République Algérienne Démocratique et Populaire

Ministére d'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane Mira de Béjaia

Faculté des Sciences Exactes

Département de Recherche Opérationnelle

Mémoire de fin de cycle

Pour l'obtention du diplôme de Master en mathématiques appliquées

Option : Modélisation Mathématique et Techniques de Décision

Thème

Quelques méthodes d'optimisation et application dans les graphes

---

Réalisé par :

HAYOUNE Souad

MAOUCHE Noura

Encadré par :

M^r ***K. KABYL***

Devant le jury composé de :

Présidente	M^{elle} **Z. AOUDIA**
Examinateur	M^r **M. AMAD**
Examinateur	M^r **N. NAIT MOUHAND**

2015 – 2016

Remerciements

Nous remercions avant tout, Dieu tout-puissant qui nous a donné la force, le courage et la volonté pour réaliser ce travail.

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

Nous sommes très reconnaissants envers Mr KABYL.K notre 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.

Nous tenons également à remercier les membres de jury qui nous ont fait l'honneur de juger ce travail.

Enfin, Nous remercions de tous cour tous ceux qui ont contribué de prés ou de loin à la réalisation de ce mémoire.

DEDICACES

Nous dédions ce modeste travail :

A *nos chers parents ;*

A *nos frères et nos sœurs ;*

A *toute nos familles ;*

A *nos ami(e)s et collègues et, tous ce qui nous ont aidés ;*

A *toute la famille RO.*

A *toute la promotion 2015-2016 ;*

TABLE DES MATIÈRES

LISTE 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

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 des problèmes de la théorie des graphes et les méthodes de la résolution.

Puis le chapitre trois est consacré à la partie réalisation dans laquelle nous avons présentée certaines situation réelles modélisées sous forme graphique, dont les solutions sont optimales.

Dans le dernier chapitre, nous abordons l'application de cinq problèmes concrets qui résument le but de notre mémoire à savoir les techniques d'optimisation dans les graphes programmés sous langage C. Nous présentons le premier c'est le réseau routier en Algérie, puis le remplacement des véhicules dans une location de voiture, aprés le problème de canalisation d'eau dans trois villes, ensuite nous entamons l'ordonnencement des tâches d'un projet. En finira par le problème de stockage et transport des produits chimiques.

En guise ce travail par une conclusion générale.d'étude.

CHAPITRE 1

NOTATIONS ET NOTIONS DE BASES

La théorie des graphes est une théorie mathématique relative aux ensembles des nœuds (sommets) et de liens. Le terme « graphe » désigne la représentation visuelle d'un tel ensemble. La théorie des graphes a été utilisés dans de nombreux domaines d'études pour modéliser des phénomènes de la réalité.

1.1 Concept des graphes

Définition 1. *Un graphe $G = (X, E)$ est constitué d'un ensemble fini $X = \{x_1, x_2, ..., x_n\}$ appelés sommets avec $\mid X \mid = n$ et d'une famille $E = \{e_1, e_2, ..., e_m\}$ de paires destincts de X appelés arêtes (arcs) avec $\mid E \mid = m$.*
Le nombre de sommets du graphe G est appelé ordre de G qu'on note O(G)[9].

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 [9].

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 $I(e_1) = 2$, $T(e_1) = 1$.

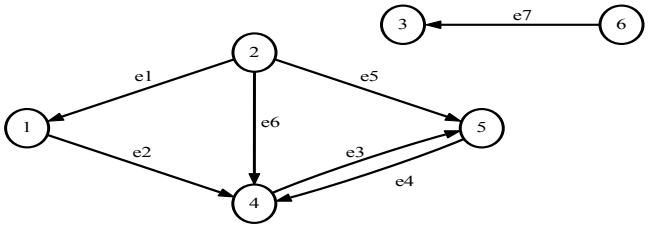


FIGURE 1.1 – Graphe orienté

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 à 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 [9].

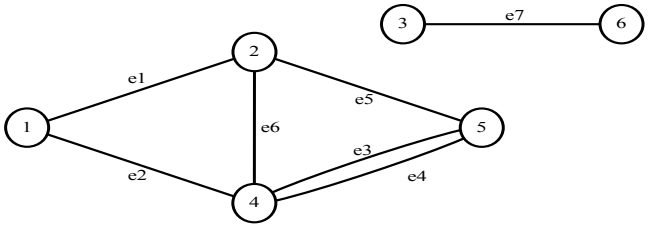


FIGURE 1.2 – Graphe non orienté

.

Succésseurs, prédecesseurs d'un sommet

- a) L'ensemble des succésseurs 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 prédecesseurs d'un sommet x_i est l'ensemble de tous les arcs entrants au sommet x_i noté par $\Gamma^-(x_i)$ tel que :
$$\Gamma^-(x_i) = \{x_j \in X/(x_j, x_i \in E)\},$$
- c) x est un voisin de y si x est un prédecesseur ou x est un succésseur qu'on note $\Gamma(x)$ avec :$\Gamma(x) = \Gamma^-(x) \cup \Gamma^+(x)$ [9].
Dans le graphe orienté de la Figure 1.3 on a : $\Gamma^-(4) = \{1, 2\}$ et $\Gamma^-(4) = \{3\}$.

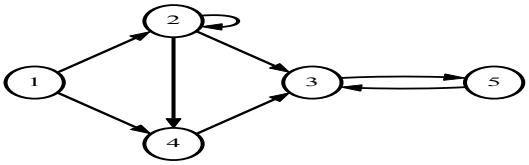


FIGURE 1.3 – Succésseurs, prédecesseurs d'un sommet

Degré d'un sommet

Soit $G = (X, E)$ un graphe orienté :

- a) *Le demi degré éxtérieur 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 : $d_G^+(x) = |\{e \in E/I(e) = x\}|$.
- b) *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 : $d_G^-(x) = |\{e \in E/T(e) = x\}|$.
- c) *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 : $d_G(x) = d_G^-(x) + d_G^+(x)$ [9].

propriété 1. *Pour un graphe non orienté G=(X, E) [9]*

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

propriété 2. *Pour un graphe orienté G=(X, E) [9]*

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

Digraphe

Est un graphe qui ne possède pas ni de boucle ni d'arcs (arêtes) multiples, ce graphe est appelé aussi graphe simple. Le graphe de la Figure 1.4 représente un digraphe de quatre sommets et cinq arêtes [4].

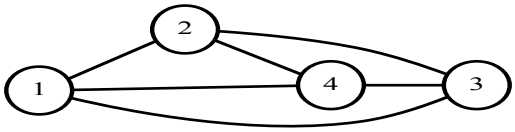


FIGURE 1.4 – Digraphe

Graphe complémentaire

Soit $G = (X, E)$ un graphe simple. Le graphe $\bar{G} = (X, \bar{E})$ est le graphe complémentaire de G [9] si :

$$\forall e \in E \Longleftrightarrow e \notin \bar{E} \ .$$

La Figure 1.5 représente le graphe G ainsi que son graphe complémentaire $\overline{G}$.

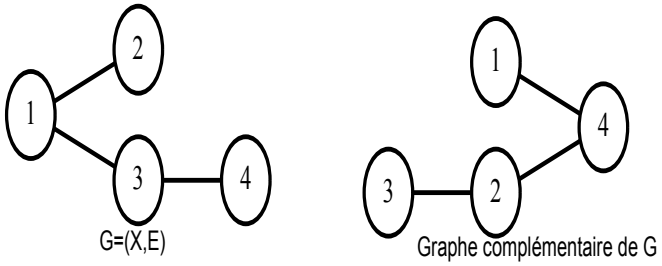


FIGURE 1.5 – Graphe G et son complémentaire

Graphe multiple

C'est un graphe qui possède des boucles ou bien des arêtes (arcs) multiples [9].

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 [12].

Le graphe de la Figure 1.6 représente un graphe G et son graphe partiel [12].

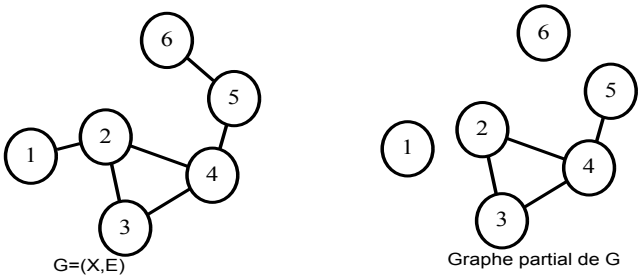


FIGURE 1.6 – Graphe G et l'un de ces graphes partiels

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 [12].
Soit G=(X, E) le graphe de la Figure 1.7, on prend $A = \{1, 2, 4\}$ et
$E(A) = \{e_1, e_2, e_6\}$ le sous graphe de G.

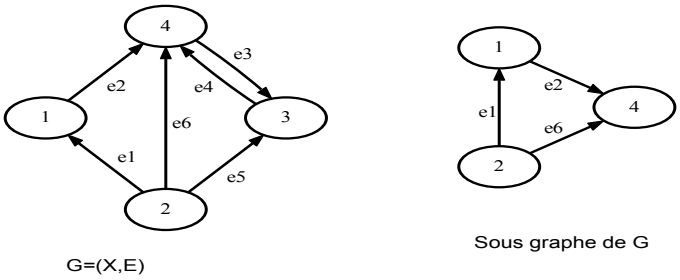


FIGURE 1.7 – Graphe G

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

Une clique

Soit $G = (X, E)$ un graphe simple tel que X un ensemble de sommets de G deux à 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 1.8 représente deux cliques de taille 3 et 5 réspectivement [1].

