du sommet correspondant à x. $d_G^-(x)$= la somme des valeurs (-1) d'une ligne du sommet correspondant à x.

1.3.3 Couplage :

Soit $G = (X, E)$ un graphe. Un couplage est un sous ensemble d'arêtes $M \subset E$ tel que deux arêtes de M ne sont pas adjacentes.

- Un sommet $x \in X$ est dit saturé par le couplage $M \subset E$ s'il existe une arête de M incidente à e.
- Un couplage M qui sature tout les sommets de graphe est appelé "couplage parfait".
- Un couplage maximum est le couplage de cardinalité maximale.[1]

Le graphe de la Figure 1.22 représente un couplage maximum et parfait.

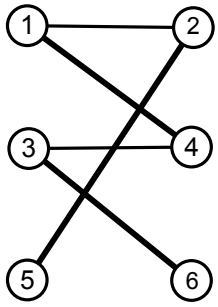


FIGURE 1.22 – Couplage.

1.4 Quelques graphes particuliers :

1.4.1 Graphe réflexif :

On appelle graphe réflexif, un graphe possédant une boucle sur chaque sommet.

Le graphe de la Figure 1.23 est réflexif.[16]

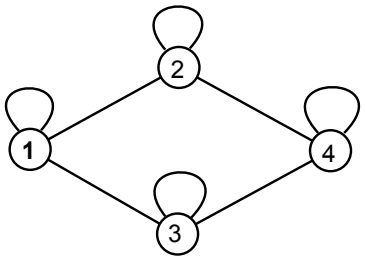


FIGURE 1.23 – Graphe reflexif.

1.4.2 Graphe symétrique :

Un graphe est dit symétrique si pour tout arc $e_i = (x, y)$ il existe un arc $e_j = (y, x)$.[16]

La Figure 1.24 montre un graphe symétrique.

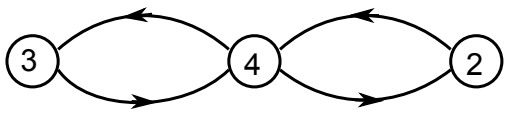


FIGURE 1.24 – Graphe symétrique.

1.4.3 Graphe antisymétrique :

Un graphe est dit antisymétrique si pour tout arc $e_i = (x, y)$ il n'existe pas d'arc $e_j = (y, x)$. Le graphe donné par la Figure 1.25 est un graphe antisymétrique.

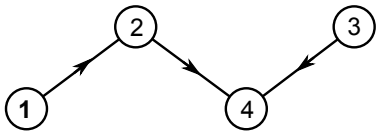


FIGURE 1.25 – Graphe antisymétrique.

1.4.4 Graphe transitif :

Un graphe est dit transitif si pour chaque deux arcs $e_i = (x, y)$ et $e_j = (y, z)$, il existe un arc $e_k = (x, z)$.[16]

Le graphe de la Figure 1.26 est un graphe transitif.

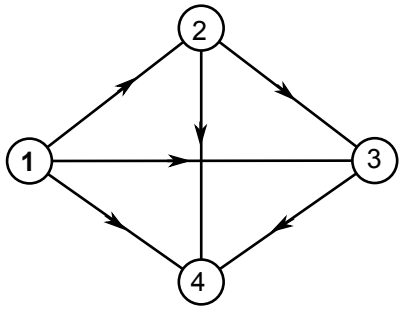


FIGURE 1.26 – Graphe transitif.

1.4.5 Graphe complet :

Un graphe est dit complet si chaque sommet est relie avec tous les autres sommets. à titre d'exemple, le graphe de la Figure 1.27 est un graphe complet.[16]

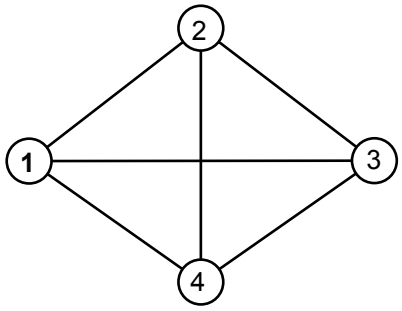


FIGURE 1.27 – Graphe complet.

1.4.6 Graphe biparti :

Un graphe est dit biparti si ses sommets peuvent être divisés en deux classes X et Y de telle sorte que deux sommets de la même classe ne sont jamais reliés. Les arêtes d'un graphe biparti sont les arêtes reliant un sommet de X à un sommet de Y.[14]

La Figure 1.28 représente un graphe biparti.

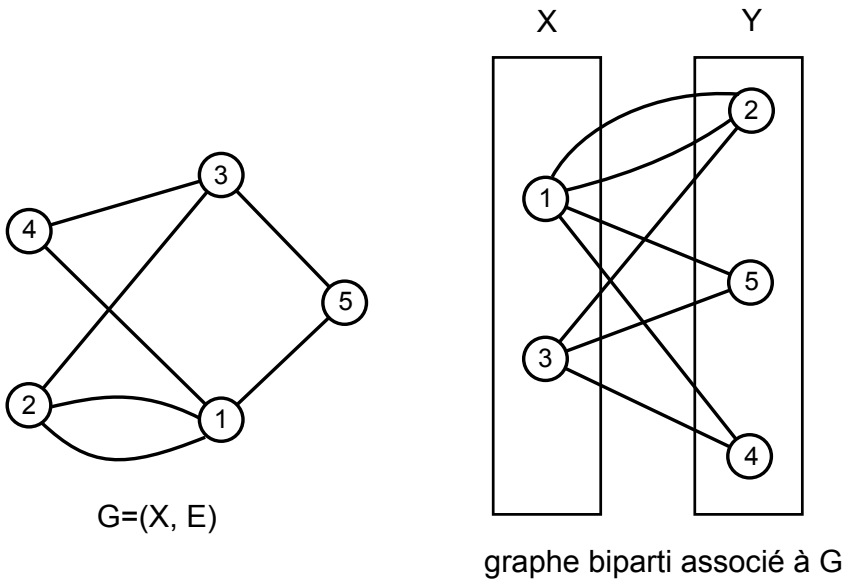


FIGURE 1.28 – Graphe biparti.

1.4.7 Graphe biparti complet :

Un graphe biparti complet est un graphe biparti tel que chaque sommet de X est relie avec tous les sommets de Y qu'on note $K_{p,q}$ ou $|X| = p, |Y| = q$. La Figure 1.29 représente un graphe

biparti-complet.[14]

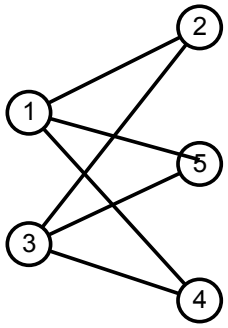


FIGURE 1.29 – Graphe biparti complet.

1.4.8 Graphe biparti équilibré :

Soit G un graphe biparti. Si $|X| = p$ et $|Y| = p$ alors, on dit que G biparti équilibré.[14]

1.4.9 Graphe planaire :

Un graphe planaire est un graphe qu'on peut dessiner sur un plan de telle sorte que deux arêtes ne se coupent pas en dehors de leurs extrémités.

Une face d'un graphe planaire est une région de plan limité par arêtes (au minimum 3 arêtes). Deux faces sont adjacentes si elles ont au moins une arête en commune.[13]

La Figure 1.30 est une représentation d'un graphe planaire.

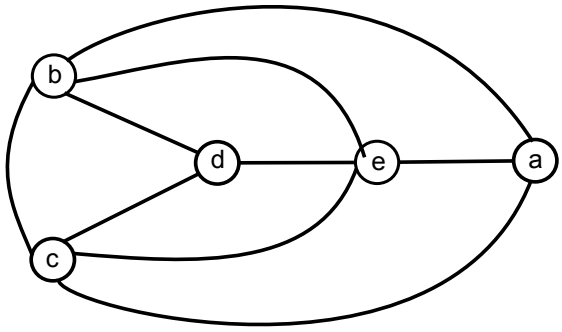


FIGURE 1.30 – Graphe planaire.

1.4.10 Arbre :

Un arbre est un graphe connexe sans cycle ayant les propriétés suivantes :

- G est connexe et sans cycle,
- G est sans cycle et possède $n-1$ arêtes,
- G est connexe et admet $n-1$ arêtes,
- G est sans cycle, et en ajoutant une arête, on crée un et un seul cycle élémentaire,
- G est connexe, et en supprimant une arête quelconque, il n'est plus connexe,
- Il existe une chaine et une seule entre 2 sommets quelconques de G.[8]

La Figure 1.31 constitue un arbre.

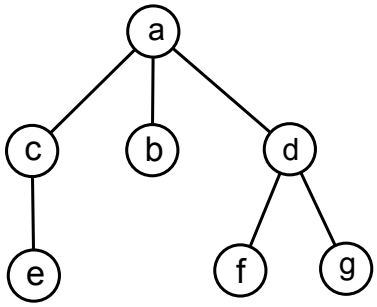


FIGURE 1.31 – Arbre.

1.4.11 Arborescence :

Est un graphe orienté sans circuit admettant une racine $x_0 \in X$ telle que, pour tout autre sommet $x_i \in X$, il existe un chemin unique allant de x_0 vers x_i. Si l'arborescence comporte n sommets, alors elle comporte exactement $n-1$ arcs.[8]

Le graphe de la Figure 1.32 est une arborescence de racine a.

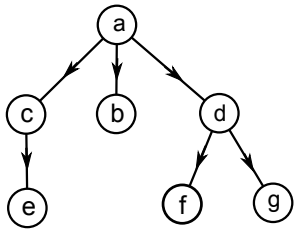


FIGURE 1.32 – Arbrescence.

Conclusion

Ce chapitre nous a permis d'introduire quelques notions de bases de la théorie des graphes et de souligner la simplicité de ce formalisme. Puis nous avons porté une attention particulière sur la connexité et forte connexité à d'énoncé les différentes opérations dans les graphes.

Chapitre 2

Quelques problèmes d'optimisation

L'organisation des horaires dans une école (emploi de temps), l'allocation de fréquences dans un réseau de télécommunication ou le transport d'une marchandise à des clients aux délais et en minimum de temps et d'argent sont des problèmes difficiles. Il existe un modèle mathématique permettant d'aborder ces situations de manière efficace : le graphe. Dans ce chapitre, nous présentons brièvement les principales définitions et la modélisation de chaque modèle, avec illustrations par des exemples.

2.1 Coloration de graphes :

Une coloration d'un graphe $G = (V, E)$ est une application $c : V \to S$, ou les éléments de S sont appelés les couleurs. Une coloration c est dite propre si pour toute arête $uv \in E(G)$, $c(u) \neq c(v)$. comme la taille de S est le paramètre important, on considère souvent que $S = \{1, 2, \ldots, k\}$. une k-coloration d'un graphe G (sans boucle) est une coloration propre de G à valeur dans $\{1, 2, \ldots, k\}$. Un graphe est k-colorable s'il admet une k-coloration. Le plus petit entier k tel que G soit k-colorable est le *nombre chromatique* de G, noté $\chi(G)$.

2.1.1 Coloration des sommets d'un graphe :

Proposition 2.1.1. *Si H est un sous-graphe de G alors $\chi(H) \leq \chi(G)$.*

Preuve : Si c est une coloration propre de G alors c est clairement une coloration propre de H.

Proposition 2.1.2. *$\chi(G) = max\{\chi(C)$, C composante connexe de $G\}$.*

Preuve : D'aprés la proposition 2.1.1 , $\chi(G) \geq max\{\chi(C)$, C composante connexe de G $\}$. Soit $C_1, C_2, \ldots, C_k$ les composantes connexes de G. Pour $1 \leq i \leq K$, soit c_i une coloration propre de C_i avec les couleurs $1, 2, \ldots, \chi(C_i)$. Considérons c défini par $c(v) = c_i(v)$ si $v \in C_i$. Comme il n'y a pas d'arêtes entre deux sommets composantes connexes différentes, c est une coloration propre

de G. Ainsi $\chi(G) \leq max\{\chi(C), C \text{ composante connexe de } G\}$.

Dans la suite, nous considérons la plupart du temps que les graphes sont connexes.

a) Bornes inférieurs pour $\chi(G)$:

Proposition 2.1.3. $\chi(G) \geq \omega(G)$.

La taille maximale d'une clique est une borne inférieur de $\chi(G)$. Malheureusement, ce n'est pas un bon indicateur du nombre chromatique.[16]

Proposition 2.1.4. $\chi(G) \geq \dfrac{|V(G)|}{\alpha(G)}$ *[16]*

Preuve : Soit c une coloration propre de G avec les couleurs $1, 2, \ldots, \chi(G)$ et $S_i = \{v \in V(G) : c(v) = i\}$ pour $1 \leq i \leq \chi(G)$. chaque S_i est un stable donc $|S_i| \leq \alpha(G)$. D'où : $|V(G)| = \sum_{i=1}^{\chi(G)} |S_i| \leq \sum_{i=1}^{\chi(G)} \alpha(G) \leq \chi(G) \cdot \alpha(G)$.

b) Bornes supérieures pour $\chi(G)$:

La plupart des bornes que nous allons établir sont obtenues en utilisant un **algorithme glouton**. On prend un ordre $\sigma = (v_1 < v_2 < \ldots < v_n)$ sur les sommets de G. On colore ensuite les sommets de manière séquentielle avec les entiers naturels de la façon suivante : le sommet v_i est coloré avec le plus petit entier non utilisé par ses voisins déjà colorés (i.e. d'indice plus petit).

Combien de couleurs cet algorithme glouton utilise-t-il ? si pour tout i le nombre de couleurs déjà attribuées aux voisins de v_i, parmi $v_1, v_2, \ldots, v_{i-1}$ est au plus D alors l'algorithme glouton utilise au plus $D + 1$ couleurs. Comme un sommet à au plus $\Delta(G)$ voisins, on a alors :

Proposition 2.1.5. $\chi(G) \leq \Delta(G) + 1$.

L'algorithme glouton n'utilise pas nécessairement $\chi(G) \leq \delta(G) + 1$ couleurs. En particulier, il existe des ordres pour lesquels l'algorithme de coloration séquentielle est optimal, i.e. utilise $\chi(G)$ couleurs : soit c une coloration propre de G avec les couleurs $1, 2, \ldots, \chi(G)$, tout ordre $\sigma_{opt} = (v_1 < v_2 < \ldots < v_n)$ tel que $c(v_i) \leq c(v_j)$ si $i \leq j$ convient. Cependant trouver un telle ordre parmi les $n!$ possibles est NP-difficile.

On peut cependant trouver des ordres qui améliorent la borne $\Delta(G) + 1$ ci-dessus.[2]

Définition 2.1.1. Soit $\sigma = (v_1 < v_2 < \ldots < v_n)$ un ordre sur les sommets d'un graphe G. on défini $g(\sigma)$ comme le maximum pour $1 \leq i \leq n$ du degré du sommet v_i dans le sous-graphe induit par $v_1, v_2, \ldots, v_i$.

La *dégénérescence* de G, noté $\delta^*(G)$ est la valeur minimum de $g(\sigma)$ sur tout les ordres des sommets σ possibles. Notons que $\delta^*(G)$ et un ordre δ^* associé peuvent être facilement trouvés. Il suffit de prendre un sommet de plus petit degré dans G, de le mettre à la fin de l'ordre, de l'enlever de G, et de répéter l'opération. Clairement on a $\delta^* \leq \Delta(G)$.[2]

L'algorithme glouton suivant σ^*, nous donne alors :

Proposition 2.1.6. $\chi(G) \leq \delta^*(G) + 1$.

Théorème 2.1.1. ***(Brooks 1941) :*** *Soit G un graphe connexe. si G n'est ni un graphe complet ni un cycle impair alors $\chi(G) \leq \Delta(G)$.[3]*

Rappel : Un graphe G est 2-connexe, si pour tout couple de sommets u et v, il existe deux chemins disjoint allant de u en v.

preuve :

- Si G n'est pas $\Delta(G)$-régulier, on a le résultat par les propositions 2.1.6 , on peut donc désormais supposer que G est $\Delta(G)$-régulier. Si $\Delta(G) = 2$ alors G est un cycle pair. Il est donc 2-colorable. On peut donc supposer que $\Delta(G) \geq 3$.
- Si G n'est pas 2-connexe, alors il existe un sommet v tel que $G - v$ ait $k \geq 2$ composantes connexes $C_1, C_2, \ldots, C_k$. Chaque graphe G_i induit par $V(C_i) \bigcup \{v\}$ vérifie $\Delta(G_i) \leq \Delta(G)$ et $d_{G_i}(v) < \Delta(G)$. Les graphes G_i ne sont pas réguliers. Ainsi par une proposition, chaque G_i est $\Delta(G)$-colorable. Prenant pour chaque G_i une $\Delta(G)$-coloration c_i tel que $c_i(v) = 1$, l'union des c_i nous donne une $\Delta - coloration$ de G.
- Supposons enfin que G soit 2-connexe. Comme G n'est pas le graphe complet, il existe trois sommets u, v et w tel que $uv \in E(G)$, $uw \notin E(G)$ et $G - \{u, w\}$ soit connexe. il existe alors un ordre $\sigma = (v_1 = u < v_2 = w < v_3 < \ldots < v_n = v)$ de $V(G) \backslash \{u, w\}$ tel que pour $i < n$, le sommet v_i soit adjacent à un v_j avec $j > i$ (il suffit d'ordonner les sommets de $V(G) \backslash \{u, w\}$) par rapport à leur distance par rapport à v), Colorons alors u et w avec 1. On peut ensuite colorer les sommets de $V(G) \backslash \{u, w\}$ suivant σ avec la première couleur disponible de $\{1, 2, \ldots, \Delta(G)\}$. En effet pour $i < n$, le sommet v_i à au plus $\Delta(G) - 1$ voisins déjà colorés, ainsi il peut être coloré par une des couleurs de $\{1, 2, \ldots, \Delta(G)\}$. pour le sommet v, il a deux voisins colorés par 1, il est donc adjacent à au plus $\Delta(G) - 1$ autres couleurs et il peut donc être coloré par une couleur de $\{1, 2, \ldots, \Delta(G)\}$.

La borne supérieure Δ pour χ est loin d'être optimale et elle peut être améliorée par de nombreuses classes de graphes. Par exemple pour les graphes sans triangle (pour lesquels $w = 2$).

Proposition 2.1.7. *Soit G un graphe sans triangle. Alors $\chi(G) \leq 3 \left[\frac{\Delta(G)+1}{4} \right]$.[3]*

Preuve : Posons $k = \left[\frac{\Delta(G)+1}{4} \right]$. soit $(V_1, V_2, \ldots, V_k)$ la partition de $V(G)$ en k ensembles tels que le nombre d'arêtes internes (i.e. avec les deux extrémités dans une même partie) est minimum.

pour tout i le graphe G_i induit par V_i est de degré maximum au plus 3. En effet supposons qu'un sommet x d'une des parties, disons V_1, ait 4 voisins dans V_1. Alors une autre partie, disons V_2 contient au plus 3 voisins de x, sinon x aurait au moins $4k \geq \Delta(G) + 1$ voisins ce qui est impossible. Ainsi la partition $(V_1 - x, V_2 + x, \ldots, V_k)$ a moins d'arêtes internes que $(V_1, V_2, \ldots, V_k)$ ce qui contredit la minimalité de celle-ci.

Ainsi $\Delta(G_i) \leq 3$ et G_i ne contient pas de 3-clique car c'est un sous-graphe de G. Donc par le théorème de Brooks, $\chi(G_i) \leq 3$.

Ainsi on colorant les G_i avec des ensembles de couleurs distinctes, on obtient une $3k$-coloration de G.

En fait on pense que la borne supérieur adéquate est $\chi(G) \leq \left\lceil \frac{\Delta(G) + \omega(G) + 1}{2} \right\rceil$ (Reed 1998).