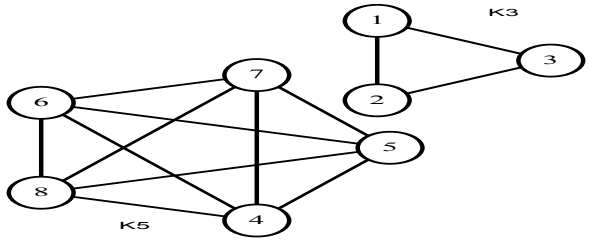


FIGURE 1.8 – Cliques

Un stable

Un stable est un ensemble de sommets de G deux à deux non-adjacents[1].
Le graphe de la Figure 1.9 admit deux stables tel que : $S_1 = \{1, 3, 6, 8\}$ et $S_2 = \{2, 4, 5, 7\}$.

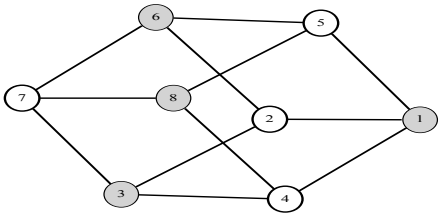


FIGURE 1.9 – Stable

1.2 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 $(x_0, x_1, ..., x_k)$.

- 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 *chaine simple* et celle qui passe une fois par chaque sommet est dite *chaine élémentaire.*

b) Chemin : Un chemin de x_0 à x_k dans un graphe est une suite de sommets relie succéssivement par des arcs orietés dans le même sens, on le note $(x_0, x_1, ..., x_k)$.

- 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 à un autre sommet x_j, la longeur du plus court chemin de x_i à x_j [3] ;
- Le diamètre de graphe G est alors la plus grand distance entre deux sommet de G.

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

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

1.3 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 G1 de la Figure 1.10 est composé d'une seule composante connexe, tandis que le graphe G2 est composé de 2 composantes connexes.

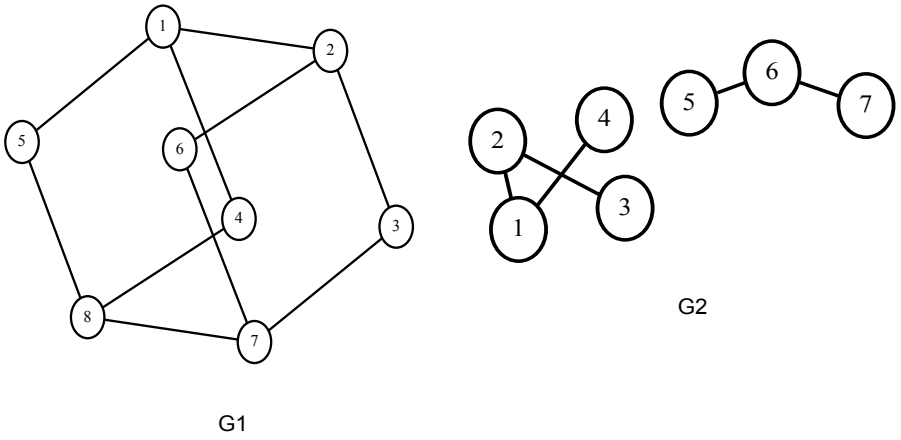


FIGURE 1.10 – Composante connexe

Définition 2. *Un graphe G est dit* ***fortement connexe*** *si :*$\forall x, y \in X^2$*, il existe un chemin de x à y et un autre chemin de y à x [12].*

Définition 3. *Un graphe G est dit fortement connexe s'il possède une seule composante fortement connexe [12].*

Le graphe de la Figure 1.11 contient 2 composantes fortements 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}.

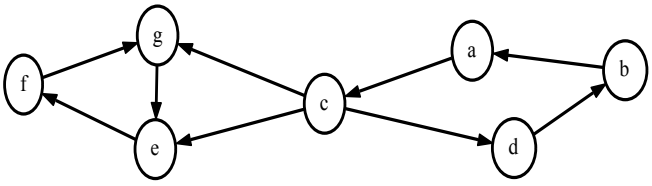


FIGURE 1.11 – Composantes fortements connexes

On appelle le graphe reduit 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. [12]
Le graphe de la Figure 1.12 représente le graphe reduit de la Figure 1.11.

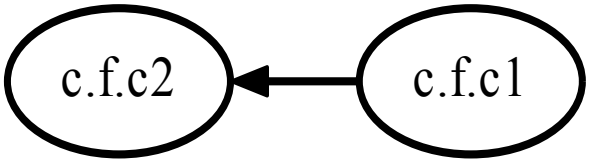


FIGURE 1.12 – Graphe reduit

1.3.1 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 marquier tous ses voisins d'un singne (+) (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 ;
4. Retirer de W les sommets C_k et poser $W = W - C_k$;
5. Tester si $W = \emptyset$
   * si oui terminer ;
   * sinon aller en (1).

Le nombre de composantes connexes est k et C_k sont les composantes connexes de G [12].

On applique l'algorithme précédent sur le graphe de la Figure 1.13 :

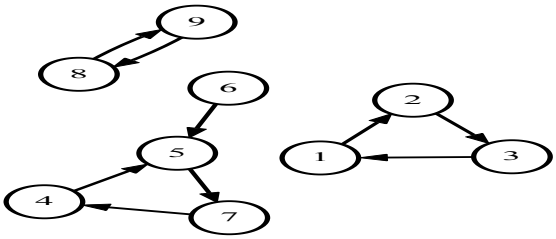


FIGURE 1.13 – Recherche des composantes connexes

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

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

1.3.2 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 succésseurs direct et indirect d'un singne (+) ;
3. Marquer tous ses prédécesseurs direct et indirect d'un singne (-) ;
4. Poser $k = k + 1$,C_k l'ensemble des sommets marqués d'un singne (+) et (-) ;
5. Poser $W = W - C_k$, et éffacer toutes les marques ;
6. Tester si $W = \emptyset$
   * si oui terminer ;
   * si non aller en (1).

Le nombre de composantes fortement connexes est k et C_k sont les composantes fortement connexes de G [12].
On applique l'algorithme ci-déssus sur le graphe de la Figure 1.14, on obtient :

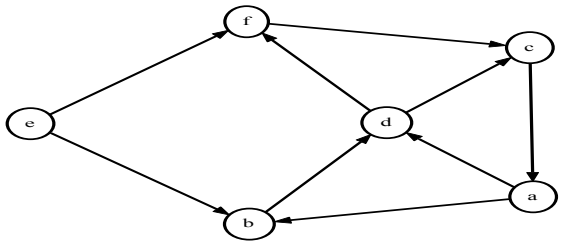


FIGURE 1.14 – Recherche des composantes fortement connexes

- $k = 0$, $W = \{a, b, c, d, e, f\}$
- $k = k + 1 = 1$,$C_1 = \{a, b, c, d, f\}$, $W = W - C_1 = \{e\}$,$W \neq \emptyset$,aller en (1) ;
- $k = k + 1 = 2$, $C_2 = \{e\}$, $W = W - C_2 = \emptyset$, terminer.
- $k = k + 1 = 3$, $C_3 = \{8, 9\}$, $W = W - C_3 = \emptyset$,

D'ou le nombre de composantes fortement connexes est 2 et les composantes connexes sont : $C_1 = \{a, b, c, d, f\}$,$C_2 = \{e\}$.

1.3.3 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 $\tau^- G(x) = \emptyset$;
3. Poser N_0 (niveau nul) l'ensomble 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 couvrer tous les sommets du G ;
6. Présenter par la suite le graphe ordonné.

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	$\emptyset$

$N_4 = \{e\}$

Pour le graphe ordonné de graphe G est donné par la Figure 1.15 :

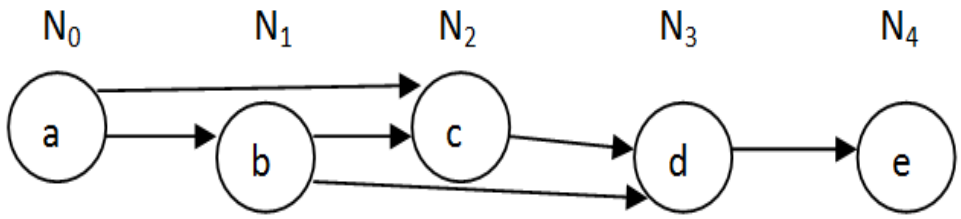
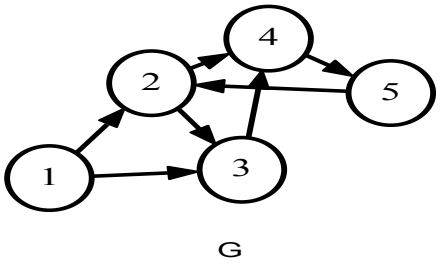


FIGURE 1.15 – Graphe ordonné

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 1.16. On obtient alors le dictionnaire des prédécesseurs donné.



G

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 1.16 – Mise en ordre d'un graphe

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

Comment determiner le circuit :

Le sommet 5 nous renvoit à sa ligne dans la colonne x, on choisit un sommet dans la ligne 5 soit 4, le sommet 4 nous renvoit à sa ligne dans la colonne x, on choisit un sommet dans la ligne 4, soit 2 , le sommet 2 nous renvoit 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 lécture est de gauche à droite) $\{5, 4, 2, 5\}$ est le circuit de graphe qui donné par la Figure 1.17

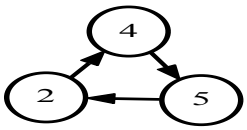


FIGURE 1.17 –

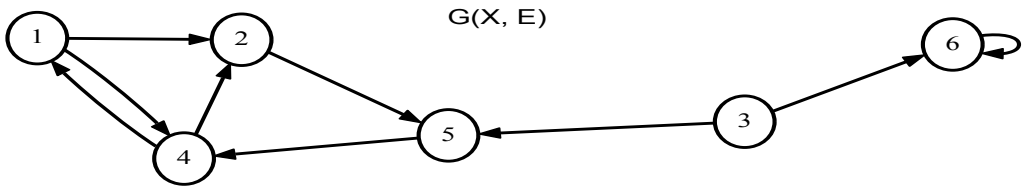
1.4 Representation matricielle d'un graphe

1.4.1 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 :

$$a_{ij} = \begin{cases} 1 & \text{s'il éxiste un arc } (x_i, x_j) \text{ ou } x_i = I(e) \text{ et } x_j = T(e) \\ 0 & \text{sinon} \end{cases} \tag{1.1}$$

La Figure 1.18 représente Un graphe G et sa matrice d'adjacence .



	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 1.18 – Graphe G et Sa matrice d'adjacence

1.4.2 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 [8]

$$a_{ij} = \begin{cases} 1 & \text{si } x_i \text{ est l'extrémité initial de } e_j \\ -1 & x_i \text{ est l'extrémité terminal de } e_j \\ 0 & \text{sinon} \end{cases} \tag{1.2}$$

La Figure 1.19 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

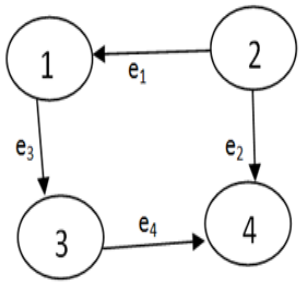


FIGURE 1.19 – Matrice d'incidence

Remarque :

$d^+G(x)$= 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.

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 1.20 représente un couplage maximum et parfait.

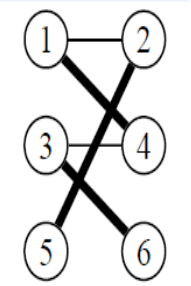


FIGURE 1.20 – Couplage

1.5 Coloration d'un graphe

Le coloriage des sommets (ou des arêtes) d'un graphe consiste à affecter une couleur à chaque sommet (resp. arête) tel que deux sommets (resp. arêtes) adjacentes ne soient pas coloriées de la même manière (coleur).
Le nombre minimum de couleurs nécessaires pour colorier un graphe G est appelé le nombre chromatique de G, et noté $\gamma(G)$.[1]

1.5.1 Le nombre chromatique dans un graphe

- **Coloration simple**
  Soit $G = (X, E)$ un graphe simple, une coloration simple est une coloration des sommets, tel qu'il existe un nombre de sommets coloriés par une couleur donnée qui soit différent au moins à un seul nombre de sommets coloriés par une autre couleur.

Le graphe de la Figure 1.21 représente une coloration simple des sommets du graphe G ci-aprés.

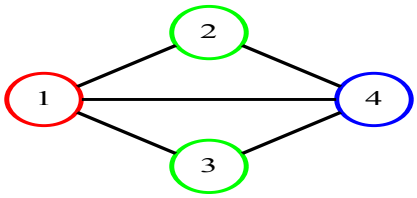


FIGURE 1.21 – Coloration simple

Soit $G = (V, E)$ un graphe donné, une coloration K-équitable est une coloration des sommets du graphe G vérifiant :
– Le graphe G est k-coloriable.
– $\forall (i, j) \in 1, ..., K$, le nombre de sommets coloriés par la couleur i est égale au nombre de sommets coloriés par la couleur j.
Le graphe de la Figure 1.22 représente une coloration K-équitable des sommets.

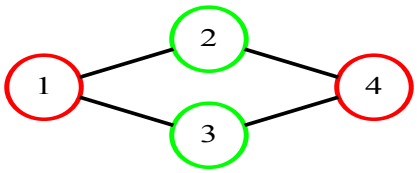


FIGURE 1.22 – Coloration K-équitable

1.5.2 Quelques opérations sur les graphes :

1. **Subdivision de graphe :**
   En remplaçant les arêtes d'un graphe $G = (X, E)$ par des chânes disjointes intérieurement, on obtient une subdivision de G.[5] Une subdivision d'une arête $e \to v$ s'obtient en ajoutant au graphe de départ de nouveaux sommets $(x_1, x_2, ..., x_n)$ et en remplaçant l'arête $e \to v$ par un chemin $(e \to x_1 \to ... \to x_n \to v)$ (la subdivision triviale obtenue pour $n = 0$ consiste à ne rien changer). Un graphe $G^{'}$ est une subdivision d'un graphe G si on l'obtient par subdivision distinctes de ses arêtes. Comme cas de subdivision nous avons le graphe de la Figure 1.23

FIGURE 1.23 – Subdivision de graphe

2. **Somme cartésienne de deux graphes :**

   Soient $G = (X, E)$ et $H(Y, F)$ deux graphes ; on définit le graphe
   $G \bigoplus H = (S, T)$ appelé somme cartésienne de G et H, tel que $S = X \times Y$ et $T = \{e = (x, y)(x^{'}, y^{'})/(x, y) \sim (x^{'}, y^{'})$ ssi $x = x^{'}$et $y \sim y^{'}$ ou $y = y^{'}$ et $x \sim x^{'}\}$.

La Figure 1.24 montre la somme cartésienne de deux graphes G1 et G2.

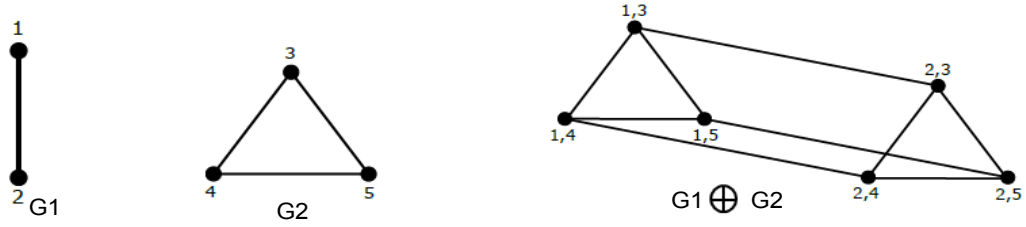


FIGURE 1.24 – Somme cartésienne

3. **Produit cartésien de deux graphes :**
   Soient $G = (X, E)$ et $H(Y, F)$ deux graphes ; on définit le graphe $G \bigotimes H = (S, T)$ appelé produit cartésien de G et H, tel que $S = X \times Y$ et $T = \{e = (x, y)(x^{'}, y^{'})/(x, y) \sim (x^{'}, y^{'})$ ssi $x \sim x^{'}$et $y \sim y^{'}\}$. La Figure 1.25 montre le produite cartésien de deux graphes M_1 et M_2

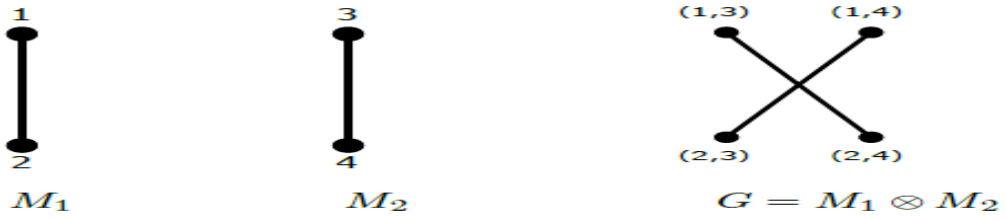


FIGURE 1.25 – Produit cartésien

4. **Morphismes de graphes :**
   Soit $G = (X, E)$ et $H = (T, B)$ deux graphes non-orientés. On dit qu'une application ϕ de X dans T réalise un morphisme du graphe G vers le graphe H si elle vérifie la propriété suivante :

$$uv \in E \Longrightarrow \phi(u)\phi(v) \in B$$

   Si ϕ est une application injective, on dit que le graphe G s'injecte dans le graphe H. On dit aussi parfois que le graphe H contient le graphe G [2].
   On dit que ϕ est un isomorphisme si il vérifie les deux conditions suivantes :

– ϕ réalise une bijection entre A et B.
– $uv \in E \iff \phi(u)\phi(v) \in B$

La Figure 1.26 représente un isomorphisme d'un graphe.

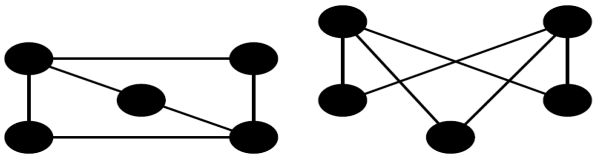


FIGURE 1.26 – Isomorphes

1.6 Quleques graphes particuliers

1.6.1 Graphe reflexif

On appelle graphe reflexif, un graphe possèdant une boucle sur chaque sommet. Le graphe de la Figure 1.27 est reflexif [9].