2.1.2 Coloration des arêtes d'un graphe :

Une arête-coloration d'un graphe $G = (X, E)$ est une application $c : E \to S$ telle que pour deux arêtes e et f adjacents on a $c(e) \neq c(f)$. Une k-arête-coloration d'un graphe G est une coloration dans $S = \{1, 2, \ldots, k\}$. Un graphe est k-arête-colorable s'il admet une k-arête-coloration. Le plus petit entier k telle que G soit k-arête-colorable est *l'indice chromatique de* G, note $\chi'(G)$.

Il est évident que tout graphe G vérifie $\chi'(G) \geq \Delta(G)$. Nous allons voir que pour les graphes bipartis $\Delta(G)$ couleurs différents.

Théorème 2.1.2. ***(Konig 1916) :*** *Si G est un graphe biparti alors $\chi'(G) = \Delta(G)$.[3]*

Preuve : Par récurrence sur le nombre d'arêtes. Si $|E(G)| = 0$, le théorème est trivialement vrai. Supposons maintenant que $|E(G)| \geq 1$ et que le résultat est vrai pour les graphes ayant moins d'arêtes. Posons $\Delta(G) = \Delta$. Soit xy une arête de G. Par hypothèse de récurrence, $G - xy$ admet une Δ-arête-coloration.

Dans $G - xy$, les sommets x et y sont chacun incident à au plus $\Delta - 1$ arêtes. Ainsi il existe $c_x, c_y \in \{1, 2, \ldots, \Delta\}$ tels que x (resp y) ne soit pas adjacent à une arête colorée c_x (resp c_y). Si $c_x = c_y$, alors on peut colorer xy avec cette couleur et on obtient une Δ-arête-coloration de G. Nous pouvons donc supposer que $c_x \neq c_y$ et que le sommet x est incident à une arête e colorée c_y.

étendons cette arête en une marche maximale P dont les arêtes sont colorées c_x et c_y alternativement. Comme un sommet est adjacent à au plus une arête d'une couleur, P est un chemin. De plus, P ne convient pas y. Sinon P terminerait en y avec une arête colorée c_x et P serait un chemin pair. Mais alors $P + xy$ formerait un cycle impair, ce qui est impossible d'après une proposition. On peut donc recolorer P en intervertissant les couleurs c_x et c_y. Par maximalité de P, deux arêtes adjacents sont toujours de couleurs différentes. En colorant xy par c_y, on obtient

alors une $\Delta-$arête-coloration de G.

Ce théorème n'est cependant pas vrai lorsque le graphe n'est pas biparti.

Théorème 2.1.3. *Pour tout graphe G, $\Delta(G) \leq \chi'(G) \leq \Delta(G)+1$.[3]*

Preuve : Nous allons prouver la deuxième inégalité par récurrence sur $|E(G)|$. Pour $|E(G)| = 0$, le résultat est trivialement vrai. Supposons maintenant que $|E(G)| \geq 1$ et que le résultat est vrai pour tout graphe avec moins d'arêtes que G. Posons $\Delta(G) = \Delta$. Soit xy_0 une arête de G. Par hypothèse de récurrence, $G - xy_0$ admet une $(\Delta+1)-$arête coloration. Comme y_0 est incident à au plus $\Delta - 1$ arêtes dans $G - xy_0$, il existe $c_1 \in \{1, 2, \ldots, \Delta+1\}$ tel que aucune arête incidente à y_0 ne soit colorée c_1. Si aucune arête incidente à x n'est colorée c_1 alors, en colorant xy_0 par c_1, on obtient une $(\Delta+1)-$arête-coloration de G. On peut donc supposer qu'il existe une arête xy_1 colorée c_1. Comme y_1 est incident à au plus Δ arêtes, il existe $c_2 \in \{1, 2, \ldots, \Delta+1\}$ tel qu'aucune arête incidente à y_1 ne soit colorée c_2. Si aucune arête incidente à x n'est colorée c_2, en recolorant xy_1 par c_2 et en colorant xy_0 par c_1 on obtient une $(\Delta+1)-$arête-coloration de G. on peut donc supposer qu'il existe une arête xy_2 colorée c_2. Et ainsi de suite on construit une suite $y_1, y_2, \ldots$ de sommets et une suite de couleurs $c_1, c_2, \ldots$ telles que :

(i) xy_i est colorée c_i, et

(ii) c_{i+1} ne colore aucune arête incidente à y_i.

Comme le degré de x est fini, il existe un plus petit entier l tel que pour un entier $k < l$,

(iii) $c_{l+1} = c_k$.

Maintenant, pour $0 \leq i \leq k-1$, recolorons l'arête xy_i avec c_{i+1}.

il existe une couleur $c_0 \in \{1, 2, \ldots, \Delta+1\}$ qui n'est attribuée à aucune des incidentes à x. En particulier, $c_0 \neq c_k$. Soit P le chemin maximal issu de y_{k-1} coloré alternativement c_0 et c_k. Intervertissons les couleurs c_0 et c_k sur $P + xy_{k-1}$. Si P ne contient pas y_k, nous avons une $(\Delta+1)-$arête-coloration de G. Si P contient (et termine en) y_k, en recolorant l'arête xy_i avec c_{i+1} pour $k \leq i \leq l$, on obtient une $(\Delta+1)-$arête-coloration de G.

2.2 Problème de cheminement :

Les problèmes de cheminement dans les graphes comptent parmi les problèmes les plus anciens de la théorie des graphes et les plus importants par leurs applications.

Le plus court chemin est le plus typique. Il se rencontre soit directement, soit comme sous problème dans de nombreuses applications.

2.2.1 Définitions :

a) Réseau :

Un réseau est un graphe $G = (X, U)$ muni d'une application $d : U \to \Re$ qui à chaque arc fait correspondre sa longueur $d(u)$, on note un tel réseau par : $R(X, U, d)$. En pratique, $d(u)$ peut matérialiser un coût, une distance, une durée etc. La Figure 2.1 montre un réseau routier : [1]

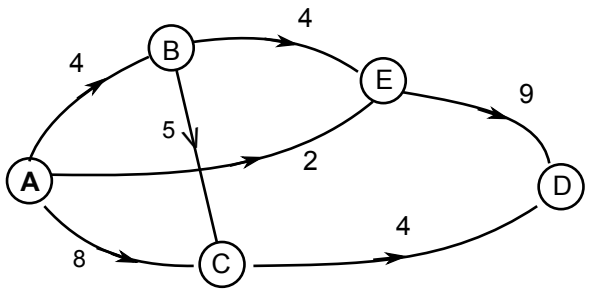


FIGURE 2.1 – Exemple d'un réseau routier.

b) Langueur d'un chemin dans un réseau :

La langueur d'un chemin C dans un réseau R est égale à la somme des longueurs des arcs comptés dans leur ordre de multiplicité (poids) dans le chemin (le nombre de fois qu'on parcourt ces arcs), on le note par $l(C) = \sum_{u \in C} d(u)\eta(u)$, ($\eta(u)$: multiplicité d'un arc u).

On définit de la même manière, la langueur d'un cycle, d'un circuit et d'une chaine.

Dans le réseau de la Figure 2.1, soit le chemin $C = (A, B, C, D)$. sa longueur est : $l(C) = \sum_{u \in C} d(u)\eta(u) = d(AB)\eta(AB) + d(BC)\eta(BC) + d(CD)\eta(CD) = 4 \times 1 + 5 \times 1 + 4 \times 1 = 13$.

Dans le chemin C, l'ordre de multiplicité de chaque arc ($\eta(u)$) est de 1.

Remarque 2.2.1. étant donnés, deux sommets x et y d'un réseau $R = (X, U, d)$, trois cas peuvent se présenter :

- Il n'existe pas de chemins entre x et y.
- Il existe un chemin entre x et y dans R mais pas de chemin de longueur minimale.
- Parmi tous les chemins joignant x et y dans R il en existe un qui est de longueur minimale.[1]

Comme exemple, considérons le graphe de la Figure 2.2 :

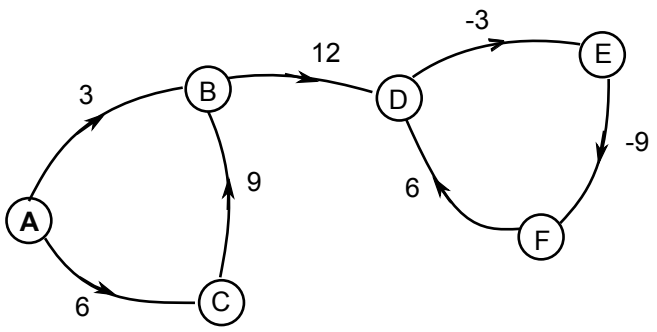


FIGURE 2.2 – Chemin dans un réseau.

Dans le réseau R,

- Il existe un chemin de langueur minimale de A à B, mais pas de chemin inverse de B à A.
- Il existe deux chemins de A à D, le chemin ABD et le chemin $ACBD$. Le chemin ABD est de longueur minimale.
- Il existe un chemin de B à F mais pas de chemin de longueur minimale, En effet si on considère l'ensemble des chemins passant par l'arc (B, D) et par le circuit $[(D, E), (E, F), (F, D)]$; puis on contourne le circuit, plus la langueur de chemin devient plus petite, donc on ne pourra pas déterminer un plus court chemin issu de B à x dans le réseau R.

c) Plus court distance :

Si un plus court chemin (chemin de longueur minimale) d'un sommet s à un sommet x existe dans un réseau. La longueur de ce plus court chemin sera appelée " plus court distance de s à x " et se notera $\pi(x)$.

Dans le réseau de la Figure 2.2, il existe deux chemins de A à B. le plus court chemin est le chemin AB de longueur 1, donc le plus court distance de A à B est $\pi(B) = 1$.[1]

d) Circuit absorbant :

Un circuit est dit absorbant si sa longueur est négative.$l(C) = \sum_{u \in C} d(u) < 0$.

dans le réseau de la Figure 2.3,

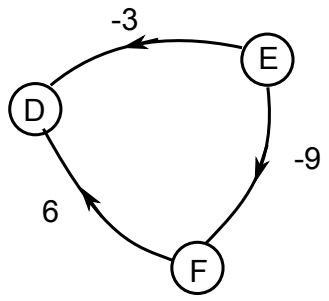


FIGURE 2.3 – Circuit absorbant

Le circuit $C = (D, E, F, D)$ ci-dessus est absorbant car $l(C) = -6 < 0$ et plus on contourne le circuit, plus sa longueur devient minimale. (tend vers $-\infty$)

Dans cette section, on s'intéresse à la recherche d'un plus court chemin, joignant un sommet s donné à chaque sommet x du réseau $R = (X, U, d)$.

ce type de problème admet une solution si et seulement si :
- s est une racine du réseau.
- Le réseau n'admet pas de circuit absorbant.[1]

Note : La solution de ce type de problème si elle existe est une arborescence des plus courts chemins admettant s comme racine.

2.3 Problème du transport :

2.3.1 Modèle mathématique du problème de transport :

étant donné m points de production (usines) $A_i, i = \overline{1, m}$ et n points de consommations (dépôts) $B_j, j = \overline{1, n}$.

Chaque usine produit une quantité $a_i, i = \overline{1, m}$, qui doit être acheminé vers le dépôt $B_j, j = \overline{1, n}$, dont la capacité est $b_j, j = \overline{1, n}$.

Le problème de transport est est identique au problème de l'offre et de la demande. Ici notre problème consiste à transporter les quantités produits vers les dépôts, avec un coût de transport minimum.

Soit $x_{ij}, i = \overline{1, m}, j = \overline{1, n}$ la quantité de produit à transporter de A_i vers B_j, et c_{ij} le cout de transport d'une unité du produit à transporter de A_i vers B_j.

Si une quantité x_{ij} est acheminée de A_i vers B_j, le cout de transport de cette quantité est égale à $c_{ij}x_{ij}$. De là le cout globale de transport est égale à : $Z = Z(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}x_{ij}$, ou

$x = (x_{ij}, i = \overline{1,m}, j = \overline{1,n})$.

De plus les quantités à transporter doivent satisfaire les contraintes suivantes :

i) Toute la marchandise produite doit être acheminée : $\sum_{j=1}^{n} x_{ij} = a_i, i = \overline{1,m}$

ii) Toutes les demandes doivent être satisfaites : $\sum_{i=1}^{m} x_{ij} = b_i, j = \overline{1,n}$

iii) Toutes les quantités $x_i j$ doivent être positives ou nulles : $x_{ij} \geq 0, i = \overline{1,m}, j = \overline{1,n}$

Ainsi, le modèle mathématique du problème de transport prend la forme du programme de programmation linéaire suivant :

$$Z = Z(x) = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \to min \tag{2.1}$$

$$\sum_{j=1}^{n} x_{ij} = a_i, i = 1, m \tag{2.2}$$

$$\sum_{i=1}^{m} x_{ij} = b_i, j = 1, n \tag{2.3}$$

$$x_{ij} \geq i = 1, m, j = 1, n \tag{2.4}$$

L'ensemble $x = x_{ij}, i = \overline{1,m}, j = \overline{1,n}$ est dit plan du transport du problème $(2.1) - (2.4)$ s'il satisfait les contraintes $(2.2), (2.3)$ et (2.4). Il est dit plan optimal de transport si de plus il réalise le minimum de la fonction objectif (2.1).

2.3.2 Condition d'existence d'un plan optimal de transport :

Dans le problème $(2.1) - (2.4)$, 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 soient supérieurs ou égales aux quantités demandées, c'est à dire :

$$\sum_{i=1}^{m} a_i \geq \sum_{j=1}^{n} b_j. \tag{2.5}$$

Dans le cas ou on aurait la condition (2.5), on crée un dépôt fictif (supplémentaire), ou on produit la quantité produite restante pour avoir l'égalité :

$$a_i = \sum b_j = K \tag{2.6}$$

Par la suite on supposera qu'on a toujours la condition (2.6) appelée **condition de balance**, c'est-à-dire l'offre est égale à la demande.

Théorème 2.3.1. *Le problème* $(2.1) - (2.4)$ *possède un plan optimal de transport si et seulement si la condition de balance* (2.6) *est vérifiée.[5]*

2.3.3 Graphe et tableau de transport associé au problème :

Le graphe associé au problème $(2.1) - (2.4)$ est un graphe simple biparti (2.4). L'ensemble U des arcs de ce graphe est de la forme : $U = \{(i,j), i \in I, j \in J\}, I = \{1, 2, \ldots, m\}, j = \{1, 2, \ldots, n\}$.

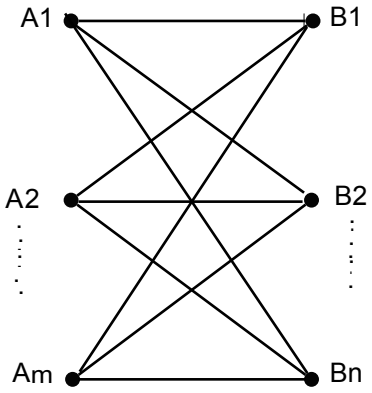


FIGURE 2.4 – Graphe représentatif du problème de transport.

On associe ainsi au problème $(2.1) - (2.4)$ un tableau de transport à m lignes et à n colonnes, défini de la manière suivante :

Chaque ligne $i(i \in I)$ correspond au point de production A_i et chaque colonne $j(j \in J)$ au point de distribution B_j.

La case (i,j) du tableau correspond au chemin qui relie A_i et B_j. Chaque case (i,j) contient les quantités associées à l'arc (i,j) :

- Le coût unitaire c_{ij} de transport de A_i à B_j, indiqué en haut et à droite de la case.
- La valeur de la variable x_{ij} écrite en bas et à gauche de la case.

En autre, le tableau est bordé à droite par une colonne indiquant les disponibilités a_i et par une ligne située en bas et indiquant les demandes b_j.

Le tableau de transport présente alors la forme suivante :

	B_1	B_2	B_3	...	B_n	a_j
A_1	c_{11} x_{11}	c_{12} x_{12}	c_{13} x_{13}	...	c_{1n} x_{1n}	a_1
A_2	c_{21} x_{21}	c_{22} x_{22}	c_{23} x_{23}	...	c_{2n} x_{2n}	a_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
A_m	c_{m1} x_{m1}	c_{m2} x_{m2}	c_{m3} x_{m3}	...	c_{mn} x_{mn}	a_m
b_j	b_1	b_2	b_3	...	b_n	$\sum b_j = \sum a_i$

La sonatrach possède trois points de production, Hassi Massaoud (A_1), Ain Salah (A_2) et Illizi (A_3), avec des capacités de production 100 milles, 150 milles et 200 milles barils/jour respectivement. Et deux points de distribution, Skikda (B_1) et Arzew (B_2), avec des demandes de 250 milles et 200 milles barils/jour respectivement.

Les couts unitaires de transport des points de production aux ponts de distribution sont donnés par le tableau suivant :

	B_1	B_2	Offres a_i (en milliers)
A_1	90 DA	75 DA	100
A_2	80 DA	70 DA	150
A_3	60 DA	95 DA	200
Demandes b_j (en milliers)	250	200	450

Le problème consiste à déterminer le nombre d'unités que la société doit transporter des différents points de production aux différents points de distribution, de telles façon que le cout global de transport soit minimale.

soit :

x_{ij} = quantité à transporter du point de production " i " vers le point de distribution " j ".

Le problème de transport correspondant à ce problème est le suivant :

	B_1	B_2	a_j
A_1	90 x_{11}	75 x_{12}	100
A_2	80 x_{21}	70 x_{22}	150
b_j	250	200	450

2.3.4 Le problème du flot maximum :

Définitions :

a) Réseau de transport :

Un réseau de transport est un graphe sans boucles, où chaque arc est valué par un nombre positif c(u) appelé capacité de l'arc u. Ce réseau comporte un sommet sans prédécesseurs appelé " l'entrée du réseau " ou " la source " et un autre sommet sans successeurs appelé " la sortie du réseau " ou " le puits " .[6]

La Figure 2.5 représente un réseau de transport :

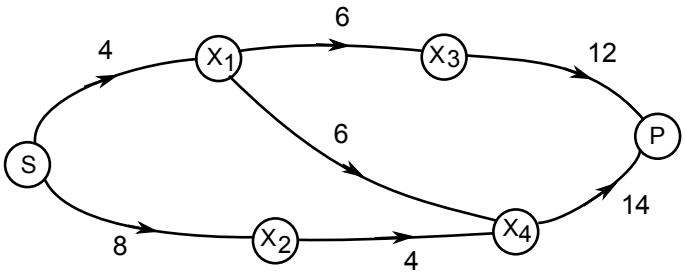


FIGURE 2.5 – Exemple d'un réseau de transport.

b) Flot :

Un flot f dans un réseau de transport, associe à chaque arc u une quantité $f(u)$ qui représente la quantité de flux qui passe par cet arc en provenance de la source vers le puits.[6]

Remarque 2.3.1. Un flot est conservatif, s'il obéit à la règle de khirchoff aux noeuds (aux sommets) suivante :

La somme des quantités de flux sur les arcs entrant dans un sommet doit être égale a la somme des quantités de flux sur les arcs sortant de ce même sommet. Comme le montre la Figure 2.6[6]

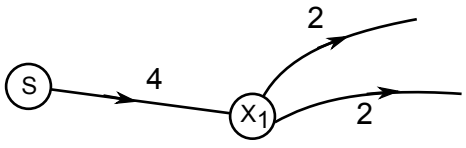


FIGURE 2.6 – Exemple d'un flot.

Dans le graphe de la Figure 2.6, la quantité de flux entrant dans x_1 est égale à la somme des

quantités de flux sortant de x_1 est égale à la somme des quantités de flux sortant de x_1, la quantité de flot a pour valeur 2.

La loi de Khirchoff est vérifiée au sommet x_1.

c) Flot compatible :

Un flot f est compatible dans un réseau si pour tout arc $u = (x, y), 0 \leq f(u) \leq c(u)$, autrement dit pour chaque arc u, le flux qui le traverse ne dépasse pas sa capacité.[6]

Le graphe de la Figure 2.7 montre un flot compatible.