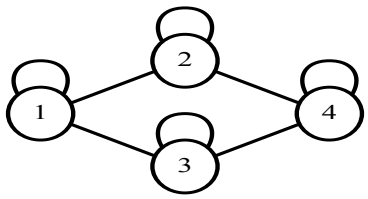


FIGURE 1.27 – Graphe reflexif

1.6.2 Graphe symétrique

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

La Figure 1.28 montre un graphe symétrique [9].

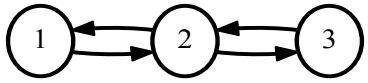


FIGURE 1.28 – Graphe symétrique

1.6.3 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 1.29 est un graphe antisymétrique [9].

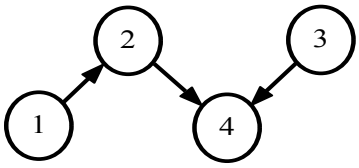


FIGURE 1.29 – Graphe antisymétrique

1.6.4 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 $e_k = (x, z)$ [9].

Le graphe de la Figure 1.30 est un graphe transitif.

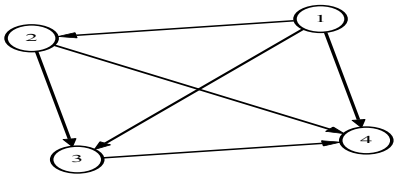


FIGURE 1.30 – Graphe transitif

1.6.5 Graphe complet

Un graphe est dit complet si chaque sommet est relie avec tous les autres sommets. À titre exemple, le graphe de la Figure 1.31 est un graphe comlpet [9].

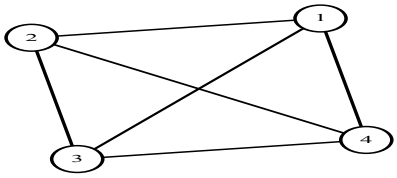


FIGURE 1.31 – Graphe complet

1.6.6 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 [9].

La Figure 1.32 représente un graphe biparti.

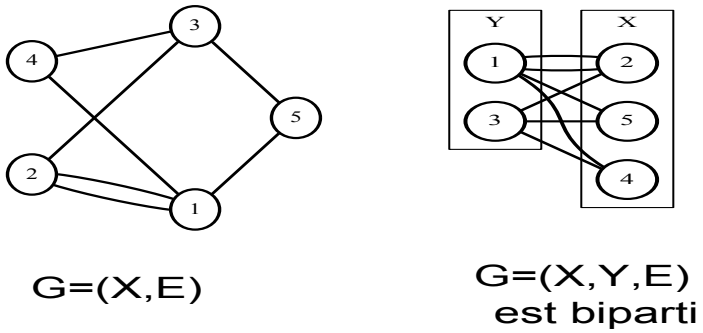


FIGURE 1.32 – Graphe biparti

1.6.7 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 $\mid X \mid = p$, $\mid Y \mid = q$ [9]. La Figure 1.33 représente un graphe biparti-complet..

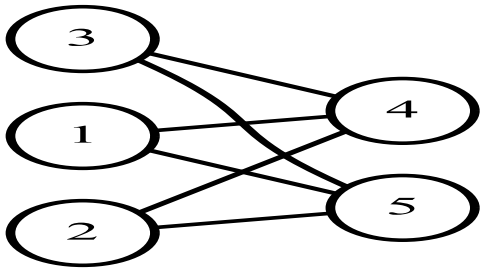


FIGURE 1.33 – Graphe biparti complet

1.6.8 Graphe biparti équilibré

Soit G un graphe biparti. Si $\mid X \mid = p$ et $\mid Y \mid = p$ alors, on dit que G biparti équilibré [9].

1.6.9 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 [9].
La Figure 1.34 est une représentation d'un graphe planaire.

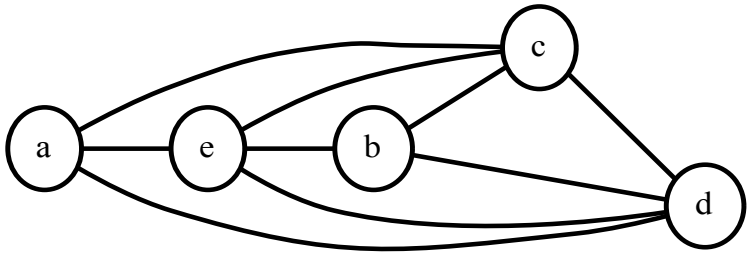


FIGURE 1.34 – Graphe planaire

1.6.10 Hypercube

L'hypercube de dimension n, noté Q_n est le graphe dont l'ensemble des sommets est formé des n-uplets binaires, et deux sommets sont adjacents si et seulement s'ils diffèrent par exactement une seule composante (coordonnée).
Notons que $Q_0 = K_1$, $Q_1 = K_2$ et d'une manière générale, Q_n peut être défini récursivement en utilisant la somme cartésienne de Q_{n-1} avec K_2.
Il est clair que pour tout $n \geq 2$, Q_n est isomorphe à $\underbrace{K_2 \square K_2 \square ... \square K_2}_{n fois}$.

La Figure 1.35 montre les premiers hypercubes.

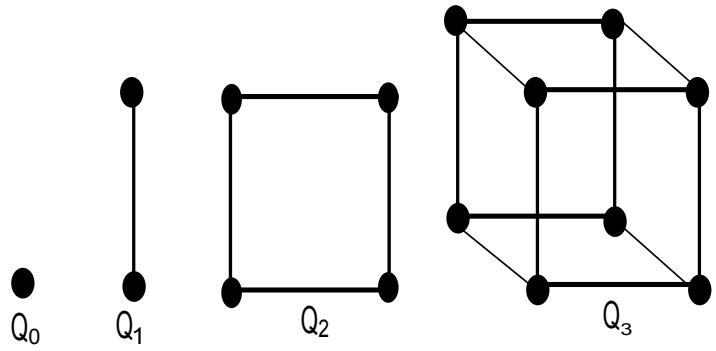


FIGURE 1.35 – Hypercube

%center

FIGURE 1.36 – Q_0, Q_1, Q_2 et Q_3

1.6.11 Arbre

Un arbre est un graphe connexe sans cycle ayant les proprétés 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 [9].

La Figure 1.37 constitue un arbre.

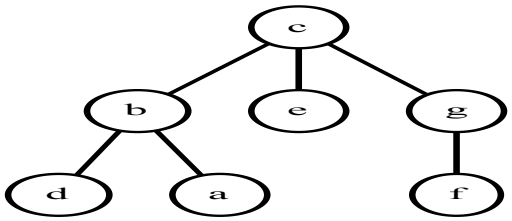


FIGURE 1.37 – Arbre

1.6.12 Arboréscence

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 [9]. Par exemple, le graphe de la Figure 1.38 est une arborescence de racine a.

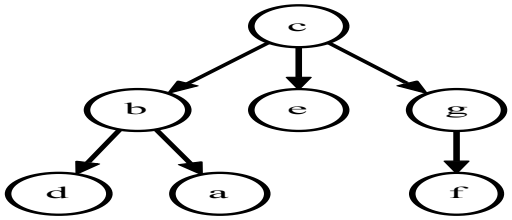


FIGURE 1.38 – Arboréscence

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 ALGORITHMES D'OPTIMISATION DANS LES GRAPHES

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 nous étudions plusieurs méthodes de résolution importantes en les illustrons avec simples exemples.

Graphes valués (pondérés)

Soit G(X, E, c) un graphe orienté ou :

- $X = \{x_1, x_2, ..., x_n\}$: est l'ensemble des sommets ;
- $E = \{e_1, e_2, ..., e_m\}$: est l'ensemble des arcs ;
- $c : E \mapsto V$: est une fonction qui associé a tout arc $e_{ij}(x_i, x_j)$ une capacité $c(x_i, x_j) \in V$ tel que V peut être (poids, longeur, temps, couleur,....etc). [12]

2.1 Recherche du plus court chemin

Soit un graphe G, on associe à chaque arc e de G une longueur c(e) telle que $\forall e \in E, c(e) \geq 0$. Soit a et b deux sommets de G, il s'agit de trouver un chemin $\mu(a,b)$ tel que $l(\mu) = \sum_{e \in \mu} c(e)$ soit le plus petit possible. $\mu(a,b)$ est alors appelé le plus court chemin de a à b.

2.1.1 Algorithme de Bellman

Principe

Cet algorithme ce base sur le calcul de l'arborescence des plus courtes distances, issue du sommet s à un sommet donné p.
On ne calcule la plus courtes 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.[6]

1. Initialisation
   Soit s un sommet de X, on pose $C = \{s\}$, $\Pi(s) = 0$ et $A = \emptyset$.

2. Chercher un sommet hors de C dont tous les prédécesseurs sont dans C.
   Si un tel sommet n'existe pas ; terminer.
   Dans ce cas soit $C = X$, ou le sommet s n'est pas une racine dans R.
   Si un tel sommet existe ; aller en (3)

3. On pose $\Pi(x) = min\{\Pi(I(e)) + c(e)\}$,tel que :

   – I(e) : sommet initial de l'arc e,

   – c(e) : poids de l'arc e.
   Soit $e^{'}$ l'arc pour lequel $\Pi(x) = \{\Pi(I(e^{'})) + c(e^{'})\}$
   On pose $A := A \cup \{e^{'}\}$; $S = S \cup \{x\}$; aller à (2).

Soit $R = (X, E, c)$ le réseau donné par la Figure 2.1 :

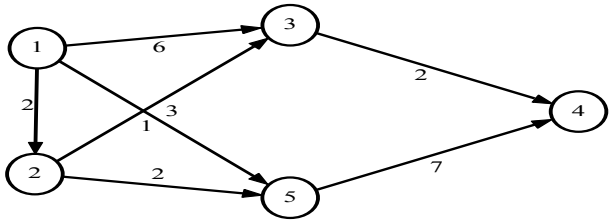


FIGURE 2.1 – Le réseau $R = (X, E, c)$

On pose :

$\Pi(1) = 0$, $C = \{1\}$ et $A = \varnothing$

- $2 \in \overline{C}$ et $\Gamma_2^{-1} \subset C$
- $\Pi(2) = \{\Pi(1) + c(1,2)\} = 0 + 2 = 2, e = (1,2) \in A$
- $C = \{1,2\}$, $A = \{(1,2)\}$
- $|C| \neq |X|$
- *Les sommets* $3, 5 \in \overline{C}$ et $\Gamma_3^{-1}, \Gamma_5^{-1} \subset C = \{1,2\}$ alors :
  $\Pi(5) = min\{0+1, 2+2\} = min\{1,4\} = 1, e = (1,5)$
  $\Pi(3) = min\{0+6, 2+3\} = min\{6,5\} = 5, e = (2,3)$
  $C = C \cup \{5,3\}$
  $A = A \cup \{(1,5),(2,3)\} = \{(1,2),(1,5),(2,3)\}$ et on a : $|C| \neq |X|$
- *Le sommet* $4 \in \overline{C}$ et $\Gamma_4^{-1} \subset C$ alors :
  $\Pi(4) = min\{1+7, 5+2\} = min\{8,7\} = 7, e = (3,4)$
  $C = C \cup \{4\} = \{1,2,3,4,5\}$
  $A = A \cup \{(3,4)\} = \{(1,2),(1,5),(2,3),(3,4)\}$ et on a : $|C| = |X|$, terminer.

D'où l'arborescence des plus courts chemin est donné par la Figure 2.2

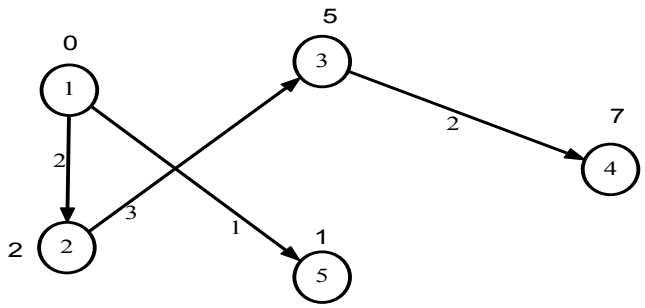


FIGURE 2.2 – Arborescence des plus court chemin de Bellman

2.1.2 Algorithme de Dijkstra

- Données : $R = (s, X, E)$ avec s :racine ;
- Résultat : Π, A, C avec :
  Π : Les plus courtes distances de s au sommet i, A :arborescence,
  C :sous-ensemble de sommets.

1. **Initialisation :** $C = \{1\}$, $\Pi(1) = 0$, $A = \emptyset$

$$\Pi(j) = \begin{cases} c_{ij} & \text{si } (i,j) \in A \\ +\infty & \text{sinon} \end{cases} \tag{2.1}$$

2. **Procédure de calcul :** chercher un sommet $i \in \overline{C}$ vérifiant :
   $\Pi(i) = min\{\Pi(j)\}$;
   Poser $C = C \cup \{i\}$ et $\Pi(j) = min\{\Pi(j), \Pi(i) + c_{ij}\}$, $j \in \Gamma(x) \cap C$

3. **Test d'arrêt :**
   Si $|C| = |X|$, terminer et le réseau R contient au moins un plus court chemin de s a tout sommet i ;
   Sinon, retourner en l'etape (2).

On applique l'algorithme de Dijkstra sur le graphe de la Figure 2.3 :

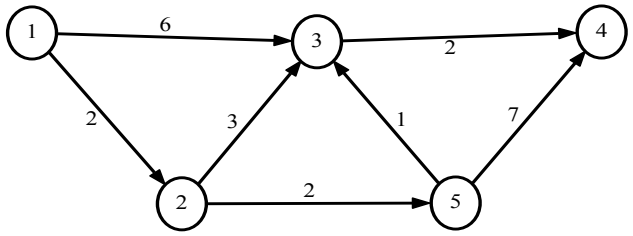


FIGURE 2.3 – Réseau R

$i \in \overline{C}$ vérifiant :$\Pi(i) = min\{\Pi(j)\}$; et

1. Initialisation :
   On pose $C = \{1\}, A = \varnothing, \Pi(1) = 0$

x	1	2	3	4	5
$\Pi(x)$	0	∞	∞	∞	∞

2. Examiner les arcs ayant leurs extrémités terminales $\notin C$ et leurs extrémités initiales dans C, ces arcs sont :$(1,2),(1,3)$
   $\Pi(1) + c(1,2) = 0 + 2 = 2 < \Pi(2) = \infty \Rightarrow \Pi(2) = 2$
   $\Pi(1) + c(1,3) = 0 + 6 = 6 < \Pi(3) = \infty \Rightarrow \Pi(3) = 6$
   $\Pi(i) = min\{\Pi(j), j \notin C\} = min\{\Pi(2), \Pi(3), \Pi(4), \Pi(5)\} = 2 \Rightarrow i = 2$
   Donc $C = \{1,2\}, A = \{(1,2)\}$. L'algorithme continue car : $|C| \neq |X|$;
3. On examine l'arc $(2,5)$
   $\Pi(2) + c(2,5) = 2 + 2 = 4 < \Pi(5) = \infty \Rightarrow \Pi(5) = 4$
   $\Pi(i) = min\{\Pi(j), j \notin C\} = min\{\Pi(3), \Pi(4), \Pi(5)\} = 4 \Rightarrow i = 5$
   $C = \{1,2,5\}, A = \{(1,2),(2,5)\}$, et : $|C| \neq |X|$;
4. En examine $(5,4),(5,3)$
   $\Pi(5) + c(5,3) = 4 + 1 = 5 < \Pi(3) = 6 \Rightarrow \Pi(3) = 5$

$\Pi(5) + c(5,4) = 4 + 7 = 11 < \Pi(4) = \infty \Rightarrow \Pi(4) = 11$
$\Pi(i) = min\{\Pi(j), j \notin C\} = min\{\Pi(3), \Pi(4)\} = 5 \Rightarrow i = 3$
$C = \{1,2,5,3\}, A = \{(1,2),(2,5),(5,3)\}, |C| \neq |X|$;

5. On examine l'arc $(3,4)$
   $\Pi(3) + c(3,4) = 5 + 2 = 7 < \Pi(4) = 11 \Rightarrow \Pi(4) = 7$
   $i = 4, C = \{1,2,5,3,4\}, A = \{(1,2),(2,5),(5,3),(3,4)\}, |C| = |X|$;

Le résultat obtenu par l'algorithme précedent est donné par la Figure 2.4

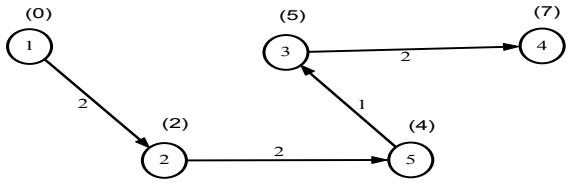


FIGURE 2.4 – L'arborescence des plus court chemin de Dijkstra

2.1.3 Algorithme de Ford

Le principe de l'algorithme de Ford consiste à améliorer une arborescence réalisable (initiale) (X,A) de racine s jusqu'à l'obtention d'une arborescence des plus courtes chemins, issue de s si celle-ci existe.

1. 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 l'arborescence (X,E).
2. Chercher un arc $e = (i,j)$ dans le réseau R, n'appartennant pas à A, tel que :
   $\sigma(e) = \Pi(j) - \Pi(i) - c_{i,j} > 0$
   Si un tel arc n'existe pas, terminer.
   Sinon, aller en (2).
3. Tester si $(X, E \cup \{e\})$ 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 en (3).

4. Chercher un arc $v \in E$ tel que : $T(v) = j = T(e)$.

   – On pose : $E = E \cup \{e\}/\{v\}$

   – Soit $X' \cup \{descendant\ de\ j\ dans\ l'arborescence\ A\}$.
   – On pose : $\Pi(y) = \Pi(y) - \sigma(e) \forall y \in X'$. Aller en (1).