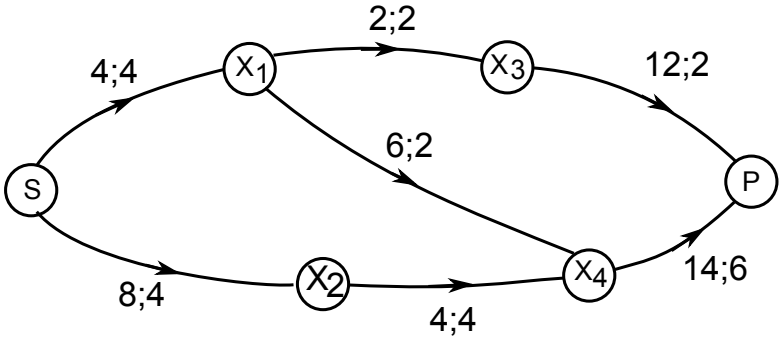


FIGURE 2.7 – Flot compatible.

d) Flot complet :

Un flot est complet si pour tout chemin allant de la source au puis il y a au moins un arc saturé, c'est-à-dire : le flux qui le travers est égale à sa capacité $\big(f(u) = c(u)\big)$.[6]

Dans le réseau de la Figure 2.7 , on a 3 chemins qui mènent de s à pour lesquels on a au moins un arc saturé. Le flot est complet.

2.4 Problème d'ordonnancement :

La gestion d'un projet composé de plusieurs tâches, présente de grands problèmes quand à l'établissement d'un calendrier du déroulement et du contrôle de leurs exécution.

C'est ainsi que sont apparus les problèmes d'ordonnancement dans la planification des projets, et ce dans le but de gagner du temps dans leur réalisation (minimiser la durée de réalisation d'un projet) compte tenu des contraintes d'antériorité reliant les déférents tâches. C'est-à-dire, une tache ne peut commencer si une ou plusieurs prennent fin.

Résoudre un problème d'ordonnancement c'est, c'est d'abord donner **l'ordre**, dans lequel doivent être exécutées les tâches, de façon à **minimiser la durée d'exécution totale du projet**, tout en satisfaisant les conditions d'antériorité.

A partir de cette planification, nous constaterons qu'un retard dans la réalisation de certaines tâches n'influencera pas la durée du projet, alors que d'autres tâches dites **critiques** retardent le projet au moindre retard local.

Deux grandes méthodes sont utilisées pour résoudre un problème d'ordonnancement de projet, on a la méthode américaine **CPM** (Critical Path Methode) avec sa variante **PART** (Program Evaluation and Review Techenique) et la méthode française MPM (méthode des potentiels). Nous nous intéressons à la méthode **PERT**.[7]

2.4.1 Présentation du réseau PERT :

La méthode PERT consiste à présenter un problème d'ordonnancement de projet d'un graphe dit " réseau PERT " ou " diagramme événement-tâche " , qu'on note $R = (X, U, D)$ ou :

- Un arc correspond à une tâche.
- La valeur d'un arc u représente la durée d'une tâche d(u).
- Un sommet est un événement signifiant le début ou la fin d'une ou plusieurs tâches.

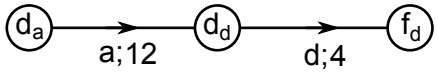


FIGURE 2.8 – Exemple d'ordonnancement des tâches.

Remarque 2.4.1. Soit $R = (X, U, d)$ un réseau pert.[7]

La succession de deux tâches consistera en premier lieu à ajouter une tâche fictive de durée nulle pour indiquer que l'événement fin de la première tâche coïncide avec l'événement début de la tâche suivant, on distinguera les deux cas suivants :

a) Premier cas :

Soit a et d deux tâches du réseau R, telles que la tâche a précède la tache d.

La représentation des deux tâches par un arc fictif. Comme le montre la Figure 2.9 :

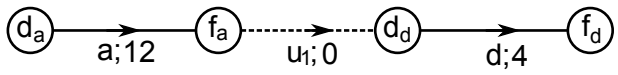


FIGURE 2.9 – Représentation de deux taches par un arc fictif.

L'arc u_1 correspond à la tâche fictive de durée nulle.

La représentation des deux tâches sans arc fictif :

Pour simplifier de graphe, on peut supprimer la tâche fictive, et obtenir ainsi la représentation ci-dessous.

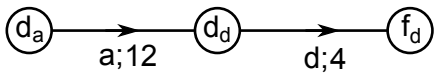


FIGURE 2.10 – Représentation de deux tâches sans arc fictif.

Les événements f_a et f_d se réalisent en même temps, on ne notera dans ce cas qu'un seul événement.

b) Deuxième cas :

Soit a, b, c et d trois tâches de R telle que :
- La tache a est précède les tâches d et c.
- La tâche b précède la tâche c.

La première représentation des tâches est donnée par la Figure 2.11 :

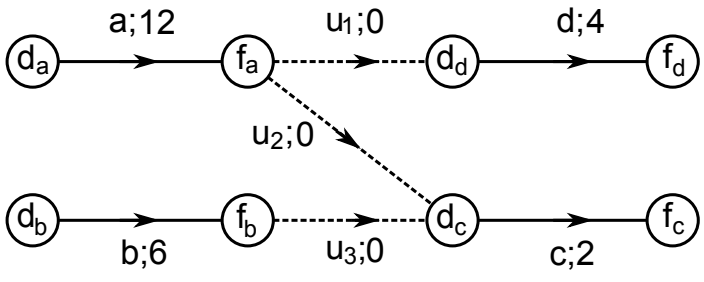


FIGURE 2.11 – Exemple de la représentation de tâches.

Dans la Figure 2.11 Les taches fictives u_1 et u_3 peuvent être supprimées alors que u_2 est nécessaire ; elle signifie que a précède c en même temps que d, tout en évitant la confusion quant à l'antériorité unique de a

rapport à d. on obtient ainsi la représentation de la Figure 2.12 :

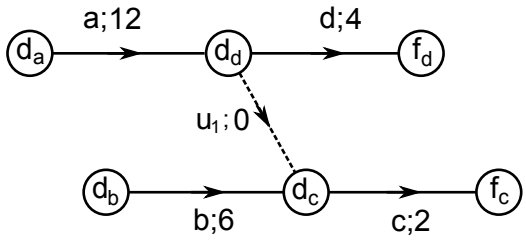


FIGURE 2.12 – Exemple de réseau pert.

Remarque 2.4.2. On rajoute deux sommets D et F au réseau pert pour représenter respectivement le début et la fin du projet. La représentation graphique est ordonnée par niveaux des sommets (des évènements).[7]

Considérons le problème suivant : Un projet est composé de 6 tâches dont les durées de réalisation et les contraintes d'antériorité (de précédence) sont données dans le tableau suivant :

Tâches	Tâches précédents	Durées des Tâches
a	-	12
b	-	6
c	b-a	2
d	a	4
e	d	8
f	c	2

Il s'agit de trouver un calendrier de déroulement des tâches, c'est à dire déterminer les dates de démarrage de chaque tâche, qui minimisent le temps totale de réalisation du projet et d'indiquer les tâches critiques.

La représentation graphique du problème est le réseau pert $R = (X, U, d)$ montré dans la Figure 2.13 :

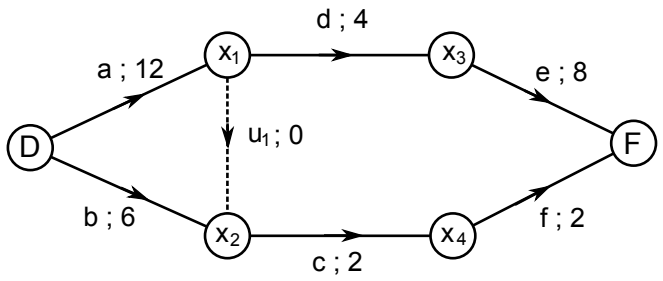


FIGURE 2.13 – Représentation de problème de calendrier de tâches.

Dans la Figure 2.13 les tâches a et b ne sont précédées par aucune tâche, alors le début de ces tâches coïncident avec le début du projet (D), de même, les tâches e et f ne sont suivis par aucune tâche, alors la fin de ces tâches coïncident avec la fin du projet (F).

2.4.2 Détermination du calendrier des dates au plus tôt et des dates au plus tard :

a) Calendrier des dates au plus tôt des événements :

Une fois qu'on aura établi le réseau **PERT**, on pourra déterminer pour chaque événement la date à laquelle il peut au plus tôt se réaliser, autrement dit, la date de début au plus tôt de chaque tâche.

On considère la date début de l'événement D (début du projet) égale à 0. Un événement x ne se réalise que si tous les avènements le procédant se sont réalisées. On calcule alors la longueur maximale du chemin aboutissant à x. La date au plus tôt d'un événement x notée t_x se calcul comme suit :

$$t_x = max[t_y + d(y,x)/y \in \Gamma_G^-(x)]\,;$$

Le max étant pris sur les prédécesseurs de x

Dans le réseau de la Figure 2.13 ; Les dates au plus tôt des événements se calcule comme suit :

- $t_D = 0$
- $t_{x1} = t_D + d(a) = 12$
- $t_{x2} = max[t_D + d(b); t_{x1} + d(u)] = max[6; 12] = 12$
- $t_{x3} = t_{x1} + d(d) = 16$
- $t_{x4} = t_{x2} + d(c) = 14$
- $t_F = max[t_{x3} + d(e); t_{x4} + d(f)] = max[24; 16] = 24$

Remarque 2.4.3. Pour le sommet F fin du projet, t_f correspond à la durée minimale de réalisation du projet.[7]

b) Calendrier des dates de début au plus tôt des tâches :

La date de début au plus tôt de la tache a est égale à la date au plus tôt de l'événement d'où elle est issue, on la note T_a.[7]

$T_a = t_x$ si la tache a est issue de l'événement x. Comme montre la Figure 2.14 :

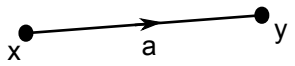


FIGURE 2.14 – Exemple d'une Tache a issue de l'avenement x.

Dans le réseau pert de la Figure 2.13, les dates de début au plus tôt des taches sont : $T_a = t_D = 0, T_b = t_D = 0, T_c = t_{x2} = 12, T_d = t_{x1} = 12, T_e = t_{x3} = 16, T_f = t_{x4} = 14$.

Pour chaque événement, on a obtenu la date à laquelle il peut se réaliser au plus tôt, ce qui permet la planification des taches pour une plus petite durée possible du projet, soit 12 jours.

la Figure 2.15 présente un réseau avec les dates au plus tôt des événements.

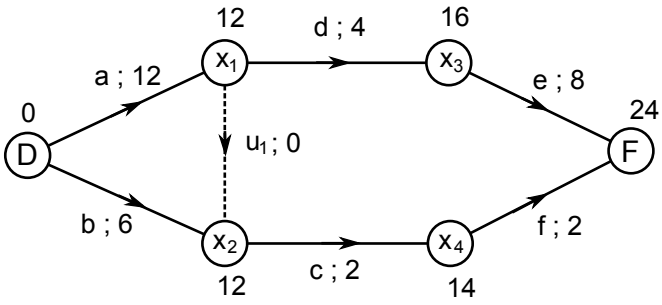


FIGURE 2.15 – Réseau avec les dates au plus tôt des évenement.

c) Calendrier des dates au plus tard des événements :

De la même manière on va déterminer la date à laquelle un événement peut au plus tard se réaliser sans retarder la fin du projet dans les délais prévus. Pour cela connaissant la date de fin du projet on pourra déterminer la date au plus tard à laquelle les événements peuvent se réaliser.

Dans le réseau de la Figure 2.15, l'événement F se réalise à la date 24, cela signifie que la tâche e de durée 8 doit démarrer au plus tard à la date $24 - 8 = 16$.

Si un événement a plusieurs successeurs, on calculera le minimum de la différence entre la date au plus tard de ces événements moins la durée des tâches qui y aboutissent. On note la date au plus tard d'un événement par t_x^*,

$t_x^* = t_f$ Pour le sommet fin du projet,

$t_x^* = Min[t_y^* - d(x,y)/y \in \Gamma_G^+]$;

Le min étant pris sur les successeurs y de x.

Dans le réseau pert de la Figure 2.13 ; Les dates au plus tard des événements se calcul comme suit :

- $t_F^* = t_F = 24$
- $t_{x3}^* = t_F - d(e) = 24 - 8 = 16$
- $t_{x4}^* = t_F - d(e) = 24 - 2 = 22$
- $t_{x2}^* = t_{x4} - d(c) = 16$
- $t_{x1}^* = min[t_{x2}^* - d(u_1); t_{x3}^* - d(d)] = min[20; 12] = 12$
- $t_D^* = min[t_{x1}^* - d(a); t_{x4}^* - d(b)] = min[0; 14] = 0$

d) Calendrier des dates de début au plus tard des tâches :

La date de début au plus tard de la tâche a est égale à la date au plus tard de l'événement auquel elle aboutit, diminué de la durée de la tâche T_a^*

$T_a^* = t_y^* - d(x,y)$ si la tâche a va du sommet x au sommet y.(Voir Figure 2.14)[7]

Dans le réseau pert de la Figure 2.13, Les dates de début au plus tard des tâches sont calculées comme suit :

- $T_f^* = t_F^* - d(f) = 24 - 2 = 22$
- $T_e^* = t_F^* - d(e) = 16$
- $T_d^* = t_{x3}^* - d(d) = 12$
- $T_c^* = t_{x4}^* - d(c) = 20$
- $T_b^* = t_{x2}^* - d(b) = 16$
- $T_a^* = t_{x1}^* - d(a) = 0$

La Figure 2.16 montre un réseau pert avec des dates au plus tard des événements.

2.4.3 Analyse et identification des tâches critiques :

Une fois la date au plus tard et la date au plus tôt de chaque événement calculé, on peut analyser la situation. La date au plus tôt fournit la date planifiée pour chaque événement et les dates au plus tard indiquent de combien un événement peut être retardé sans retarder le projet, ceci permet d'identifier les tâches critiques.

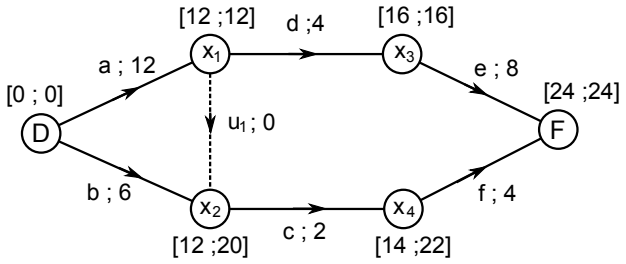


FIGURE 2.16 – Réseau pert avec des dates au plus tard des événements.

Une tâche i correspond à l'arc (x, y) peut voir sa durée augmentée d'un délai $\delta(i) = T_y^* - t_x^* - d(i) = T_i^* - T_i$ sans que ce retard ne se répercute nécessairement sur la durée de réalisation du projet.[7]

a) Tâche critique :

Une tâche est dite " critique " si tout retard dans son exécution se répercute automatiquement sur la durée de réalisation du projet, en d'autres termes, une tâche est critique si sa date de début et sa date de fin sont critiques, et si la différence entre la date de début et la date de fin est égale à la durée de la tâche.

Autrement dit, une tâche i est critique si $\delta(i) = 0$.[7]

Remarque 2.4.4. Il est à noter que, sur le plus long chemin de D à F, toutes les tâches sont critiques ; ce chemin est dit alors chemin critique.[7]

Dans le réseau de la Figure 2.16 : $T_a = T_a^* = 0; T_d = T_d^* = 12; T_e = T_e^* = 16$, donc les tâches a, d et e sont critiques.

Le chemin critique est donc le chemin (D, x_1, x_3, F), de longueur 24 (longueur maximale). Comme le montre le réseau de la Figure 2.17.

b) Calcul des marges et de l'intervalle de flottement :

b.1) Intervalle de flottement :

Chaque tâche a une date de début au plus tôt, et une date de début au plus tard, le démarrage de cette tâche peut intervenir entre ces deux dates sans compromettre la date de fin du projet. L'intervalle de flottement d'un événement est égale à la différence entre la date de début au plus tôt et la date de début au plus tard de la tâche.[7]

Dans le réseau de la Figure 2.16 :

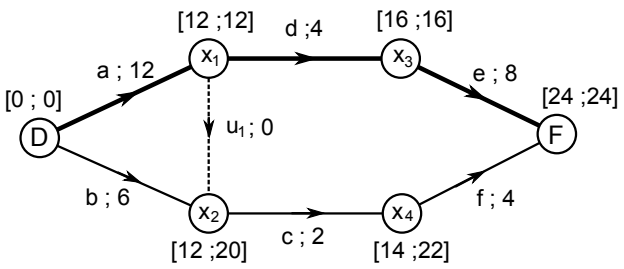


FIGURE 2.17 – Chemin critique.

- La tâche " f " a une date au plus tôt égale à 14, et une date au plus tard égale à 22, donc l'intervalle de flottement de l'événement début de f est $[14; 22]$.
- La tâche " c " débute au plus tôt le 12^{ime} jour, et au plus tard le 20^{ime} jour du début du projet, donc l'intervalle de flottement est de $[12, 20]$, ce qui fait que la tâche c peut débuter entre le 12^{ime} jour et le 20^{ime} jour sans retarder la date de fin du projet.

b.2) Les marges :

Le responsable du projet peut se pencher aussi sur le degré de liberté dont il dispose pour éventuellement augmenter la durée d'une tâche sans compromettre la durée totale du projet, On distingue trois types de marges :

- La marge libre :

Notée $M_l(u)$, c'est le retard maximum que l'on peut apporter au démarrage d'une tâche sans perturber la date de réalisation au plus tôt de l'événement suivant : Soit une tâche i représentée par l'arc $u = (x, y)$.[7]

$M_l(i) = t_y - t_x - d(i)$

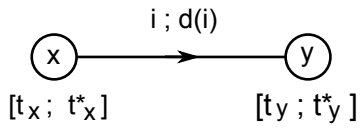


FIGURE 2.18 –

- La marge totale :

Notée $M_t(u)$, c'est le retard maximum qu'on peut apporter au démarrage d'une tâche sans perturber la date fin du projet.

Soit une tache i, représentée par l'arc $u = (x, y)$. (Voir Figure2.18)[7]

$$M_t(i) = t_y^* - t_x - d(i) = T_a^* - T_a$$

- La marge certaine :

Notée $M_c(u)$, c'est le retard maximum que l'on peut apporter au démarrage d'une tâche sans perturber la réalisation au plus tôt de l'événement suivant bien que l'événement précédent n'a été réaliser qu'a sa date limite.

Soit une tâche i représentée par l'arc $u = (x, y)$. (Voir Figure2.18)[7]

$$M_c(i) = t_y - t_x^* - d(i)$$

Conclusion :

Dans ce chapitre, nous avons présenté quelques problèmes de la théorie des graphes. Par la suite nous allons détailler quelques techniques de modélisation de ces problèmes, afin de faciliter la résolution de ces dernières.

Chapitre 3

Méthodes de résolution

Dans ce chapitre, nous allons donner les méthodes et les algorithmes adéquates aux problèmes posés dans le chapitre précédent, où on a décrit le principe et le procédé de déroulement de ces algorithmes, plus une application illustratif pour chaque méthode et algorithme en raison de faciliter la compréhension de l'utilisateur.

3.1 Algorithme de Walsh and Powelle pour la coloration des sommets d'un graphe :

1. **Principe :**
   La coloration des sommets d'un graphe $G = (X, E)$ consiste à établir une partition de l'ensemble des sommets X de G en k-classes $(X_1, X_2, \ldots, X_k)$, de telle façon que deux sommets d'une même classe ne soient pas adjacents, et les sommets d'une classe sont coloriés de la même couleur. Autrement dit, deux sommets adjacents ne sont pas colorés de la même couleur.
1. **Procédé :**
   On commence par établir une liste ordonnée des sommets (ordonner les sommets suivant l'ordre décroissant de leur degré).
   Tant qu'il existe des sommets à colorier, exécuter les actions suivantes :
   (1) Choisir une nouvelle couleur appelée couleur d'usage ;
   (2) Chercher dans la liste des sommets le premier sommet non coloré et le colorer avec la couleur d'usage ;
   (3) Examiner tour à tour, dans l'ordre de la liste, tous les sommets non coloriés et colorier chaque sommet non adjacent à un sommet déjà coloré avec la couleur d'usage.