   Soit le graphe de la Figure 2.5

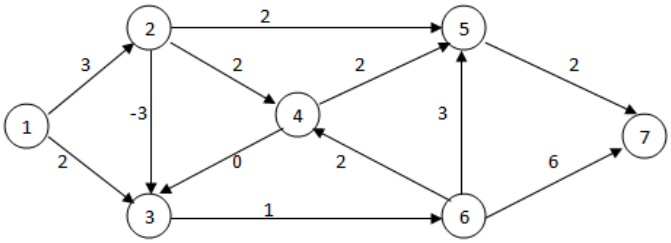


FIGURE 2.5 – Réseau possède un circuit

   On choisie une arborescence de la Figure 2.6

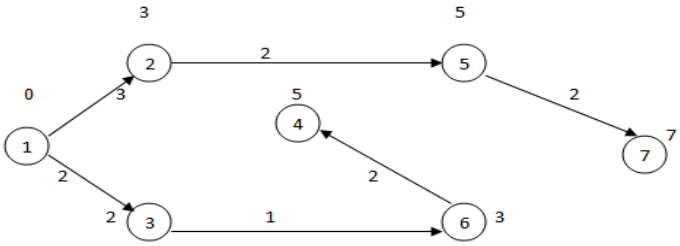


FIGURE 2.6 – L'arborescence de départ

1. L'arborescence $A = \{(1,2),(2,5),(5,7),(1,3),(3,6),(6,4)\}$
2. Les arcs qui $\notin A$ sont :
   $(2,4),(2,3),(4,3),(4,5),(6,5),(6,7)$
   $\sigma(2,4) = \Pi(4) - \Pi(2) - c_{2,4} = 5 - 3 - 2 = 0$
   $\sigma(2,3) = \Pi(3) - \Pi(2) - c_{2,3} = 2 - 3 + 3 = 2 > 0$

$\sigma(4,3) = \Pi(3) - \Pi(4) - c_{4,3} = 2 - 5 - 0 = -3$
$\sigma(4,5) = \Pi(5) - \Pi(4) - c_{4,5} = 5 - 5 - 2 = -2$
$\sigma(6,5) = \Pi(5) - \Pi(6) - c_{6,5} = 5 - 3 - 3 = -1$
$\sigma(6,7) = \Pi(7) - \Pi(6) - c_{6,7} = 7 - 3 - 6 = -2$
L'arc $e = (2,3) \notin A$ et $\sigma(e) > 0$ et $A \cup (2,3)$ ne contient pas un circuit.
Soit $v = (1,3) \in A$ tq : $T(e) = T(v) = 3$. Donc l'arc $(1,3)$ sort de l'arborescence et $(2,3)$ rentre dans l'arborescence, alors : $A = A \cup \{e\}/\{v\}$
$X' = \{3\} \cup \{succsseurs\ directs\ et\ indirects\ de\ 3\ dans\ A\} = \{3\} \cup \{6,4\}$
$= \{3,6,4\}$
$\Pi(3) = \Pi(3) - \sigma(2,3) = 2 - 2 = 0 \Longrightarrow \Pi(3) = 0$
$\Pi(6) = \Pi(6) - \sigma(2,3) = 3 - 2 = 1 \Longrightarrow \Pi(6) = 1$
$\Pi(4) = \Pi(4) - \sigma(2,3) = 5 - 2 = 3 \Longrightarrow \Pi(4) = 3$

Le résultat de l'algorithme ci-dessus est donné par la Figure 2.7

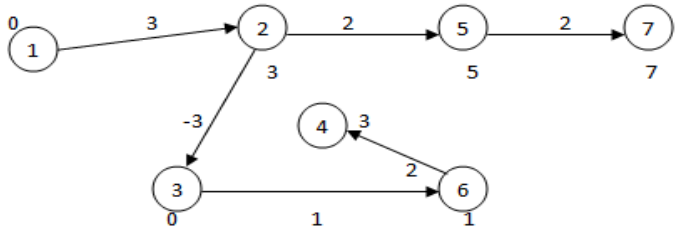


FIGURE 2.7 – L'arborescence A_1

3. Les arcs $\notin A_1$ sont : $(2,4), (1,3), (4,3), (4,5), (6,7), (6,5)$
$\sigma(2,4) = \Pi(4) - \Pi(2) - c_{2,4} = 3 - 3 - 2 = -2$
$\sigma(1,3) = \Pi(3) - \Pi(1) - c_{1,3} = -2$
$\sigma(4,3) = \Pi(3) - \Pi(4) - c_{4,3} = -3$
$\sigma(4,5) = 0$, $\sigma(6,7) = 0$, $\sigma(6,5) = 1 > 0$
L'arc $e = (6,5) \notin A$ et $\sigma(e) > 0$
$A \cup \{(6,5)\}$ ne contient pas un circuit alors l'arc (6,5) rentre et l'arc (2,5) sort de l'arborescence car $T(2,5) = T(6,5) = 5$
$X' = \{5\} \cup \{successeurs\ directs\ et\ indirects\ de\ 5\ dans\ A_1\} = \{5,7\}$
$\Pi(5) = \Pi(5) - \sigma(6,5) = 5 - 1 = 4$

$$\Pi(7) = \Pi(7) - \sigma(6,7) = 7 - 1 = 6$$

La seconde étape de l'algorithme de Ford est donné par la Figure 2.8

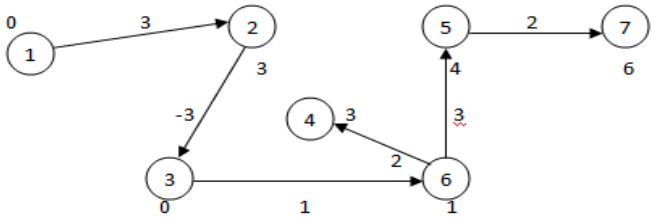


FIGURE 2.8 – L'arborescence des plus court chemin de Ford

Les arcs $\notin A_2 \quad \forall e \notin A_2,\ \sigma(e) \leq 0$ alors A_2 est l'arborescence optimale des plus court chemin.

2.2 Problème de l'arbre couvrant de poids minimal

Le problème de l'arbre de coût min consiste à la recherche de l'arbre dont la somme des valeurs sur ses arcs est min.

2.2.1 Algorithme de kruskal

Soit G(X, E, c) un graphe valué

1. Initiation : numéroter les arcs dans l'ordre croissant de leurs poids, $c(e_1) < c(e_2) < .... < c(e_m)$, $w = \emptyset$, $i = 1$;
2. Soit $w = w \cup \{e_i\}$ si $(X, w \cup \{e_i\})$ contient un cycle, aller en (3), sinon aller en (1) ;
3. On pose $w = w \cup \{e_i\}, i = \overline{i, m}$, aller en (3) ;
4. Si $i = m$ et $m = n - 1$ terminer, alors w est l'arbre max de poids min. Sinon $i = i + 1$ aller en (1), l'algorithme s'arrête lorsque $\mid w \mid = n - 1$.

On appliquons l'algorithme de Kruskal sur le graphe G de la Figure 2.9

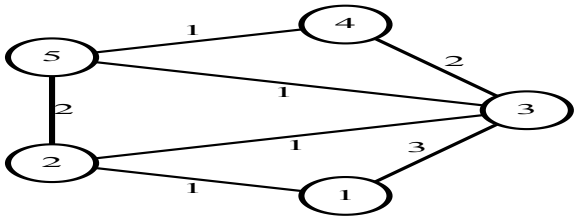


FIGURE 2.9 – Graphe G

i	1	2	3	4	5	6	7
e_i	(1,2)	(2, 3)	(3, 5)	(4, 5)	(4, 3)	(2, 5)	(1, 3)
$c(e_i)$	1	1	1	2	2	2	3

– Posons $w = \emptyset$, $i = 1$, $w = w \cup \{e_1\} = w \cup \{1, 2\} = \{1, 2\}$ on obtient la Figure 2.10

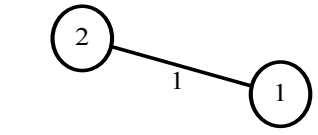


FIGURE 2.10 – Itération 1

– $\mid w \mid = 1$, $i = 2$

Ainsi, la deuxième itération de l'algorithme précédent est montré dans la Figure 2.11

– $w = w \cup \{e_2\} = \{(1, 2); (2, 3)\}$

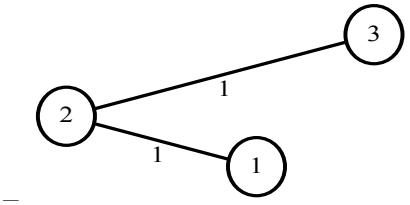


FIGURE 2.11 – Itération 2

– $\mid w \mid = 2$, $i = 3$

Dans la troixième itération, on obtient le graphe de la Figure 2.12

– $w = w \cup \{e_1\} = \{(1,2);(2,3);(3,5)\}$

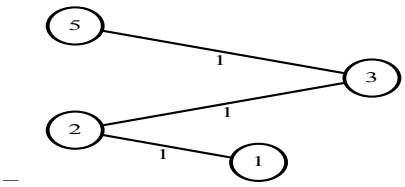


FIGURE 2.12 – Itération 3

– $\mid w \mid = 3,\ i = 4$

La quatrième itération de l'algorithme de Kruskal est montré par la Figure 2.13

– $w = w \cup \{e_1\} = \{(1,2);(2,3);(3,5);(4,5)\}$

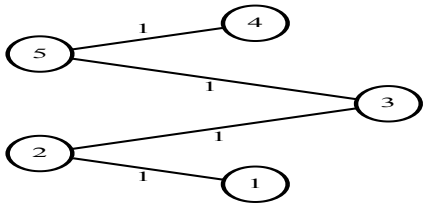


FIGURE 2.13 – Itération 4

– $\mid w \mid = 4,\ i = 5$

L'algorithme s'arrête car $\mid w \mid = n - 1 = 5 - 1 = 4$.

Remarque : Pour le problème de l'arbre max de poids max on applique le même algorithme en classant les arcs en valeur décroissant.