Application de l'algorithme de Walsh and Powell au graphe G de la Figure 3.1 :

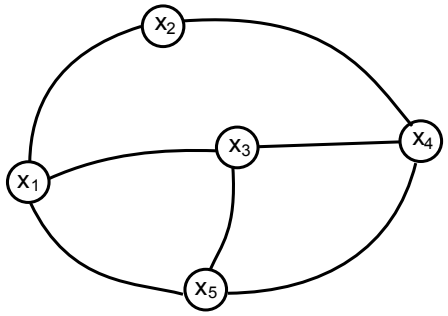


FIGURE 3.1 – Graphe initiale.

(1) On ordonne les sommets selon l'ordre décroissant de leur degré comme suit :

Sommet x_i	x_1	x_3	x_4	x_5	x_2
degré $d_G(x_i)$	3	3	3	3	2

(2) Colorier le sommet x_1 par la couleur Verte (V), puis cherchons dans l'ordre de la liste, le sommet non coloré, qui n'est adjacent à aucun sommet coloré avec la couleur verte, soit le sommet x_4. Comme indiqué sur la Figure 3.2 :

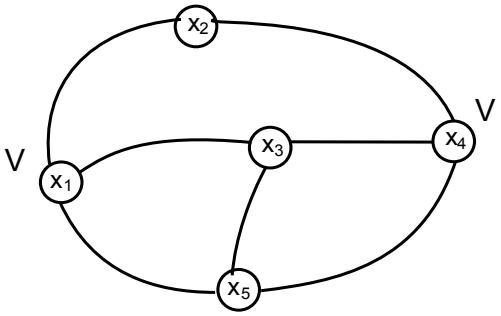


FIGURE 3.2 – Itération (1) de Walsh and Powell.

(3) Colorons ensuite le sommet x_3 par la couleur Rouge (R), Puis cherchons dans l'ordre du tableau, le sommet non coloré, qui n'est adjacent à aucun sommet coloré avec la couleur rouge, soit le sommet x_5. Comme indiqué sur la Figure 3.3 :

(4) Il reste le sommet x_3, colorons-le avec la couleur Bleu (B).
Le nombre chromatique $\Delta(G)$ est égale à $= 3$.
Le graphe G est donc 3-coloriable, il peut être partitionner en trois classes : $X_1 = \{x_1, x_4\}$; $X_2 = \{x_2, x_5\}$; $X_3 = \{x_3\}$. Comme le montre la Figure 3.4 :

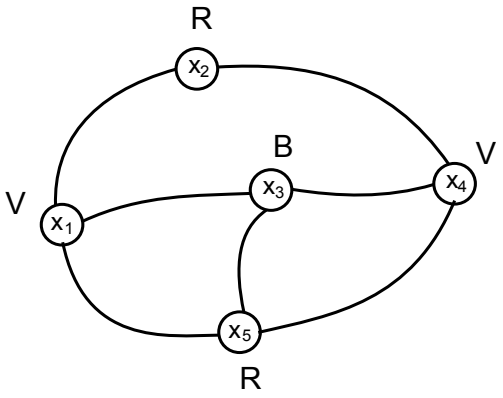


FIGURE 3.3 – Itération (2) de Walsh and Powell.

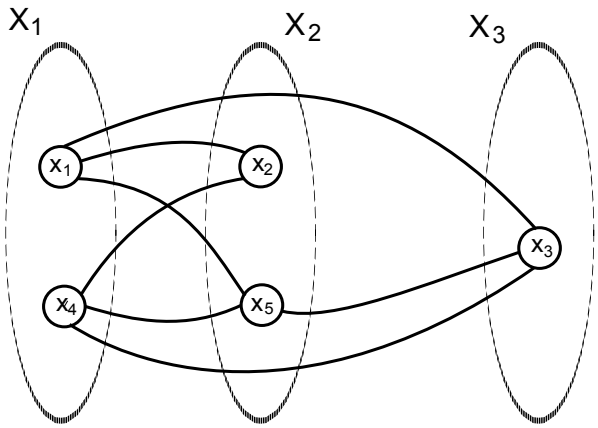


FIGURE 3.4 – Partitionnement du graphe G en un graphe 3-Coloriable.

3.2 Algorithme de recherche d'un plus court chemin :

Plusieurs algorithmes existent pour résoudre les problèmes de recherche des plus courts chemins dans un réseau, nous ne présenterons que les méthodes qui paraissent les plus performantes.

3.2.1 Algorithme de Bellman :

On applique cet algorithme pour la recherche d'une arborescence de plus courts chemins dans un réseau $R = (X, U, d)$ sans circuit.

(1) **Principe :**
L'idée de l'algorithme de Bellman, est de calculer de proche en proche, l'arborescence des plus courtes distances, issue du sommet s à un sommet donné p.

On calcule la plus courte distance du sommet s à y, que si on a déjà calculé les plus courtes distances du sommet s à tous les prédécesseurs du sommet y.

(2) **Enoncé :**

Données : Un réseau $R = (X, U, d)$ sans circuit avec $d(u) \in R$. Résultat : Arborescence de plus courtes distances A.

(1) Initialisation :

Soit s un sommet de X, on pose $S = \{s\}$ et $\pi(s) = 0$ $A = \emptyset$.

(2) Chercher un sommet hors de S dont tous les prédécesseurs sont dans S.

– Si un tel sommet n'existe pas ; terminer.

Dans ce cas soit $S = X$, ou le sommet s n'est pas une racine dans R.

– Si un telle sommet existe ; aller en (2).

(2) On pose : $\pi(x) = Min\{\pi(I(u)) + d(u)\}$. soit u' l'arc pour lequel $\pi(x) = \pi(I(u')) + d(u')$ $A := A \cup \{u'\}; S := S \cup \{x\}$ Aller à (1).

Appliquons l'algorithme de Bellman au Réseau $R = (X, U, d)$ de la Figure 3.5 :

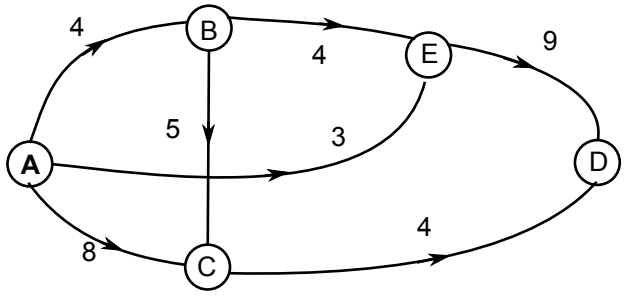


FIGURE 3.5 – Réseau R.

1. **Initialisation :**

   Soit A un sommet de X, On pose $S = \{A\}; \pi(S) = 0; A = \emptyset$

2. **Les itérations :**

- Itération 1 :

Les sommets B est hors de S et tous ses prédécesseurs sont dan S, alors :

$\pi(B) = \pi(A) + d(A, B) = 0 + 4 = 4$

$\Rightarrow u' = (A, B)$ $S = S \cup \{B\} = \{A, B\}$; $A = A \cup \{(A, B)\} = \{(A, B)\}$.

- Itération 2 :

Les sommets E et C sont hors de S et tous leurs prédécesseurs sont dans S, alors :

$\pi(E) = \text{Min}\{\pi(A) + d(A, E); \pi(B) + d(B, E)\} = \text{Min } \{0 + 5; 4 + 4\}$

$=\text{Min}\{5, 8\} = 5, \Rightarrow u'_1 = (A, E)$.

$\pi(C) = \text{Min } \{\pi(A) + d(A, C); \pi(B) + d(B, C)\}$

$=\text{Min}\{0 + 8; 4 + 5\} = \text{ Min } \{8, 9\} = 8 \Rightarrow u'_2 = (A, C)$.

$S = S \cup \{E, C\} = \{A, B, E, C\}$; $A = A \cup \{(A, E), (A, C)\} = \{(A, B), (A, E), (A, C)\}$.

- Itération 3 :

Le sommet D est hors de S et tous leur prédécesseur sont dan S, alors :

$\pi(D) = \text{Min } \{\pi(E) + d(E, D), \pi(C) + d(C, D)\} = \text{Min } \{5 + 9; 8 + 4\} = \text{Min}\{14; 12\} = 12$,

$\Rightarrow u' = (C, D)$

$S = S \cup \{D\} = \{A, B, C, D, E\}$; $A = A \cup \{(C, D)\} = \{(A, B), (A, E), (A, C), (C, D)\}$.

On a : S=X ; Terminer.

On obtient donc, l'arborescence des plus courtes distances (Voir la Figure 3.6) :

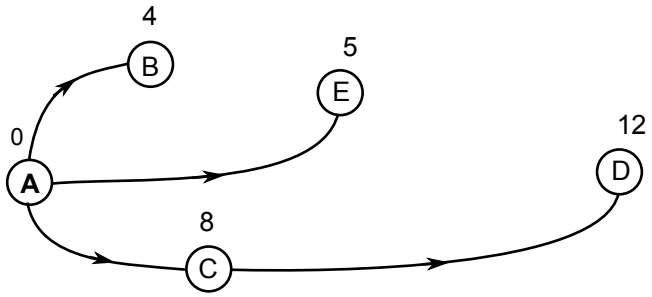


FIGURE 3.6 – Arborescence des plus courtes distances.

3.2.2 Algorithme de Dijkstra :

On applique cet algorithme pour déterminer une arborescence des plus courtes distances sur un réseau $R = (X, U, d)$, ou les longueurs des arcs sont positives ou nulles $d(u) \geq 0, \forall u \in U$.

1. **Principe :**
   L'idée de l'algorithme de Dijkstra est de calculer de proche en proche, l'arborescence des plus courtes distances, issue de sommet s à un sommet donné p.
   Une particularité de cet algorithme est que les distances n'introduises dans l'ordre croissant.
2. **Enoncé :**
   Données : Un réseau $R = (X, U, d)$ sans circuit avec $d(u) \geq 0, \forall u \in U$
   Résultat : Arborescence de plus courtes distances A.
   (0) Initialisation :
   Soit s un sommet de X.
   On pose : $S = \{s\}, \pi(s) = 0, \pi(x) = \infty, \forall x \in X/\{s\}, A(s) = \emptyset$ et $\alpha = s^*$.
   $/^*\alpha$ est le dernier sommet introduit dans S.
   (1) Examiner tous les arcs u dont l'extrémité initiale est égale à $\alpha (I(u) = \alpha)$ et l'extrémité terminale n'appartient pas S $(T(u) = y$ avec $y \notin S$
   Si $\pi(\alpha) + d(u) < \pi(y)$, on pose $\pi(y)\pi(\alpha) + d(u)$ et $A(y) = u$ aller en (2).

(2) Choisir un sommet $z \notin S$ tel que $\pi(z) = \text{Min } \{\pi(y)/y \notin S\}$
- Si $\pi(z) = \infty$; Terminer. Le sommet s n'est pas une racine dans R.
- Si $S = X$; Terminer. A définit l'arborescence des plus courts chemins issus de s.
- Si $S \neq X$ Aller en (1).

Considérons le réseau de la Figure 3.7

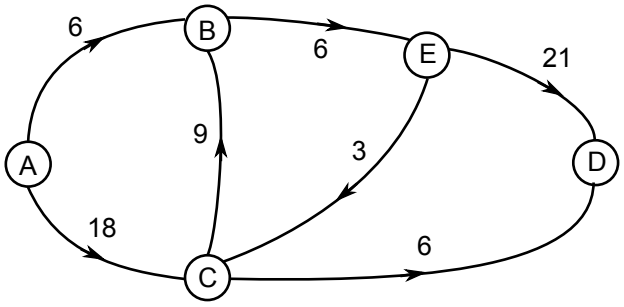


FIGURE 3.7 –

1. **Indication :** On met [ ] pour mentionner que le sommet est dans S.
2. **Initialisation :**

   Soit A un sommet de X.

   On pose : $S = \{A\}, A(A) = \emptyset, \alpha = A$

x	A	B	C	D	E
$\pi(x)$	[0]	∞	∞	∞	∞

3. **Les itérations :**

   - Itération 1 :

   On examine les arcs (A,B) et (A,C).
   $\pi(A) + d((A,B)) = 0 + 6 = 6 < \pi(B) = \infty$,
   $\Rightarrow \pi(B) = 6; A(B) = (A,B)$
   $\pi(A) + d((A,C)) = 0 + 18 = 18 < \pi(C) = \infty$,
   $\Rightarrow \pi(C) = 18; A(C) = (A,C)$

x	A	B	C	D	E
$\pi(x)$	[0]	6	18	∞	∞

   $\pi(z) = \text{Min}\{\pi(y)/y \notin S\} = \text{Min } \{\pi(B), \pi(C), \pi(D), \pi(E)\}$
   $= \text{Min } \{6, 18, \infty, \infty\} = 6 = \pi(B)$; Donc $z = B$.
   On pose alors : $S = \{A, B\} \neq X; A = \{(A,B)\}; \alpha = B$.

- Itération 2 :

On examine l'arc (B,E) $\pi(B) + d((B,E)) = 6 + 6 = 12 < \pi(E) = \infty$,
$\Rightarrow \pi(E) = 12; A(E) = (B,E)$.

x	A	B	C	D	E
$\pi(x)$	[0]	[6]	18	∞	12

$\pi(z) = \text{Min}\{\pi(y)/y \notin S\} = \text{Min}\{\pi(C), \pi(D), \pi(E)\}$
$= \text{Min}\ \{18, \infty, 12\} = 12 = \pi(E)$; Donc $z = E$.
On pose alors : $\alpha = E, S = \{A, B, E\} \neq X; A = \{(A,B), (B,E)\}$;

- Itération 3 :

On examine les arcs (E,C) et (E,D)
$\pi(C) + d((E,C)) = 12 + 3 = 15 < \pi(C) = \infty$,
$\Rightarrow \pi(C) = 15; A(C) = (E,C)$.
$\pi(E) + d((E,D)) = 12 + 21 = 33 < \pi(D) = \infty$,
$\Rightarrow \pi(D) = 33; A(D) = (E,D)$.

x	A	B	C	D	E
$\pi(x)$	[0]	[6]	15	33	[12]

$\pi(z) = \text{Min}\{\pi(y)/y \notin S\} = \text{Min}\ \{\pi(C), \pi(D)\}$
$= \text{Min}\{15, 33\} = 15 = \pi(C)$; Donc $z = D$.
On pose alors : $\alpha = C; S = \{A, B, E, C\} \neq X$;
$A = \{(A,B), (B,E), (E,C)\}$;

- Itération 4 :

On examine l'arc (C,D)
$\pi(D) + d((C,D)) = 21 < \pi(D) = 33$,
$\Rightarrow \pi(D) = 33; A(D) = (C,D)$.

x	A	B	C	D	E
$\pi(x)$	[0]	[6]	[15]	21	[12]

$\pi(z) = \text{Min}\{\pi(y)/y \notin S\} = \text{Min}\ \{\pi(D)\} = \pi(D) = 21$;
Donc $z = C$.
On pose alors : $\alpha = C, S = \{A, B, E, C, D\} \neq X$;
$A = \{(A,B), (B,E), (E,C), (C,D)\}$;
Enfin : $S = X$ Terminer.

x	A	B	C	D	E
$\pi(x)$	[0]	[6]	[15]	[21]	[12]

Les $\pi(x)$ sont les plus courtes distances. L'arborescence des plus courts chemins, issue du sommet A, dans le réseau est montré par la Figure 3.8 :

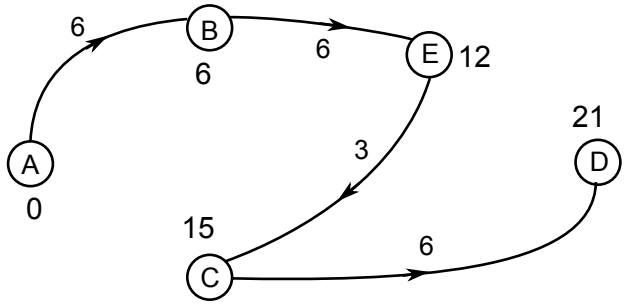


FIGURE 3.8 – Arborescence des plus courts chemins.

3.2.3 Algorithme générale de Ford :

On applique l'algorithme générale pour la recherche d'un plus court chemin sur un réseau quelconque avec $d(u) \in R$. Cet algorithme permet soit :

- De mettre en évidence un circuit absorbant si celle-ci existe.
- De déterminer une arborescence des plus courts chemins de racine s dans un réseau s'il ne contient pas de circuit absorbant.

1. **principe :**
   Le principe de l'algorithme générale consiste à améliorer une arborescence réalisable (initiale) (X, A) de racine s jusqu'à l'obtention d'une arborescence optimale des plus courts chemins, issue de s si celle-ci existe.
2. **Enoncé :**
   Données : Un réseau $R = (X, U, d)$ sans circuit avec $d(u) \in R$.
   Résultat : Arborescence de plus courtes distances A.
   (0) Initialisation :
   Soit (X, A) une arborescence de racine s dans le réseau R et $\pi(x)$ les longueurs des chemins de s à x dans dans l'arborescence $(X, A)^*/$.
   $^*/$ On utilise par exemple l'algorithme de Djikstra pour obtenir l'arborescence réalisable (X, A).
   (1) Chercher un arc (i, j) dans le réseau R, n'appartenant pas à A, tel que :
   $$\delta(u) = \pi(j) - \pi(i) - d(i, j) > 0.$$
   - Si un tel arc n'existe pas ; Terminer, (X, A) est optimale.
   - Si un tel arc existe ; Aller à (2).

(2) Tester si $(X, A \cup \{u\})$ contient un circuit
- Si oui ; examiner si ce circuit est absorbant (s'il l'est ; Terminer. Le problème n'admet pas de solution).
- Sinon ; Aller à (3).

(3) Chercher un arc $v \in A$ tel que $T(v) = j = T(u)$.
On pose : $A = A \cup \{u\}/\{v\}$.
Soit $X' = \{i\} \cup \{$ descendant de j dans l'arborescence $A\}$
On pose : $\pi(y) = \pi(y) - \delta(u)\ \forall y \in X'$ Aller en (1).

Considérons le réseau de la Figure 3.9 :

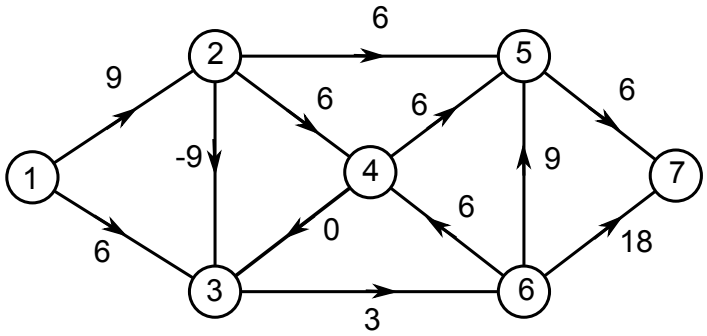


FIGURE 3.9 – Application de l'algorithme de Ford.

Ce réseau ne possède pas de circuit absorbant et le sommet 1 est une Racine ; alors ce problème admet une arborescence des plus courts chemins.

1. **Initialisation :**

   Soit $A = \{(1,2),(1,3),(3,6),(6,4),(2,5),(5,7)\}$ une arborescence initiale.

2. **Les itérations :**

- Itération 1 :

Dans le réseau R 3.10, les arcs n'appartenant pas à A sont : $(2,4),(2,3),(4,3),(4,5),(6,5),(6,7)$ tels que :

$\delta(2,3) = \pi(3) - \pi(2) - d(2,3) = 6 - 9 - 9 = 6$
$\delta(2,4) = \pi(4) - \pi(2) - d(2,4) = 0$
$\delta(4,3) = \pi(3) - \pi(4) - d(4,3) = -9$
$\delta(4,5) = \pi(5) - \pi(4) - d(4,5) = -6$
$\delta(6,5) = \pi(5) - \pi(6) - d(6,5) = -3$
$\delta(6,7) = \pi(7) - \pi(6) - d(6,7) = -15$

L'arc $u = (2,3) \notin A$ et $\delta(u) > 0$ et $A \cup \{(2,3)\}$ ne contient pas de circuit, donc l'arc $u = (2,3)$ entre dans l'arborescence.

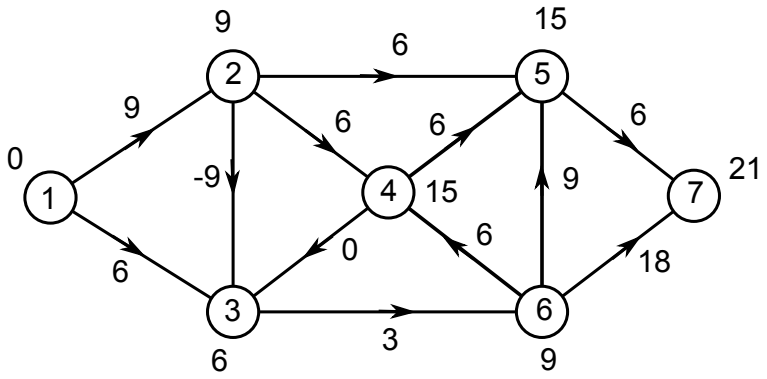


FIGURE 3.10 – Arbrescence Initiale.

Soit l'arc $v = (1,3) \in A, T(v) = 3$ donc l'arc $(1,3)$ sort de l'arborescence.

Soit $X' = \{3\} \cup \{$ descendants de sommet 3 dans l'arborescence A$\} = \{3,4,6\}$

On pose :

$$\pi(3) = \pi(3) - \delta(2,3) = 6 - 6 = 0$$
$$\pi(6) = \pi(6) - \delta(2,3) = 9 - 6 = 3$$
$$\pi(4) = \pi(4) - \delta(2,3) = 15 - 6 = 9$$

On obtient ainsi la nouvelle arborescence :

$A = A \cup \{u\}/\{v\} = \{(1,2),(2,5),(5,7),(2,3),(3,6),(6,4)\}$.

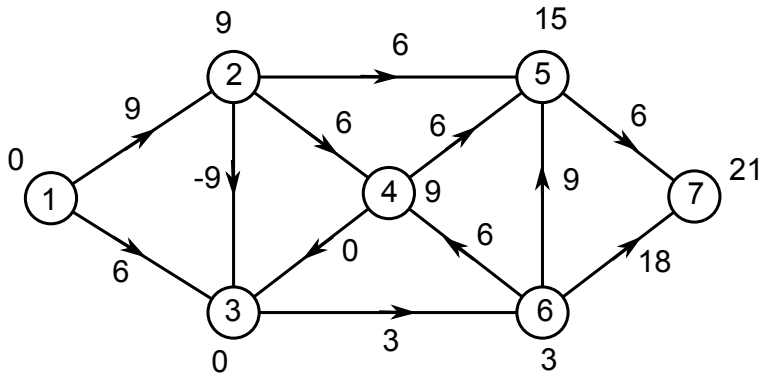


FIGURE 3.11 – Iteration 1 (Algorithme de Ford).

- Itération 2 :

Dans le réseau R, les arcs n'appartenant pas à la nouvelle arborescence A sont : $(2,4),(1,3),(4,3),(4,5),(6,5),(6,7)$ tels que :

$\delta(1,3) = \pi(3) - \pi(1) - d(1,3) = 0 - 0 - 6 = -6$

$\delta(2,4) = \pi(4) - \pi(2) - d(2,4) = -6$

$\delta(4,3) = \pi(3) - \pi(4) - d(4,3) = -9$

$\delta(4,5) = \pi(5) - \pi(4) - d(4,5) = 0$
$\delta(6,5) = \pi(5) - \pi(6) - d(6,5) = 3$
$\delta(6,7) = \pi(7) - \pi(6) - d(6,7) = 0$

L'arc $u = (6,5) \notin A$ et $\delta(u) > 0$ et $A \cup \{(6,5)\}$ ne contient pas de circuit, donc l'arc $u = (6,5)$ entre dans l'arborescence.

Soit l'arc $v = (2,5) \in A, T(v) = 5$ donc l'arc (1,3) sort de l'arborescence.

Soit $X' = \{5\} \cup \{$ descendants de sommet 5 dans l'arborescence A$\}$

$= \{5,7\}$

On pose :

$\pi(5) = \pi(5) - \delta(6,5) = 15 - 12 = 12$
$\pi(7) = \pi(7) - \delta(6,7) = 18$

On obtient ainsi la nouvelle arborescence :

$A = A \cup \{u\}/\{v\}$
$= \{(1,2),(6,7),(5,7),(2,3),(3,6),(6,4)\}$.

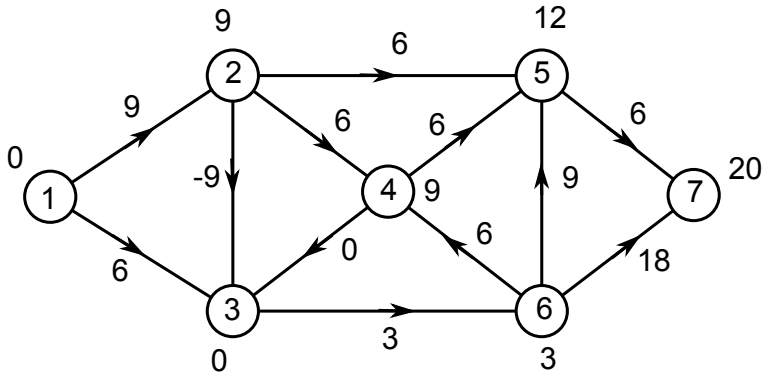


FIGURE 3.12 – Itération 2 (Algorithme de Ford).

- Itération 3 :

Dans le réseau R, les arcs n'appartenant pas à la nouvelle arborescence A sont : $(2,4),(1,3),(4,3),(4,5),(2,5),(6,7)$ tels que :

$\delta(1,3) = \pi(3) - \pi(1) - d(1,3) = -6$
$\delta(2,4) = \pi(4) - \pi(2) - d(2,4) = -6$
$\delta(4,3) = \pi(3) - \pi(4) - d(4,3) = -9$
$\delta(4,5) = \pi(5) - \pi(4) - d(4,5) = -3$
$\delta(2,5) = \pi(5) - \pi(2) - d(2,5) = -3$
$\delta(6,7) = \pi(7) - \pi(6) - d(6,7) = -3$

Il n'existe aucun arc u n'appartenant pas à A et $\delta(u) > 0$. D'où l'arborescence déjà trouvé est optimale.

L'arborescence optimale est : $A = \{(1,2),(6,5),(5,7),(2,3),(3,6),(6,4)\}$.

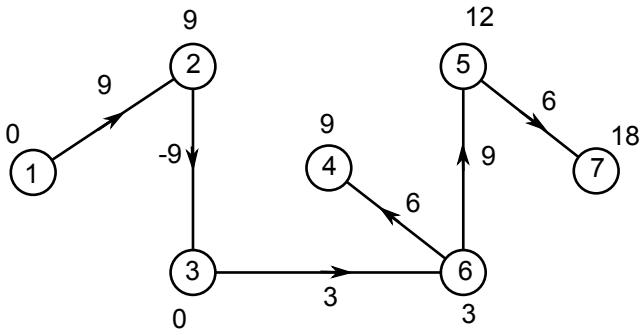


FIGURE 3.13 – Arborescence obtenue par application de l'algorithme de Ford.

3.3 Méthodes de résolution d'un problème de transport :

3.3.1 Méthode du coin nord-ouest :

1. **Principe :**
   La méthode coin nord-ouest permet d'obtenir un plan basique initial du problème de transport.
2. **Procédé :**
   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.

Considérons le problème de tableau ci-après :

$A_i \backslash B_j$	B_1	B_2	B_3	B_4	a_i
A_1	1	4	2	3	40
A_2	5	1	3	4	42
A_3	2	5	1	2	70
b_j	24	46	56	26	152

On a :

$x_{11} = min\{40, 24\} = 24$; x_{12} =min$\{40 - 24, 46\} = 16$;

$x_{22} = min\{42, 46 - 16\} = 30$; x_{23} =min$\{42 - 30, 56\} = 12$;

$x_{33} = min\{70, 56 - 12\} = 44$; x_{12} =min$\{70 - 44, 26\} = 26$.

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

$A_i \backslash B_j$	B_1	B_2	B_3	B_4	a_i
A_1	1 **(24)**	4 **(16)**	2	3	40
A_2	5	1 **(30)**	3 **(12)**	4	42
A_4	2	5	1 **(44)**	2 **(26)**	70
b_j	24	46	56	26	152

3.3.2 Méthode de l'élément minimal :

1. **Principe :**
   Cette méthode donne en général un plan basique initial plus proche du plan basique optimal que celui que celui obtenu par la méthode du coin nord-ouest.
2. **Procédé :**
   Choisir au début une case (i_1, j_1) qui correspond à l'élément $c_{i_1 j_1}$ tel que :
   (1) $c_{i_1 j_1} = \min\{c_{ij}, 1 \leq i \leq m, 1 \leq j \leq n\}$.
   (2) On posera $x_{i_1 j_1} = \min\{a_{i_1}, b_{j_1}\}$ dans la case (i_1, j_1).
   i) Si $x_{i_1 j_1} = a_{i_1}$, on exclut la ligne i_1 et on remplace le nombre b_{j_1} par $(b_{j_1} - x_{i_1 j_1})$.
   ii) Si $x_{i_1 j_1} = b_{j_1}$, on exclut la colonne j_1 et on remplace le nombre a_{i_1} par le nombre $(a_{i_1} - x_{i_1 j_1})$.
   (3) On refait la même procédure avec le tableau réduit
   Ce processus sera répété $(m + n - 1)$ fois et permettra de trouver $(m + n - 1)$ variables basiques du plan initiale recherché.

Cherchons le plan basique initial dans le tableau de transport ci-après avec la méthode de l'élément minimal :

$A_i \backslash B_j$	B_1	B_2	B_3	B_4	a_i
A_1	1	4	2	3	40
A_2	5	1	3	4	42
A_3	2	5	1	2	70
b_j	24	46	56	26	152

On a :
$c_{11} = \min\limits_{i,j} c_{ij} = 1$, $\quad x_{11} = \min\{40, 24\} = 24$;
$c_{22} = \min\limits_{j \neq 1} c_{ij} = 1$, $\quad x_{22} = \min\{42, 46\} = 42$;
$c_{33} = \min\limits_{i \neq 2, j \neq 1} c_{ij} = 1$, $\quad x_{33} = \min\{70, 56\} = 56$;
$c_{34} = \min\limits_{i \neq 2, j \neq 1,3} c_{ij}$, $\quad x_{34} = \min\{70 - 56, 26\} = 14$;
$c_{14} = \min\limits_{i \neq 2, j \neq 1,3} c_{ij}$, $\quad x_{14} = \min\{40 - 24, 26 - 14\} = 12$;
$c_{12} = \min\limits_{i \neq 2,3, j \neq 1,3} c_{ij}$, $\quad x_{12} = \min\{40 - 36, 46 - 42\} = 4$.

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

$A_i \backslash B_j$	B_1	B_2	B_3	B_4	a_i
A_1	1 **(24)**	4 **(4)**	2 **(12)**	3	40
A_2	5	1 **(42)**	3	4	42
A_4	2	5	1 **(56)**	2 **(14)**	70
b_j	24	46	56	26	152

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

$$Z(x) = (1 \times 24) + (4 \times 16) + (1 \times 30) + (3 \times 12) + (1 \times 44) + (2 \times 26) = 250.$$
$$Z(x') = (1 \times 22) + (4 \times 4) + (3 \times 12) + (1 \times 42) + (1 \times 56) + (2 \times 14) = 202.$$

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

3.3.3 Méthode des potentiels :

Notations à utiliser : U_B : Cases de base.

U_H : Cases hors base.

$u_i (1 \leq i \leq m), v_j (1 \leq j \leq n)$: Potentiels.

$\Delta_{ij} = u_i + v_j - c_{ij}$: Estimations des variables x_{ij}.

$$u_i + v_j - c_{ij} = 0, (i,j) \in U_B. \tag{3.1}$$

$$\Delta_{ij} = u_i + v_j - c_{ij}, (i,j) \in U_H. \tag{3.2}$$

Critère d'optimalité : Les inégalités :

$$\Delta_{ij} \leq 0, (i,j) \in U_H \tag{3.3}$$

Sont suffisantes pour l'optimalité du plan basique de transport x, elle sont aussi nécessaires dans le cas ou x est non dégénéré.

1. **Principe :**
   La méthode des potentielles permet de trouver une solution optimale du problème de transport tout en basant sur la solution basique trouvée par la méthode coin nord-ouest ou par la méthode de l'élément minimal.
2. **Procédé :**
   Soit x un plan basique de transport de départ, auquel correspond U_B.
   (1) Si le critère d'optimalité n'est pas vérifié, alors on cherche une case $(i_0, j_0) \in U_H : \Delta_{i_0 j_0} = \max \Delta_{ij}, (i,j) \in U_H$.
   (2) 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.
      i) Parmi les sommets du cycle affectés du signe $(-)$, on choisit celui ou la variable x_{ij} est minimale et on pose : $\theta^0 = \min x_{ij} = x_{i_1 j_1}$.
      ii) Pour les sommets affectés du signe $(+)$, on ajoute aux variables x_{ij} la quantité θ^0 et on soustrait la même quantité des variables x_{ij}, correspondantes aux sommets affectés de signe $(-)$.
      iii) Toutes les autres variables resteront inchangées
   (3) On obtient ainsi un nouveau plan basique de transport $\bar{x}$, avec un nouveau ensemble basique $\bar{U} = \{U_B \setminus (i_1, j_1)\} \cup (i_0, j_0)$.
   On répète cette itération jusqu'à ce que le critère d'optimalité soit vérifié.

On résout par la méthode des potentiels le problème de transport, présenté dans le tableau initiale de la méthode coin nord-ouest :

a) En commençant par le plan basique x trouvé dans l'exemple de coin nord-ouest.

b) En commençant par le plan basique x' trouvé dans l'exemple de l'élément minimale.

Dressons le tableau de transport avec les variables basiques x trouvé dans l'exemple du coin nord-ouest :

$A_i \backslash B_j$	B_1	B_2	B_3	B_4	a_i
A_1	1 **(24)**	4 **(16)**	2	3	40
A_2	5	1 **(30)**	3 **(12)**	4	42
A_4	2	5	1 **(44)**	2 **(26)**	70
b_j	24	46	56	26	152

En posant $u_1 = 0$, on calcul les autres potentiels par la formule (3). On placera les u_i sur une colonne à droite des a_i et les v_i sur une ligne au dessous des b_j. En suite en utilisant la formule (5), on trouve 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 :

$A_i \backslash B_j$	B_1	B_2	B_3	B_4	a_i	u_i
A_1	1 **(24)**	4 **(16)**	2 4	3 4	40	0
A_2	5 -7	1 **(30)**	3 **(12)**	4 0	42	-3
A_4	2 -6	5 -6	1 **(44)**	2 **(26)**	70	-5
b_j	24	46	56	26	152	
v_j	1	4	6	7		

Le plan basique initiale n'est pas optimal, car :

$\Delta_{i_0 j_0}$=max$\Delta_{ij} = \Delta_{13} = 8 > 0$.

A l'aide de la case $(1,3)$ on construit le cycle $(1,3) \to (2,3) \to (1,2) \to (1,3)$, on aura alors :

$\theta^0 = x_{i_1 j_1}$=min$\{16, 12\} = 12$.

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

$\bar{x}_{13}= x_{13} + \theta^0 = 12; \bar{x}_{22}= x_{22} - \theta^0 = 42; \bar{x}_{12}= x_{12} - \theta^0 = 4$.

$\bar{x}_{23}= x_{23} - \theta^0 = 0$.

Les résultats obtenus $\bar{x}_{ij}$ resteront inchangées.

On commencera donc une nouvelle itération avec le tableau suivant :

$A_i \backslash B_j$	B_1	B_2	B_3	B_4	a_i	u_i
A_1	1 **(24)**	4 **(4)**	2 **(12)**	3 0	40	0
A_2	5 -7	1 **(42)**	3 -4	4 -4	42	-3
A_4	2 -2	5 -2	1 **(44)**	2 **(26)**	70	-1
b_j	24	46	56	26	152	
v_j	1	4	6	7		

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

$x^0_{11} = 24, x^0_{12} = 4, x^0_{13} = 12, x^0_{14} = 0, x^0_{21} = 0, x^0_{22} = 42, x^0_{23} = 0,$

$x^0_{24} = 0, x^0_{31} = 0, x^0_{32} = 0, x^0_{33} = 22, x^0_{34} = 26$, est optimale

et Z^0=min $Z(x) = Z(x^0) = 202$.

Dressons le tableau de transport avec les variables basiques x' trouvé dans l'exemple précédent de l'élément minimal :

$A_i \backslash B_j$	B_1	B_2	B_3	B_4	a_i
A_1	1 **(24)**	4 **(4)**	2 **(12)**	3	40
A_2	5	1 **(42)**	3	4	42
A_4	2	5	1 **(56)**	2 **(14)**	70
b_j	24	26	56	26	152

En posant $u_1 = 0$, on calcule les autres potentiels par la formule (3.5) ci-après :

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

$v_4 = c_{14} - u_1 = 3 - 0 = 3, u_2 = c_{22} - v_2 = 1 - 4 = -3,$

$u_3 = c_{34} - u_4 = 2 - 3 = -1, v_3 = c_{33} - u_3 = 1 - (-1) = 2$

Calculons les estimations δ_{ij} :

0n a :

$$\Delta_{ij} = 0, (i,j) \in U_B, \tag{3.4}$$

$$\Delta_{ij} = u_i + v_j - c_{ij}, (i,j) \in U_H. \tag{3.5}$$

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

$\Delta_{23} = u_2 + v_3 - c_{23} = -4, \Delta_{24} = u_2 + v_4 - c_{24} = -4,$

$\Delta_{31} = u_3 + v_1 - c_{31} = -2, \Delta_{32} = u_3 + v_2 - c_{32} = -2.$

On obtient alors le tableau suivant :

$A_i \backslash B_j$	B_1	B_2	B_3	B_4	a_i	u_i
A_1	1 **(24)**	4 **(4)**	2 **(12)**	3 0	40	0
A_2	5	1 **(42)**	3	4	42	-3
A_4	2	5	1 **(56)**	2 **(14)**	70	-1
b_j	24	46	56	26	152	
v_j	1	4	6	7		

Le critère d'optimalité est vérifié, donc le plan de transport basique x' est optimal, avec $Z(x') = 202$.

3.3.4 Algorithme de Ford et Fulkerson pour la recherche d'un flot maximum :

1. **Principe :**
   L'idée de l'algorithme de Ford et Fulkerson est de faire passer un flot compatible dans le réseau, le plus évident est le flot nul, puis de l'améliorer jusqu'à ce qu'on obtienne un flot compatible.
   Une chaine pour laquelle le flux peut être augmenté est une chaine dont les arcs dans le sens direct n'ont pas atteint leur limite et les arcs dans le sens indirecte ont un flux non nul qui les traverse.
   Autrement dit : Une chaine C est dite augmentant si :
   – Pour tout arc u direct de C, c'est-à-dire $u \in C^+ : f(u) < c(u)$.
   – Pour tout arc u indirect de C, c'est-à-dire $u \in C_- : f(u) > 0$

   Le flot sur cette chaîne C peut être augmenté de la valeur suivante :
   ε = Minimum entre $\{c(u) - f(u)/u \in C^+\}$ et $\{f(u)/u \in C^-\}$.
   Pour améliorer le flot, on ajout ϵ aux flot des arcs de C^+, c'est-à-dire les arcs directs dans la chaîne, et on retranche au flot des arcs de C_-, c'est-à-dire les arcs indirects, dans la chaîne.