2.2.2 Algorithme de Prim

Principe :

On part d'un arbre initial A réduit à un seul sommet x (exemple $x = 1$), ensuite à chaque itération, on augmente l'arbre A en le connectant au plus proche sommet libre au sens des poids.

Soit I l'ensemble des noeuds déjà inclus dans l'arbre en construction et NI l'ensemble des noeuds non encore inclus.

1. Initialisation
   On pose $I = \emptyset$ et $NI = X$
2. Mettre le noeud de départ dans I
3. Si l'arbre est connexe ; terminer
4. Trouver un noeud b de NI dont la distance à un noeud quelconque a de I est minimale. On relie ce noeud a au b et faire
   - $I = I \cup \{b\}$
   - $NI = NI/\{b\}$
5. Aller en (3)

2.3 Problème de flot max de coupe min

Définition 4. *Le problème de flot maximal consiste à transporter la quantité maximale*
possible d'une origine (source) à une destination (puits) données, sans dépasser les capacités des arcs.[10]

Soit donc un réseau de transport de données noté R=(G=(X, E), s, p, c) avec :
- Un graphe oriénté $G = (X, E)$;
- $s \in X$ appelé sommet source ;
- $p \in X$ appelé sommet destination ou puits ;
- $c : E \mapsto N$ fonction capacité (à chaque arc e_{ij} associé une capacité $c_{ij} \geq 0$).

Un $s - p$ flot (f) est réalisable dans R s'il satisfait les contraintes suivantes :
- Contrainte de capacité

$$0 \leq f_{ij} \leq c_{ij} \quad \forall (i,j) \in E$$

- Contrainte de conservation de flot (Loi de Kirschff)

$$\sum_{i|(i,j)\in E} f_{ij} - \sum_{k|(j,k)\in E} f_{jk} = 0 \quad \forall j \in X \setminus \{s,p\}$$

(quantité qui entre dans x_j = quantité qui sort de x_j)

La valeur du flot est :

$$v(f) = \sum_{j|(s,j)\in E} f_{sj} = \sum_{j|(j,p)\in E} f_{jp}$$

Un arc (i , j) est dit saturé pour un flot f si $f_{ij} = c_{ij}$.[2]

2.3.1 Coupe minimum

Définition 5. *La coupe dans le graphe G = (X, X), définie par le sous-ensemble non vide de noeuds S, est l'ensemble d'arcs noté $\delta^+(S)$ sortant de S.*

$$\delta^+(S) = (i,j) \in E : i \in S, j \in X \setminus S$$

- Une s-p coupe est une coupe qui sépare s et p, i.e telle que $s \in S$ et $t \in X \setminus S$.
- La capacité d'une coupe est la somme des capacités de ses arcs.[12]

théorème 1. *(Dualité faible)*
La capacité d'une s-p coupe est supérieure ou égale à la valeur d'un s-p flot.[12]

théorème 2. *(Dualité forte)*
La valeur d'un s-p flot maximal est égale à la capacité d'une s-p coupe minimale.[12]

2.3.2 Algorithme de Ford-Fulkerson

- Idée de l'algorithme : trouver un chemin augmentant et augmenter le flot sur ce chemin.[2]
- Initialisation : $f = 0$.
- Alternance de deux phases :
  1. Phase de marquage (recherche d'un chemin augmentant).
  2. Phase d'augmentation (augmenter le flot sur le chemin trouvé en phase de marquage).

  Ces deux phases sont répétées jusqu'au moment où il n'existe plus de chemin augmentant.

Phase de marquage

1. Marquer s par $[0, \infty]$, $L = \{s\}$.
2. Tant que $L \neq \emptyset$; et t non marqué :

   a - Sélectionner i dans L et le retirer de L.

b - Pour tout j non marqué tel que $(i, j) \in X$ et $f_{ij} < c_{ij}$, marquer j par$[i, a_j]$ avec : $a_j = min(a_i, c_{ij} - f_{ij})$ et ajouter j dans L.

c - Pour tout j non marqué tel que $(i, j) \in X$ et $f_{ji} > 0$, marquer j par $[i, a_j]$ avec : $a_j = min(a_i, f_{ij})$ et ajouter j dans L.

3. Si t est non marqué, **STOP** (plus de chemin augmentant).

Phase d'augmentation

1. $j = t$
2. Tant que $j \neq s$:
   - Soit $[i, a_j]$ la marque de j. Si (i, j) est en avant, $f_{ij} = f_{ij} + a_t$. Si (i, j) est en arrière, $f_{ji} = f_{ji} - a_t$.
   - $j = i$

On applique l'algorithme de Ford-Fulkerson sur le graphe de la Figure 2.14

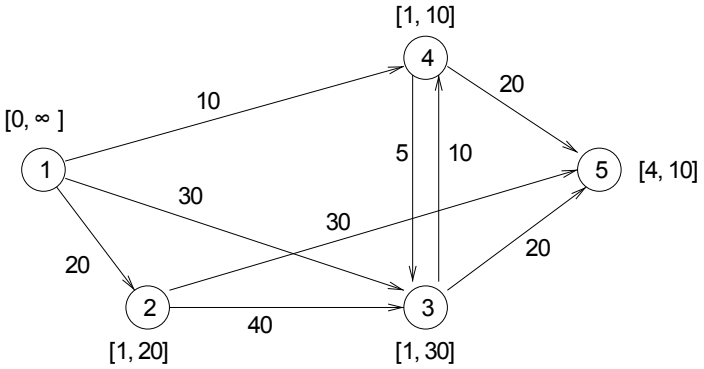


FIGURE 2.14 – Graphe de départ

$v(f) = 10$

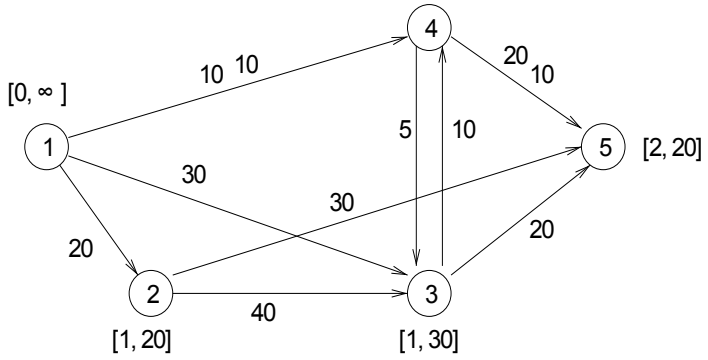


FIGURE 2.15 – Itération 1

$v(f) = 10 + 20 = 30$

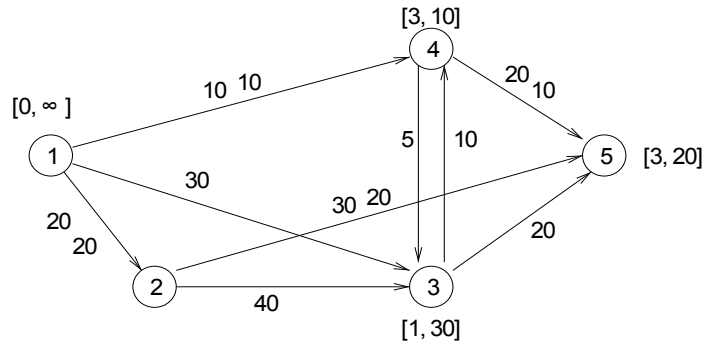


FIGURE 2.16 – Itération 2

$v(f) = 10 + 20 + 20 = 50$

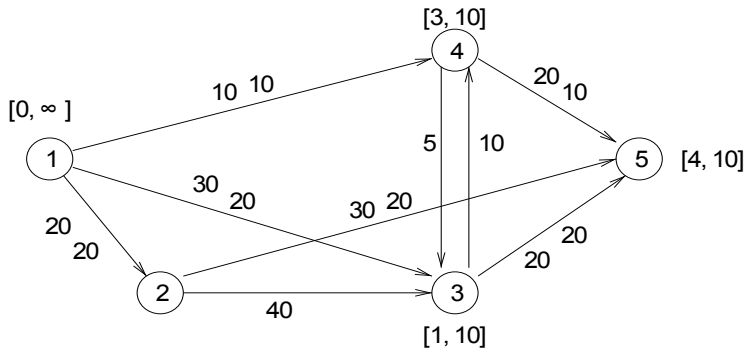


FIGURE 2.17 – Itération 3

$v(f) = 10 + 20 + 20 + 10 = 60$

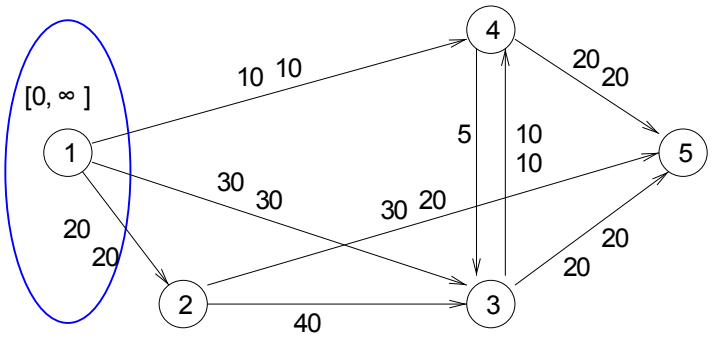


FIGURE 2.18 – Itération 4

Les noeuds marqués à la dernière itération définissent la coupe minimale $S = \{s\}$. On vérefié que $c(s,1) + c(s,2) + c(s,3) = 10 + 20 + 30 = 60 = v(f)$

2.4 Problème d'ordonnancement

Un projet est composé de N tâches ayant entre elles des contraintes d'antériorité. Chaque tâche i a une durée déterminée d_i ou le passage d'une tâche i à une tâche j a un délai d_{ij}. Le problème est de trouver un ordonnancement des tâches pour lequel le projet sera fini le plus tôt possible et de déterminer pour chaque tâche la date au plus tôt à laquelle elle peut commencer et la date au plus

tard à laquelle on peut la faire commencer sans retarder la date de fin des travaux. Pour cela on crée un graphe dont chaque sommet est une tâche et dont les arcs représentent les contraintes d'antériorité. Chaque arc (i,j) est valué soit par d_i soit par d_{ij}. Ce graphe est évidemment sans circuit. On rajoute un sommet D de début des travaux reliés par des arcs valués avec la valeur de 0 aux sommets sans prédécesseurs et un sommet F de fin des travaux reliés aux sommets sans successeurs par des arcs valués soit à d_i ou bien 0.
Afin de minimiser les retards apportés aux réalisations de projets, aux livraisons ponctuelles et aux clients, l'organisation des tâches devient indispensable et fait intervenir des notions très particulières.