2. **Enoncé :**

Données : un graphe valué $G = (X, U, c)$; f un flot maximum.

Résultats : un flot f complet.

(0) Initialisation

Marquer un sommet s et poser : $C^+ = \emptyset; C^- = \emptyset; f^k = 0; A = \{s\}; k := 0$

(1) Soit A l'ensemble des sommets marqués et soit x un sommet de A :

– Marquer le sommet y successeur de x tel que $f(x, y) < c(x, y)$.
On pose : $C^+ := C^+ \cup \{(x, y)\}; A := A \cup \{y\}$.

– Marquer le sommet y prédécesseur de x tel que $f(y, x) > 0$
On pose $C^- := C^- \cup \{(x, y)\}; A := A \cup \{y\}$.

Quand on ne peut plus marquer, deux cas se présentent :

(a) p est marqué aller en (2)

(b) p n'est pas marqué, terminé le flot est maximum.

(2) On a obtenu une chaîne augmentant $C = C^+ \cup C^-$ de s à p.

Pour améliorer le flot on calcule :

$\varepsilon_1 = \min [c(u) - f(u); u \in C^+]$.

$\varepsilon_2 = \min [f(u); u \in C^-]$.

D'où $\varepsilon = \min \{\varepsilon_1, \varepsilon_2\}$

On définit le nouveau flot :

$$f^{k+1}(u) = \begin{cases} f^k(u) + \varepsilon & \text{pour } u \in C^+, \\ f^k(u) - \varepsilon & \text{pour } u \in C^-, \\ f^k(u) & \text{pour } u \notin C \end{cases}$$

Effacer les marques sauf en s et aller en **(1)**.

Soit le réseau de transport de la Figure 3.14 :

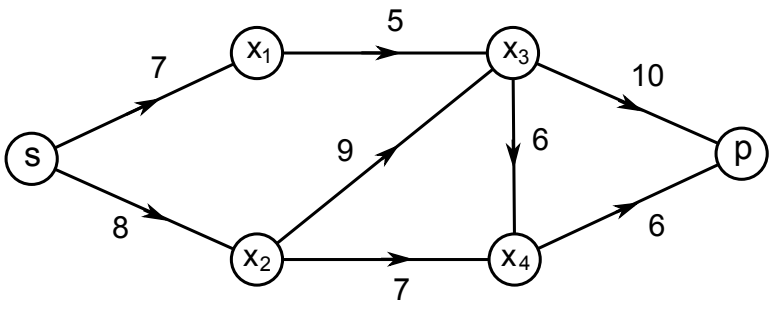


FIGURE 3.14 – réseau de transport.

Cherchons le flot maximum sur le réseau R :

1. **Initialisation :**
   On marque le sommet s (entée du réseau R) par le signe $+$.
   On pose : $A = lbraces\}; C^+ \cup C^- = \emptyset$ et $f^k = 0$; un flot défini sur le réseau $R.k = 0$

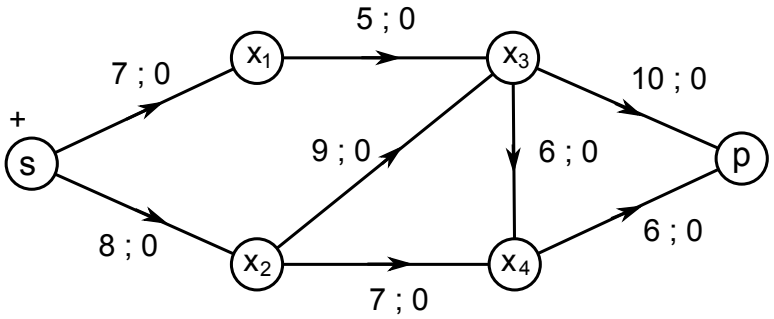


FIGURE 3.15 – Itération 0 (Ford-Fulkerson).

2. **Les itérations :**

- Itération 1 :

Dans le réseau R, on suit la procédure de marquage suivante :

– On marque la sommet x_1 d'un $+$, car il est le successeur de s et $f^0(s,x) = 0 < c(s,x_1) = 7$
  On pose : $C^+ = C^+ \cup \{(s,x_1)\} = \{(s,x_1)\}; A = A \cup \{x_1\} = \{s,x_1\}$.

– On marque la sommet x_3 d'un $+$, car il est le successeur de x_1 et $f^0(x_1,x_2) = 0 < c(x_1,x_3) = 5$
  On pose : $C^+ = C^+ \cup \{(x_1,x_2)\} = \{(s,x_1),(x_1,x_3)\}; A = A \cup \{x_3\} = \{s,x_1,x_3\}$.

– On marque la sommet v d'un $+$, car il est le successeur de x_3 et $f^0(x_3,s) = 0 < c(x_3,s) = 10$
  On pose : $C^+ = C^+ \cup \{(x_3,p)\} = \{(s,x_1),(x_1,x_3),(x_3,p)\}; A = A \cup \{p\} = \{s,x_1,x_3,p\}$.

Le sommet p étant marqué, la procédure s'arrête. On obtient donc la chaine augmentant $C = C^+ \cup C^- = C^+ = \{s,x_1,x_3,p\}$ reliant le sommet s et p.

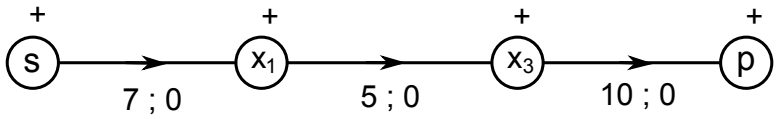


FIGURE 3.16 – Flot f^0.

On calcule : $\varepsilon_1 = min[c(u) - f(u); u \in C^+] = min[c(s,x_1) - f^0(s,x_1); c(x_1,x_3) - f^0(x_1 - x_3); c(x_3,p) - f^0(x_3,p)] = min[7-0; 5-0; 10-0] = 5$

On améliore ainsi le flot f^0 pour obtenir un nouveau flot f^1, en ajoutant la quantité ε_1 au flot des arcs de C^+. Le flux des arcs n'appartenant pas à la chaine, reste inchangé.

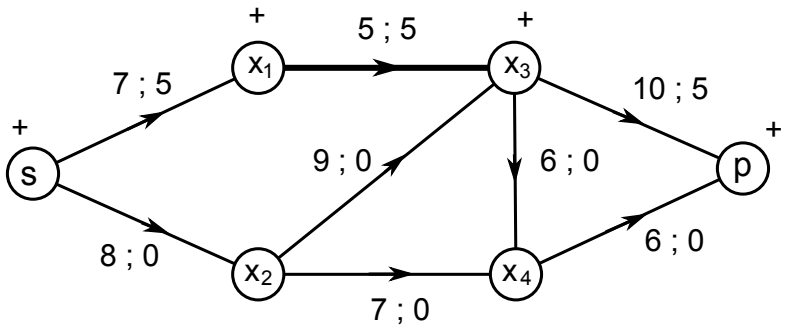


FIGURE 3.17 – Itération 1 (Ford-Fulkerson).

On efface les marques sauf en s.

- Itération 2 :

Dans le réseau R, on suit la procédure de marquage suivante :

– On marque la sommet x_1 d'un $+$, car il est le successeur de s et $f^1(s, x_1) = 5 < c(s, x_1) = 7$
On pose : $C^+ = C^+ \cup \{(s, x_1)\} = \{(s, x_1)\}$; $A = A \cup \{x_1\} = \{s, x_1\}$.
Le sommet x_3 est le successeur de x_1 mais l'arc (x_1, x_3) est saturé $f^1(x_1, x_3) = c(x_1, x_3) = 5$.
Donc, on ne peut pas marquer x_3 (on abandonne ce marquage).

– On marque la sommet x_2 d'un $+$, car il est le successeur de s et $f^1(s, x_2) = 0 < c(x_1, x_3) = 8$
On pose : $C^+ = C^+ \cup \{(s, x_2)\} = \{(s, x_2)\}$; $A = A \cup \{x_2\} = \{s, x_2\}$.

– On marque la sommet x_3 d'un $+$, car il est le successeur de x_2 et $f^1(x_2, x_3) = 0 < c(x_2, x_3) = 9$
On pose : $C^+ = C^+ \cup \{(x_2, x_3)\} = \{(s, x_2), (x_2, x_3)\}$; $A = A \cup \{x_3\} = \{s, x_2, x_3\}$.

– On marque la sommet x_4 d'un $+$, car il est le successeur de x_3 et $f^1(x_3, x_4) = 0 < c(x_2, x_3) = 6$
On pose : $C^+ = C^+ \cup \{(x_3, x_4)\} = \{(s, x_2), (x_2, x_3), (x_3, x_4)\}$; $A = A \cup \{x_3\} = \{s, x_2, x_3, x_4\}$.

– On marque la sommet p d'un $+$, car il est le successeur de x_4 et $f^1(x_4, p) = 0 < c(x_2, x_3) = 6$
On pose : $C^+ = C^+ \cup \{(x_4, p)\} = \{(s, x_2), (x_2, x_3), (x_3, x_4), (x_4, p)\}$; $A = A \cup \{p\} = \{s, x_2, x_3, x_4, p\}$.

Le sommet p est marqué, on arrête le marquage. On obtient donc la chaine augmentant $C = C^+ \cup C^- = C^+ = \{s, x_1, x_3, p\}$ reliant le sommet s et p.

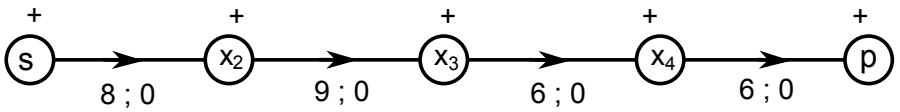


FIGURE 3.18 – Flot f^1

On calcule :
$\varepsilon_1 = min[c(u) - f(u); u \in C^+]$
$= min[c(s,x_2) - f^1(s,x_2); c(x_2,x_3) - f^1(x_2 - x_3); c(x_3,x_4) - f^1(x_3,x_4); c(x_3,x_4) - f^1(x_3,x_4)]$
$= min[8-0; 9-0; 6-0; 6-0] = 6$
On améliore ainsi le flot f^1 pour obtenir un nouveau flot f^2, en ajoutant la quantité ε_2 au flot des arcs de C^+. Le flux des arcs n'appartenant pas à la chaine, reste inchangé.

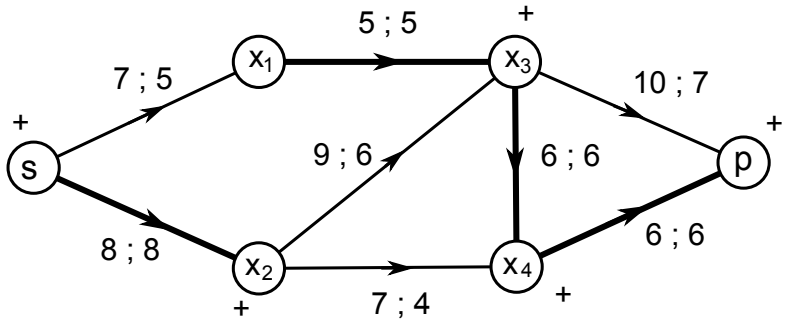


FIGURE 3.19 – Itération 2 (Ford-Fulkerson)

On efface les marques sauf en s.

- Itération 3 :
Dans le réseau R, on suit la procédure de marquage suivante :

- On marque la sommet x_2 d'un $+$, car il est le successeur de s et $f^1(s,x_2) = 6 < c(s,x_2) = 8$
  On pose : $C^+ = C^+ \cup \{(s,x_2)\} = \{(s,x_2)\}; A = A \cup \{x_2\} = \{s,x_2\}$.
- Le sommet x_4 est le successeur de x_2 mais l'arc (x_2,x_4) et $f^2(x_2,x_4) = 0 < c(x_2,x_4) = 7$
  On pose : $C^+ = C^+ \cup \{(x_2,x_4)\} = \{(s,x_2),(x_2,x_4)\}; A = A \cup \{x_4\} = \{s,x_2,x_4\}$.
- On marque la sommet x_3 d'un $+$, car il est le successeur de x_4 et $f^2(x_3,x_4) = 6 > 0$
  On pose : $C^+ = C^+ \cup \{(x_3,x_4)\} = \{(s,x_2),(x_2,x_3),(x_3,x_4)\}; A = A \cup \{x_3\} = \{s,x_2,x_4,x_3\}$.
- On marque la sommet p d'un $+$, car il est le successeur de x_3 et $f^2(x_3,s) = 7 < c(x_3,s) = 10$
  On pose : $C^+ = C^+ \cup \{(x_4,s)\} = \{(s,x_2),(x_2,x_4),(x_3,x_4),(x_3,p)\}; A = A \cup \{p\} = \{s,x_2,x_3,x_4,p\}$.

Le sommet p est marqué, le procédure s'arrête. la chaine obtenu est augmentant $C = C^+ \cup C^- = C^+ = \{s, x_1, x_4, x_3, p\}$ et relie les sommets s et p.

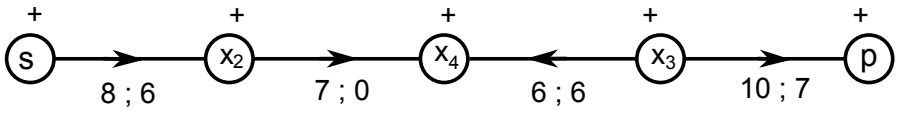


FIGURE 3.20 – Flot f^2

On calcule : $\varepsilon_1 = \min[c(u) - f(u); u \in C^+]$
$= \min[c(s, x_2) - f^2(s, x_2); c(x_2, x_4) - f^2(x_2 - x_4); c(x_3, s) - f^2(x_3, s)]$
$= \min[8 - 6; 7 - 0; 10 - 7] = 2$
$\varepsilon_2 = \min[f(u); u \in C^-]$
$= \min[f^2(x_3, x_4)] = 4$
$\varepsilon = \min[\varepsilon_1; \varepsilon_2] = \min[2, 4] = 2$
On améliore ainsi le flot f^2 pour obtenir un nouveau flot f^3, en ajoutant la quantité ε au flot des arcs de C^+. Le flux des arcs n'appartenant pas à la chaine, reste inchangé.

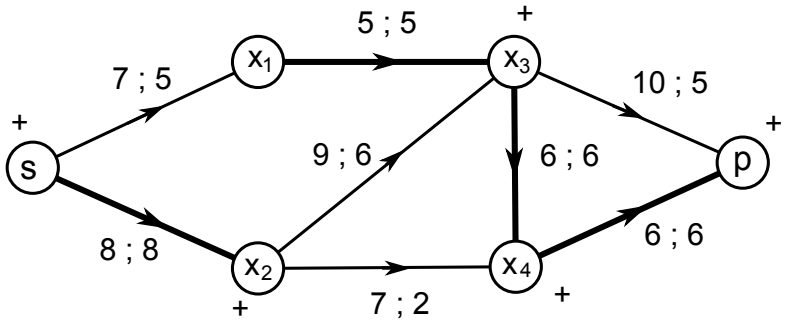


FIGURE 3.21 – Itération 3 (Ford-Fulkerson)

On efface les marques sauf en s

- Itération 4 :

Dans le réseau R, on ne peut pas marquer le sommet p. Donc le flot obtenu est maximum, On le représente comme suit :

Arc	(s, x_1)	(s, x_2)	(x_1, x_3)	(x_2, x_3)	(x_2, x_4)	(x_3, x_4)	(x_3, p)	(x_4, p)
Flux	5	8	5	6	4	4	7	6

La valeur du flot maximum est égale à celle des flux sortant de la source (de s), ou à la somme des flux entrant au puits (à p).

C'est-à-dire : $f_{max} = \sum(f(s,x))/x \in \Gamma^+(s)) = \sum(f(x,p))/x \in \Gamma^-(p))$.

La production maximale qui peut écouler l'usine s vers la ville p est :
$f_{max} = f^3(x_3,p) + f^3(x_4,p) = 7 + 6 = 13$
ou
$f_{max} = f^3(p,x_1) + f^3(p,x_2) = 5 + 8 = 13$

Conclusion :

Dans ce chapitre nous présentons certains algorithmes et méthodes de la théorie des graphes, correspondants aux problèmes posés dans le chapitre précédent en précisant le principe et le procédé de résolution, ainsi des applications appropriées aux algorithmes et méthodes décrits.

Chapitre 4

Résolution de quelques problèmes concrets

Dans ce chapitre nous allons résoudre quelques problèmes concrets, on les modélise sous forme des graphes, puis on utilise les méthodes appropriées vues au chapitre 3, et les programmer sous forme des application Matlab, puis renvoyer et afficher les résultats trouvés.

4.1 Résolution d'un problème d'ordonnancement :

La construction d'une maison, nécessite la réalisation d'un nombre de tâches dont les durées de réalisation, et les contraintes de précédence sont données dans le tableau suivant :

Tâches	Désignation	Durée	Tâches antérieurs
a	Obtenir des briques	10	/
b	Obtenir des toits	24	/
c	Préparer les fondations	14	/
d	La coquille droite	20	a,c
e	La construction du toit	8	d,c
f	Les égouts	14	c
g	installation	20	d
h	Plâtrer	12	i,e,g
i	sanitaires	24	f,d
j	Parqueter	10	i,e,g
k	Aménager le parc	4	n
l	La peinture	12	m,j
m	La menuiserie	4	h
n	L'allée	4	d,f

La tĉhe a " Obtenir des briques ", dure 5 jours, et la tâche d ne peut commencer que si les tâches a et c sont terminées.

On modélise ce problème sous forme d'un réseau Pert comme suit :

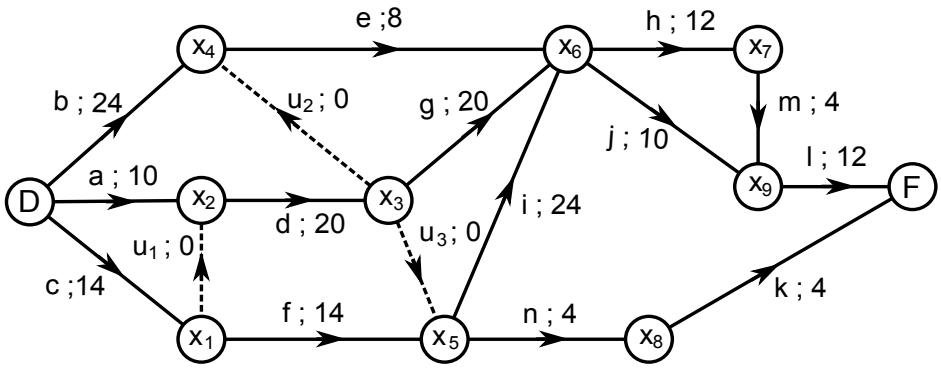


FIGURE 4.1 – Réseau pert du projet de la construction d'un maison.

Les dates de début au plus tôt et les dates de début au plus tard sont déterminées dans le graphe de la Figure 4.2 suivant :

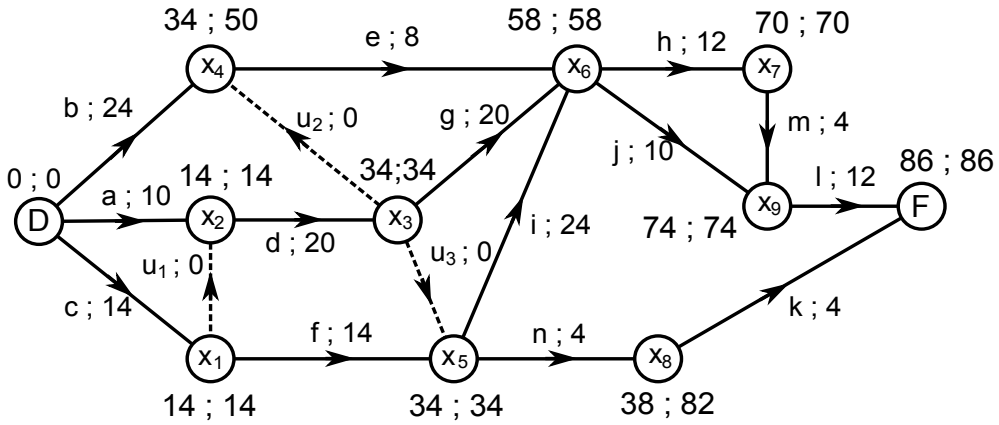


FIGURE 4.2 – Les dates de début au plus tôt et les dates de débuts au plus tard.

Les taches critiques sont les tâches, pour lesquelles la différence : Les dates de début au plus tôt T_i^* - les dates de début au plus tard $T_i = 0$.

- $T_c^* = T_c = 0$
- $T_d^* = T_d = 14$
- $T_i^* = T_i = 34$
- $T_h^* = T_h = 58$
- $T_m^* = T_m = 70$
- $T_l^* = T_l = 74$

On constate que les tâches c,d,i,h,m et l sont critiques, La Figure 4.3 montre le chemin critique associe au réseau du Projet qui est précisé en gras.

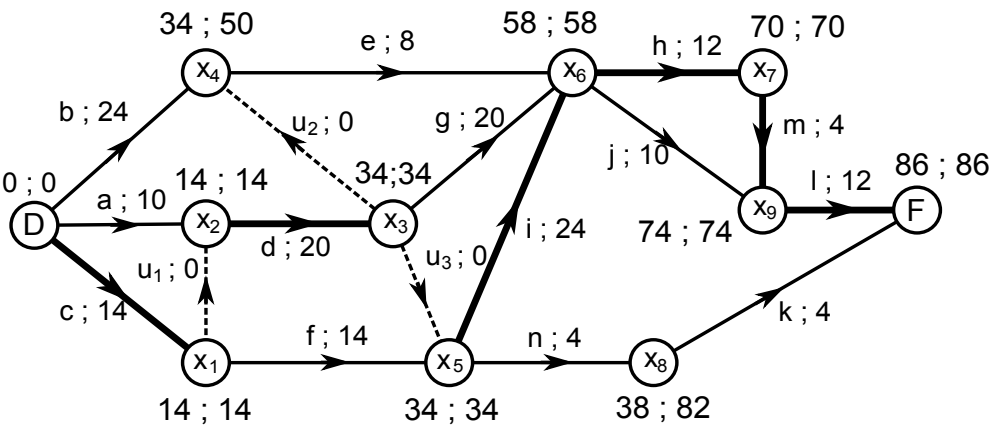


FIGURE 4.3 – Le chemin critique du projet.

Le chemin critique du projet est :

c (Préparer les fondations) $\longrightarrow$ d (La coquille droite) $\longrightarrow$ i (Sanitaires) $\longrightarrow$ h (Plâtrer) $\longrightarrow$ m (La menuiserie)$\longrightarrow$ l (La peinture).

La durée minimale nécessaire à la réalisation du projet est égale à 86 jours.

4.2 Application :

4.2.1 Présentation du logiciel MATLAB :

Partout dans le monde, des millions d'ingénieurs et de scientifiques utilisent Matlab pour analyser et concevoir les systèmes et produits de demain. Matlab est utilisé dans les domaines de l'apprentissage automatique, le traitement du signal, la vision par ordinateur, les communications, la finance computationnelle, la conception de contrôleurs, la robotique et bien plus.

- Mathématiques et Programmation :

La plate-forme Matlab est optimisée pour résoudre les problèmes scientifiques et techniques. Le langage Matlab basé sur les matrices, est la moyen le plus naturelle pour exprimer les mathématiques computationnelles. Les graphiques intégrés permettent de visualiser facilement les données afin d'en dégager des informations. Grâce a la vaste bibliothèque de boites à outils prédéfinis, vous pouvez commencer directement par les algorithmes essentiels à votre domaine. L'environnement bureau encourage l'expérimentation, l'exploitation et la découverte. Les outils et les fonctionnalités Matlab sont tous testés rigoureusement. Ils sont connus pour fonctionner conjointement.

- Principales fonctionnalités :

- Langage de haut niveau pou le calcul scientifique et technique ;
- Environnement bureau pensé pour l'exploitation itérative, la conception et la résolution de problèmes ;
- Graphiques destinés à la visualisation de données et outils connus pour créer des tracés personnalisés ;
- Application dédiées à l'ajustement de courbes, la classification de données, et bien d'autre taches spécialisées ;
- Boite à outils additionnelles connues pour répondre à de nombreux besoins spécifiques aux ingénieurs et aux scientifiques ;
- Outils permettant la création d'applications avec interface utilisateur personnalisée ;
- Interface vers C/C++, Java, .NET, Python, SQL, Hadoop, et Microsoft Excel ;
- Options de déploiement libre de droits permettant de partager des programmes Matlab avec les utilisateurs finaux.

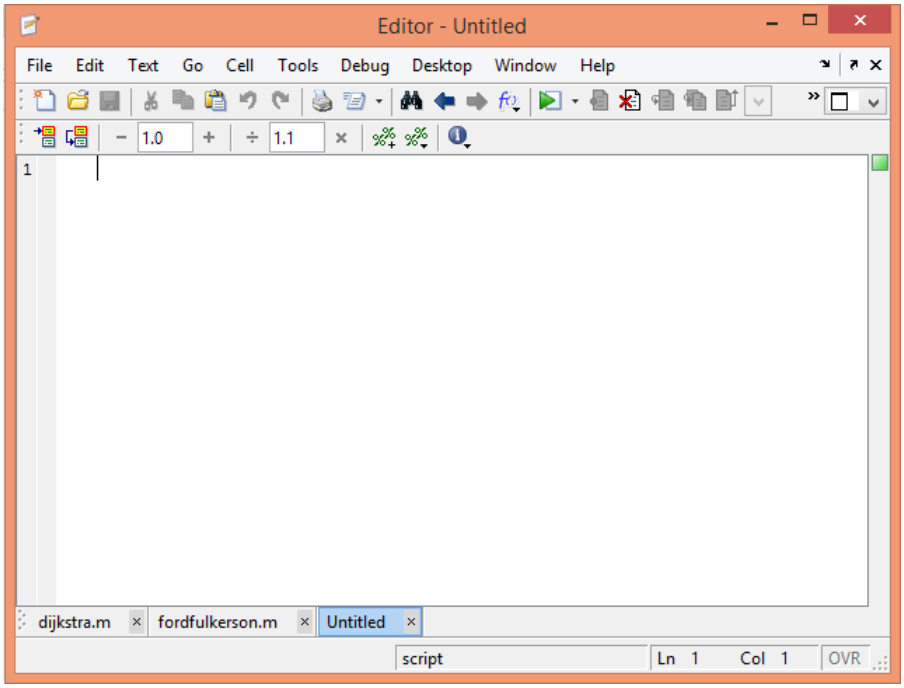


FIGURE 4.4 – Fenêtre Matlab contenant la zone d'écriture et de manipulation des fonctions.

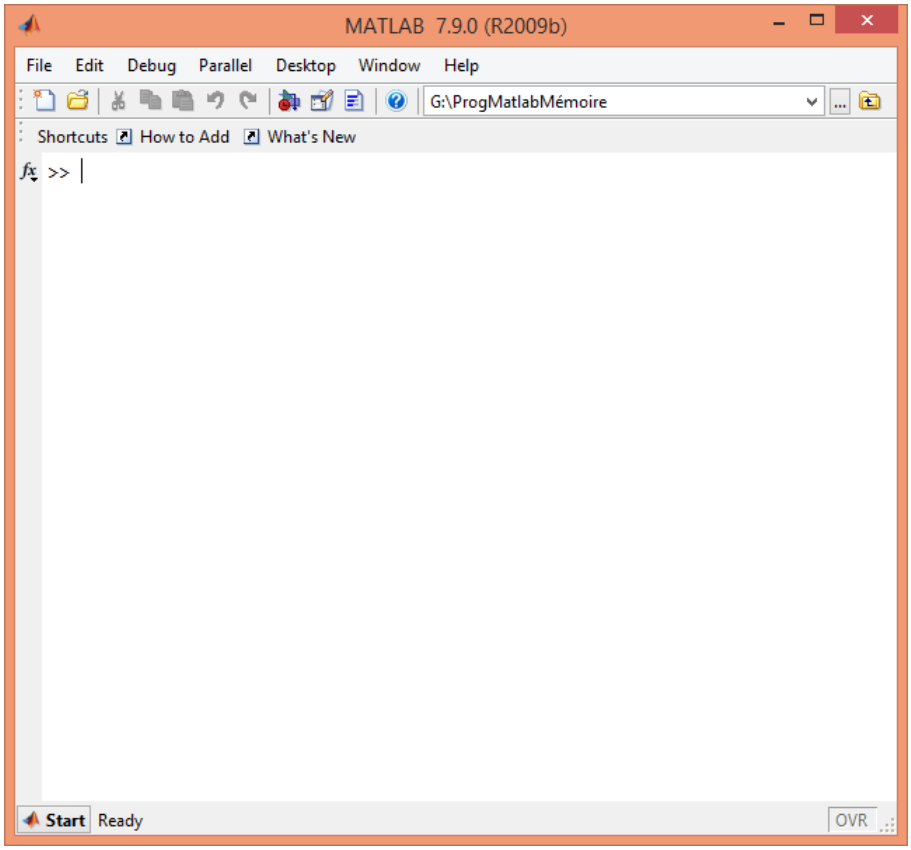


FIGURE 4.5 – Fenêtre Matlab d'entrée-sortie de données.

4.2.2 Résolution de problèmes de plus court chemin :

Plus court chemin dans un réseau routier en Algérie :

Plus court chemin d'une ville a une autre ville :

Un voyageur se trouve dans une ville en Algérie se demande quelle itinéraire doit-il apprendre pour aller à une ville voisine ou lointaine. Ce problème se modélise sous forme d'un graphe non orienté prenant la condition que la route est à deux sens telle que :

- Une ville représente un sommet ;
- Deux ville ayant des frontières en commun, on relié les sommets représentant les deux villes par une arête ;
- La capacité de chaque arête représente la distance en kilomètre entre les deux villes. La Figure 4.6 illustre le graphe modélisant ce problème.

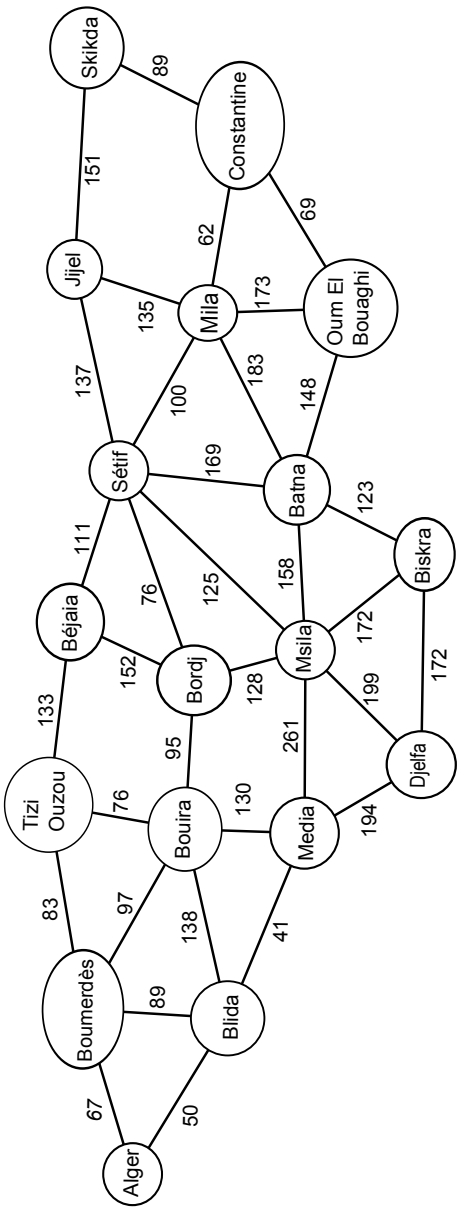


FIGURE 4.6 – Réseau routier en Algérie.

Pour simplifier et illustrer les différentes itérations de recherche de plus court chemin, on utilisera le graphe de la Figure 4.7 en remplaçant les noms des villes par leurs codes administratifs tels qu'ils sont indiqués dans le tableau ci-après.

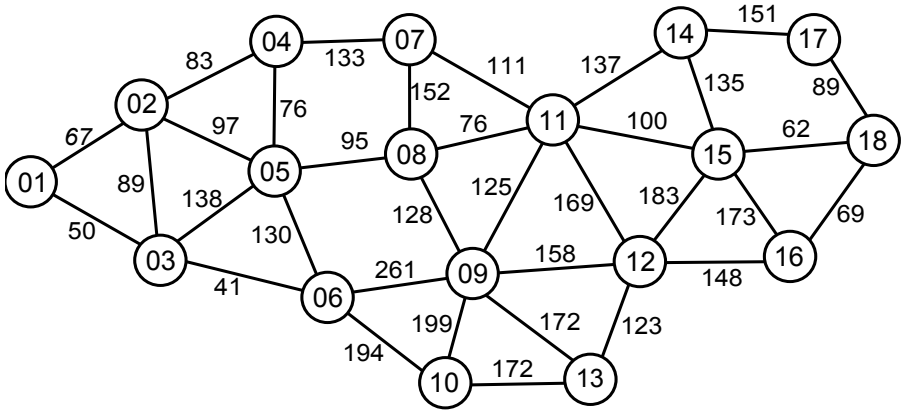


FIGURE 4.7 – Réseau routier simplifier.

Code(Numéro)	Wilaya		Code(Numéro)	Wilaya
11	Sétif		08	Bordj Bouarreridj
07	Béjaia		05	Bouira
04	Tizi-Ouzou		03	Blida
02	Boumerdes		06	Medea
01	Alger		10	Djelfa
14	Jijel		13	Biskra
15	Mila		16	Oum El Bouaghi
12	Batna		18	Constantine
09	Msila		17	Skikda

Un voyageur se trouve à la ville d'Alger cherche le plus court chemin entre les villes d'Alger et Constantine on désigne la ville d'Alger comme ville de départ (01) et la ville de Constantine comme ville d'arrivé (18), l'algorithme de Dijkstra s'arrête lorsque le sommet (18) est atteinte.

Résultats obtenu par application programme matlab Figure 4.8 :

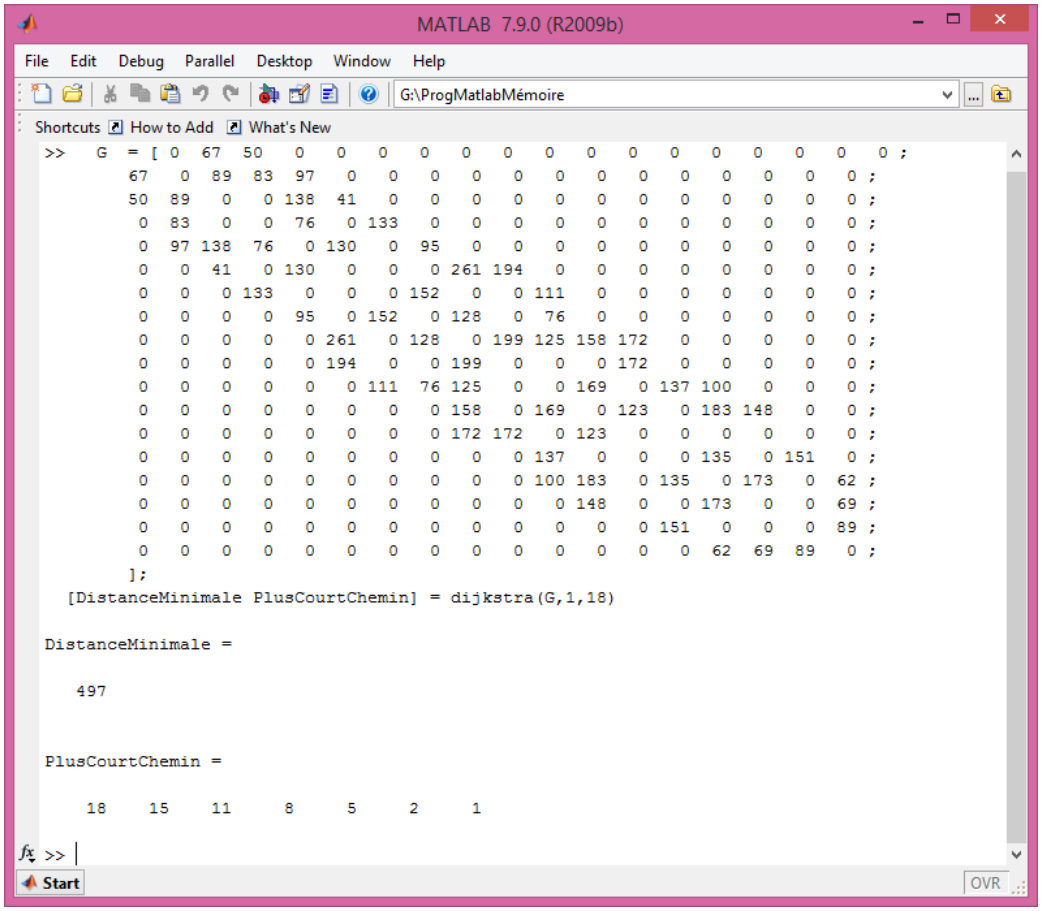


FIGURE 4.8 – Le plus court chemin entre Blida et Skikda obtenu par application matlab.

Le plus court chemin entre la ville d'Alger et la ville de Constantine est (lire " PlusCourtChemin " Figure 4.8 de droite à gauche) :

Alger(01)→ Boumerdes(02)→ Media(05)→ Bordj Bouarerridj(08)
→ Sétif(11)→ Mila(15)→ Canstantine(18).

Avec une Distance de 497 Km.

Si un voyageur se trouve à la ville de Blida cherche le plus court chemin entre les villes de Blida et Skikda on désigne la ville de Blida comme ville de départ (03) et la ville de Constantine comme ville d'arrivé (17), l'algorithme de Dijkstra s'arrête lorsque le sommet (17) est atteinte.

Rsultats obtenu par application programme matlab Figure 4.9 :

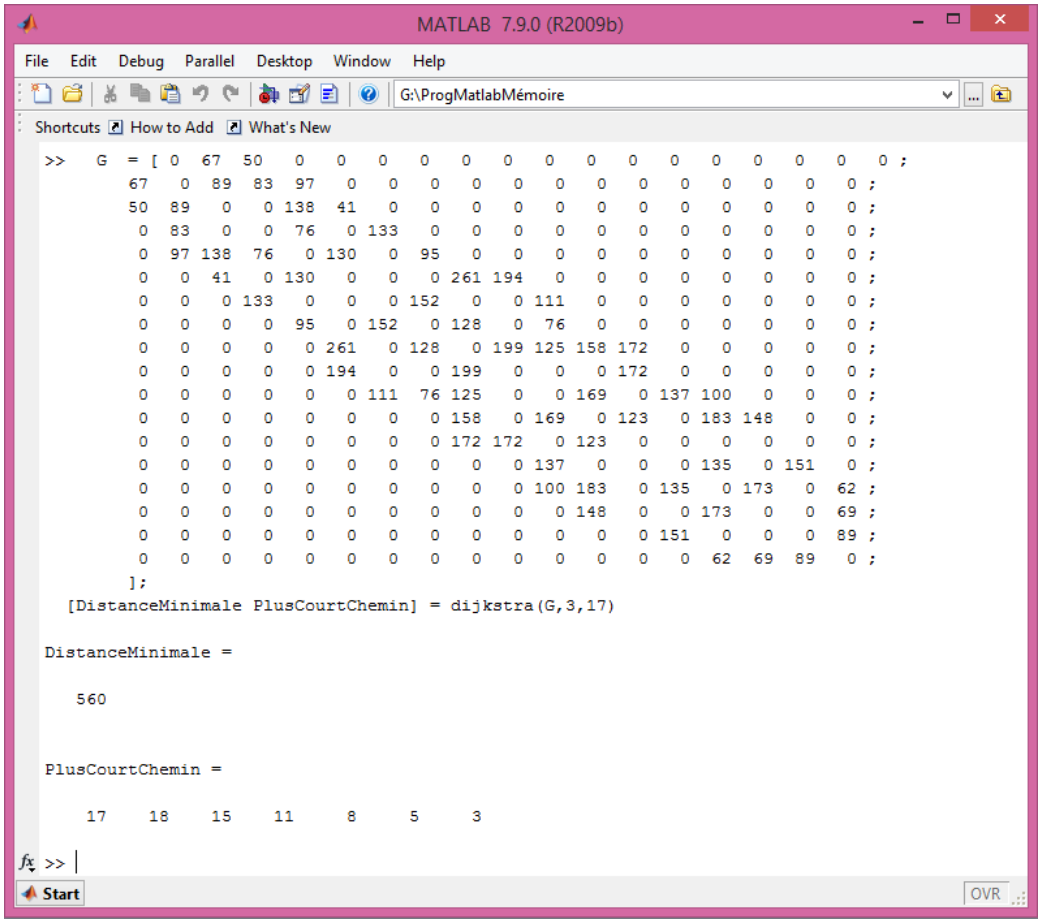


FIGURE 4.9 – Le plus court chemin entre Blida et Skikda obtenu par application matlab.

Le plus court chemin entre la ville de Blida et la ville de Skikda est (lire PlusCourtChemin Fig 4.9 de droite à gauche) :

Alger(03)→ Media(05)→ Bordj Bouarerridj(08)→ Sétif(1)
→ Mila(15)→ Canstantine(18)→ Skikda(17).

Avec une Distance de 560 Km.

4.2.3 Problème de canalisation d'eau dans 3 villes :

Deux châteaux d'eau alimentent 3 villes à travers un réseau de canalisations au sein duquel se trouvent également des stations de pompage. Les châteaux d'eau ont une capacité limitée qui s'élève pour chacun d'eux à 100 000 m^3. Les villes ont exprimées une demande qui est au minimum de 50 000 pour la ville 1, 40 000 pour la 2 et 80 000 pour la ville 3 en m^3. Les canalisations entre les châteaux d'eau et les villes ont des débits limités. Par exemple, pour la canalisation reliant le château 1 à la ville 1, le débit maximum est de 30 alors que celui de la canalisation reliant la station de pompage 1 à la ville 2 est de 50 en milliers de m^3. Ces valeurs figurent sur le graphique de la Figure 4.10 le long des canalisations :

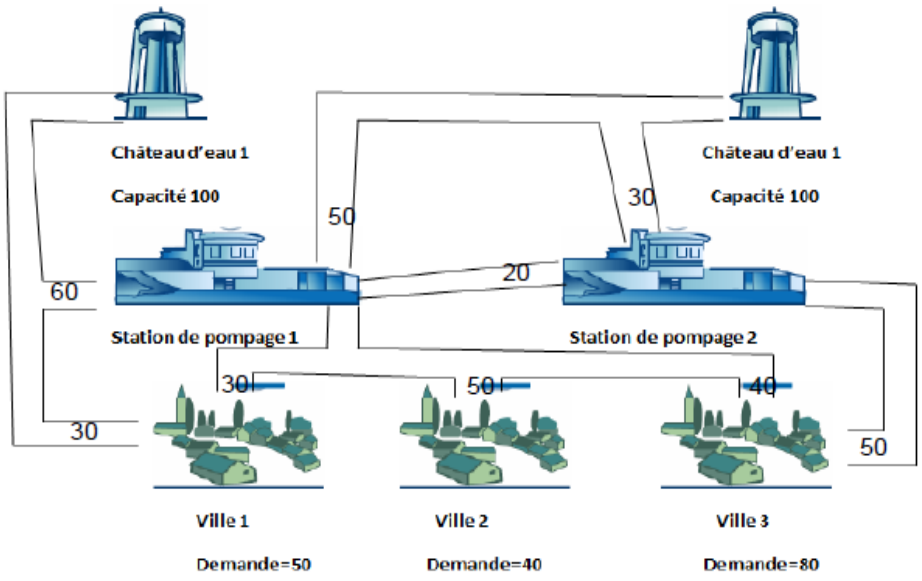


FIGURE 4.10 – Schéma de canalisation d'eau dans 3 villes.

a) Modélisation du problème de distribution d'eau en un problème de flot maximal :

Pour modéliser les capacités des châteaux d'eau, on introduit un sommet supplémentaire s, qui sera la source du réseau, et deux arcs (s, C_1) et (s, C_2) avec une capacité supérieure ou égale à 100. La conservation des flux au sommet C_1 permet de traduire qu'il ne peut pas partir de C_1 une quantité supérieure à 100. Il en est de même pour C_2. Si on veut mesurer ce qui arrive en chaque ville, on introduit un sommet supplémentaire p ; qui sera le puits ; et des arcs de chacune des villes vers p. Pour imposer que les demandes des villes soient satisfaites, on munit ces arcs d'une capacité inférieure ou égale à la demande. Ce qui part de chaque ville sera au moins égal à la demande et, d'après la loi de conservation des flux, ce qui arrive en chaque ville sera aussi au moins égal à la demande.

Le graphe modélisant ce problème est celui de la Figure 4.11 :

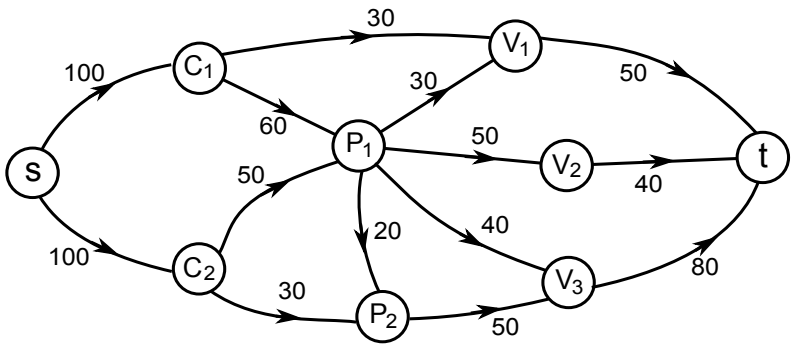


FIGURE 4.11 – Réseau de canalisation d'eau dans 3 villes.

b) Résolution du problème à l'aide de Matlab :

Après l'implémentation de notre algorithme Ford-Fulkerson sur Matlab, on a introduit la matrice correspondant au flux du réseau de canalisation d'eau de la Figure 4.11, après le déroulement du programme on aura les résultats montrées dans la Figure 4.12 :

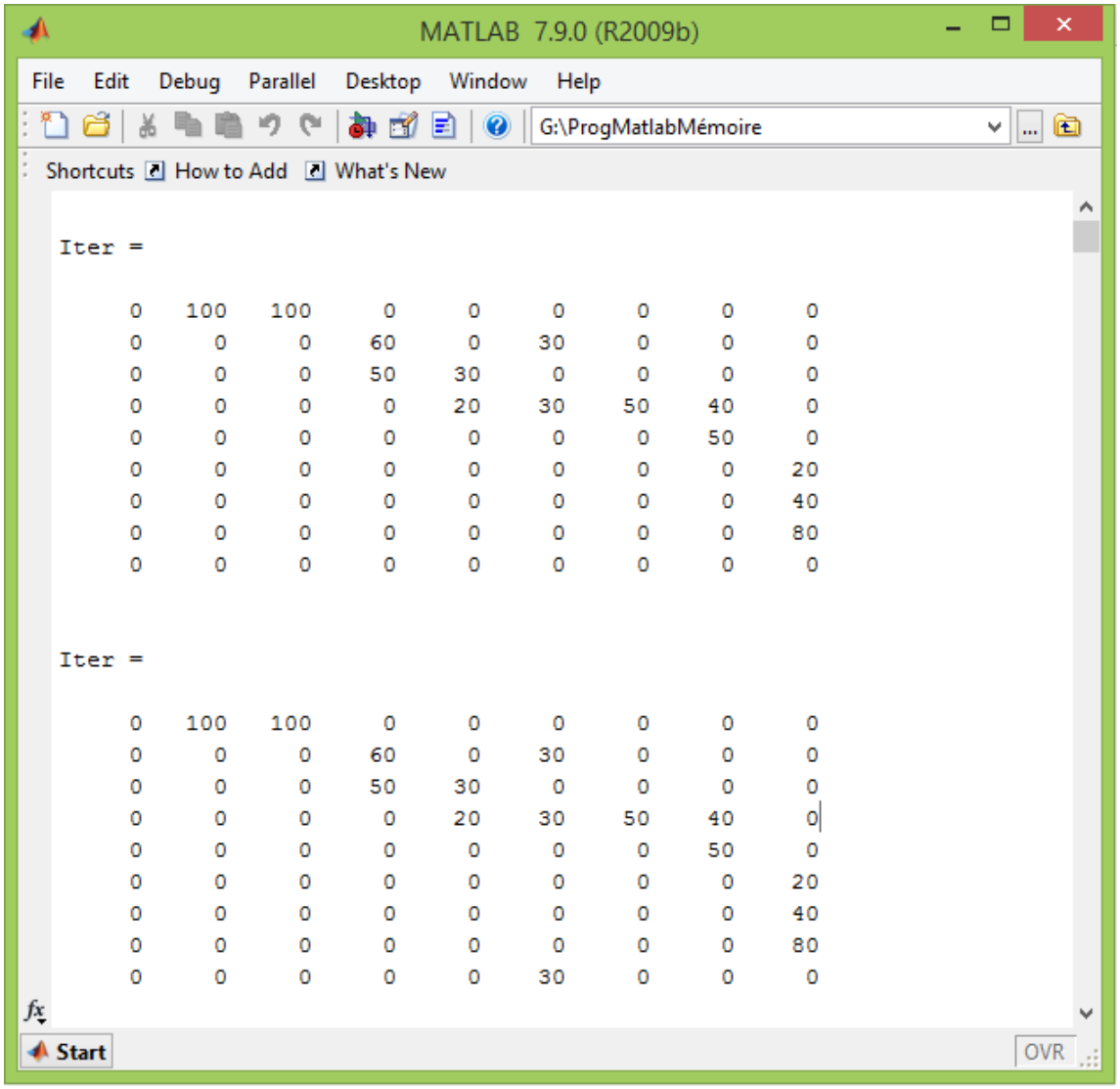


FIGURE 4.12 – Iteration 1 et 2.

Après plusieurs itérations (48 Iter), on obtient le résultat montré sur la Figure 4.13 :

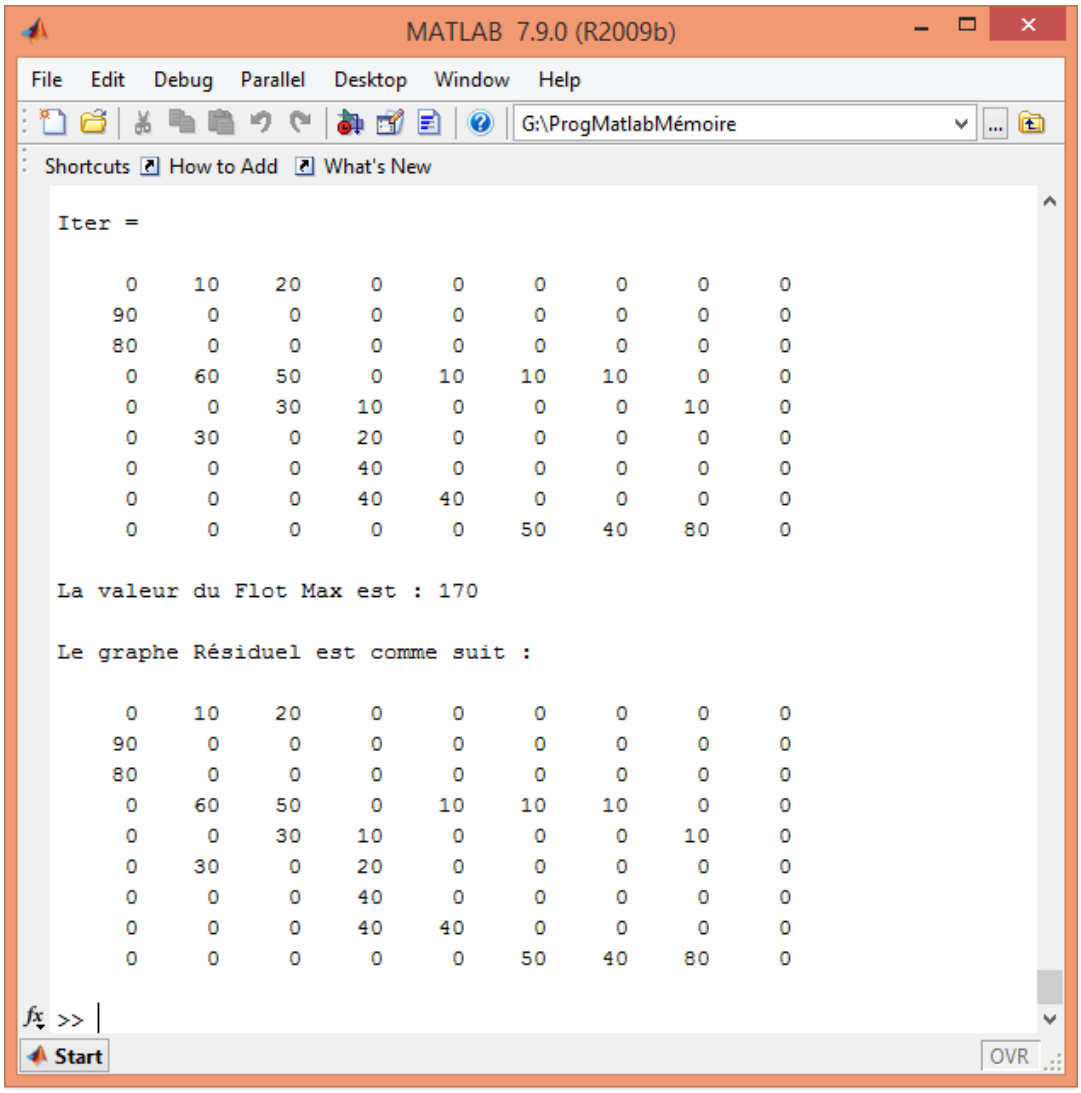


FIGURE 4.13 – Résultat final associé au problème de canalisation d'eau.

Conclusion

Dans ce chapitre, nous intéressons au partie réalisation où nous avons donné les résultats obtenus relatif aux problèmes concrets, modélisés et affichés sous Matlab.

Conclusion générale

Ce travail présente aux lecteurs une vision globale sur la théorie des graphes qui englobe un sujet très intéressant qui est l'optimisation dans les graphes.

L'importance de la théorie des graphes vient du fait qu'elle fournit un cadre conceptuel adéquat pour l'analyse et la résolution de nombreux problèmes.

Elle constitue l'un des instruments les plus courants et les plus efficaces pour résoudre des problèmes discrets posés en Recherche Opérationnel(RO).

Notre travail consiste à donner aux lecteurs un certain nombre d'outils (algorithmes) de la théorie des graphes directement utilisables pour résoudre des problèmes qui peuvent se poser à lui, et l'exactitude de notre objectif consiste à résoudre des problèmes en utilisant l'optimisation en théorie des graphes, pour cela nous nous sommes intéressés à la résolution de quelques problèmes :

Le premier problème traite l'ordonnancement des tâches d'un projet de construction d'un maison afin de trouver la durée minimale de réalisation de ce projet.

Le deuxième problème consiste à traiter un réseau routier en Algérie, où un voyageur se trouve dans une ville en Algérie se demande quelle itinéraire doit-il prendre pour aller à une ville voisin au lointaine ou bien d'une ville à tout les autres villes. Ce problème a été résolu à l'aide de l'algorithme de Dijkstra.

Le troisième problème concerne le réseau de canalisation d'eau en cherchant à déterminer s'il est possible de satisfaire la demande d'eau de trois villes. Ce problème est modélisé sous forme d'un problème de flot maximal dont on a utilisé l'algorithme de Ford-Fulkersen pour sa résolution.

Finalement, notre travail consiste à étudier et implémenter de façon plus approfondie les algorithmes les plus adaptés sous Matlab, afin de trouver la solution la plus adaptée pour l'utilisateur.

Bibliographie

[1] S.Agueniou L.Djerroud A.Rebehi, Problème de cheminement dans les graphes,mémoire Université de Béjaia.(2010/2011).

[2] Khireddine,La k-coloration dans les réseau, mémoire Université de Béjaia.

[3] Fournier. Jean Claude, Paris. Théorie des graphes et applications : Hermès Science publications, 2011.

[4] Franck Butelle. Contribution à l'algorithme distribuée de contrôle : Arbre couvrants avec et sans contranites. Université de Paris, Mars.

[5] Mohamed Aidene et Brahim Oukache. La Programmation Linéaire,Algérie 2005.

[6] Sylvie Barne. Flots dans les réseaux. Université Paris, (2011/2012).

[7] Christine.Solnom. Théorie des graphes et optimisation dans les graphes.

[8] Mlle Nadia Beharrat. Théorie des graphes : Recherche opérationnelle. Algérie (2002-2003).

[9] Tableau kilometrage algerie. Recherche google Site internet 2018.

[10] Pierre Lopez. Cour des graphes. LAAS-CNRS, Novembre.

[11] J.Sirisang K.Nguyens Ouoc N.Nguyens Huu A.Sae Lee C.tram Nepc. Le flot maximal,et la coupe minimale. Institut de la Francophonie pour l'informatique (2012).

[12] Laurent Smoch. Méthodes d'ptimisation. Université de Littoral, Septembre.

[13] Mr S.Taouinat. Cour Théorie des graphes avancée. Université A.mira Béjaia (2016/2017).

[14] Michel Minoux, Michel Gondran,Paris.Théorie des graphes et applications : Hermès Science publications.1979.

[15] F.Khezzari et B.Laidani. Optimisation dans les réseaux. Mémoire Université de Béjaia (2014/2015).

[16] Aimé Sache. La théorie des graphes.Presse université de France 1974.

RÉSUMÉ

Dans ce mémoire, nous nous intéressons à abordé certainement l'un des plus fameux sujets de la théorie des graphes. En particulier, nous montrons quelques méthodes d'optimisation utilisées dans ce cadre afin d'obtenir des meilleurs résultats.

Après avoir montré quel genre de résultat nous pouvions attendre, nous étudions comment adapter les meilleurs méthodes connus à ce jour à savoir l'algorithme de Bellman et Dijkstra, Ford,Ford-Fulkerson...etc, pour la résolution des problèmes concrets, les limites de ces algorithmes sont utilisés dans le cadre de l'optimisation .

Mots clés : Graphe, Optimisation, Chemin, Coloration, Algorithme.

ABSTRACT

In this thesis, we interested to elaborate certainly one of the most famous areas of graph theory. In particular, we show some optimization techniques used in this context in order to obtain better results.

Having shown what kind of results we could expect, we had studied how to adapt the best methods known to date lake the Bellman and Dijkstra, Ford, Ford-Fulkerson ..., for solving concrete problems, the limits of these algorithms used in the context of optimization.

Key words : Graph, Optimisation, Path, Coloration, Algorithme.