2.4.1 Notions de projet, tâche et ordonnancement

Notion de projet

Un projet est un ensemble d'opérations permettant d'atteindre un objet fixé, ces opérations étant soumises à un certain nombre de contraintes.[11]

Notion de tâche

Une tâche est une opération dont l'ensemble de ces tâches forme le projet.[11]
On associe à chaque tâche sa durée et une contrainte d'antériorité par rapport aux autres tâches. On dira que la tâche i est immédiatement antérieure à j si j ne peut débuter que lorsque i est achevée.

Méthode d'ordonnancement

C'est un ensemble de méthodes qui permettent au responsable du projet de prendre les décisions nécessaires dans les meilleures conditions possibles, donc c'est un problème d'organisation du projet.[11]
Il existe plusieurs méthodes de résolution de ce problème à savoir :

Diagramme de Gantt

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

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

Ainsi la représentation des tâches précédentes est donnée dans la Figure 2.19 :

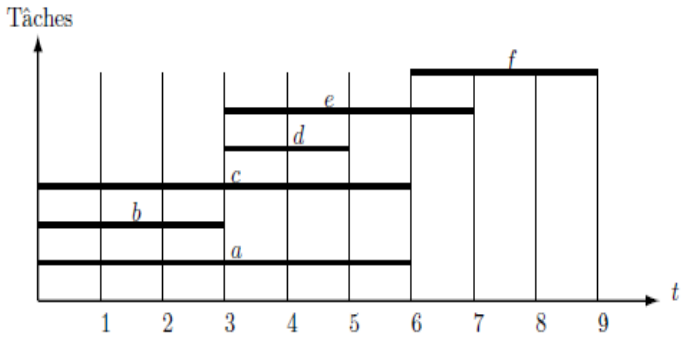


FIGURE 2.19 – Diagramme de Gantt

2.4.2 Méthode MPM

Cette méthode consiste à calculer pour châque tâche du projet :
– La date de début au plus tôt : notée par t_i est la date pour laquelle on peut commencer sa réalisation. Elle correspond à la date pour lesquelles toutes les

tâches antérieurs à i sont achevées, cette date est calculé de la façon suivante : Supposons que :t_d=0 telle que d représente le début du projet.

$t_i = max_{j \in P^-(i)}\{t_d + d_j\}$ avec :

t_i : représente le plus long chemin entre le sommet d et i

– La date de début au plus tard : notée par T_i est l'ultime date pour son exécution afin de déterminer le projet dans les délais telle que : $T_F = t_F$ et $T_i = min_{j \in P^+(i)}\{T_j - d_i\}$

Le graphe de la Figure 2.20 représente la construction d'un batiment, dont la durée de réalisation du projet est de 35 jours.

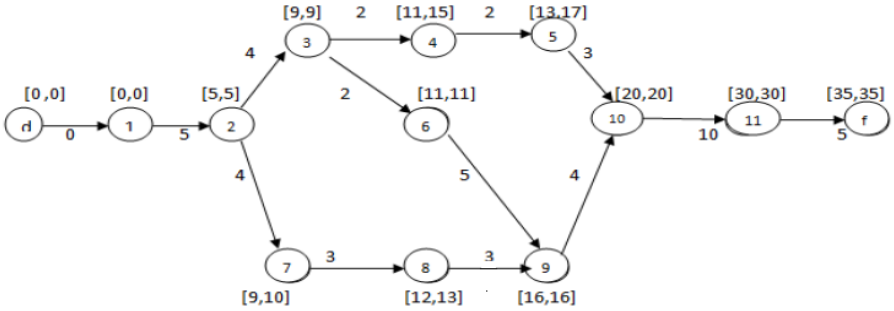


FIGURE 2.20 – Construction d'un batiment

Le chemin critique est composé des tâches critiques :1-2-3-6-9-10-11

Remarque

Une tâche i est dite critique si : $t_i = T_i$.

2.4.3 Méthode PERT

Pour cette méthode, on calcul les dates au plus tôt et au plus tard des évènements tel que la tâche (i,j) de durée d_{ij}, où i,j représentent respectivement l'évènement initial et terminal de cette tâche, avec la date de début au plus tôt d'un évènement i notée t_i est donné par :

$$\begin{cases} t_d = 0 \\ t_j = max_{j \in P^-(i)}\{t_i + d_{ij}\} \end{cases} \tag{2.2}$$

on calcul la date de début au plus tard d'un évènement i notée T_i :

$$\begin{cases} T_f = t_f \\ T_i = min_{j \in P^+(i)}\{T_j - d_{ij}\} \end{cases} \tag{2.3}$$

Dans le graphe de la Figure 2.21, la durée minimale de réalisation du projet est de 35 jours. Ainsi le chemin critique est : 1-2-3-6-9-10-11.

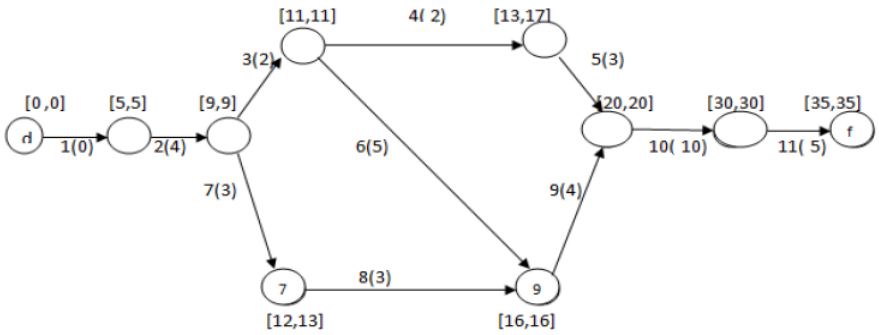


FIGURE 2.21 – Construction d'un batiment

2.5 Coloration dans les graphes

Problèmes

- Trouver le nombre minimal de couleurs pour colorier un graphe ;
- Rechercher un coloriage effectif pour un nombre donné de couleurs.

2.5.1 Coloration des sommets d'un graphe

La détermination du nombre chromatique $\gamma(G)$ qui désigne le nombre minimum de couleurs d'un graphe G, ainsi que l'obtention d'une coloration minimale des

sommets de G reste un problème assez complexe, en effet on procède de la manière suivante :

- Enumerer tous les ensembles stables maximum de G ;
- Chercher un recouvrement des sommets formé d'un nombre minimum d'ensemble stables maximaux ;
- Deduire de recouvrement une partition minimum de X en sous ensemble stables maximaux, et donc une coloration minimal de G. Cependant le nombre d'ensemble stables maximaux sera souvant trop important par ce que cette procedure soit accéptable, il est outil de recevoir à des algorithmes de coloration heuristiques simples et qui menant à une coloration des sommets de G non nécessairement minimal. L'algorithme ci-après nous permettons de répondre au problème précédent.

Algorithme de Powell

- Ordonner toute les sommets de G selon l'ordre de degrés décroissant ;
- Choisir une nouvelle couleur ;
- Chercher dans la liste le premier sommet non colorier et le colorié avec la nouvelle couleur ;
- Examiner tour à tour dans l'ordre de la liste tous les sommets non coloriés et colorier chaque sommet non adjacent à un sommet colorié avec la nouvelle couleur et retourner à l'étape (2).

On applique l'algorithme de Powell sur le graphe G_1 donné par la Figure 2.22 pour détermimer une k- coloration possible.

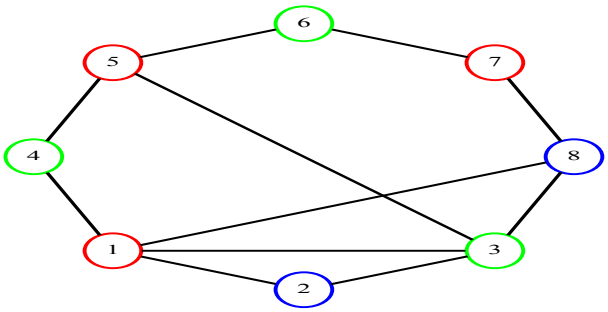


FIGURE 2.22 – Graphe G_1

i	1	2	3	4	5	6	7	8
$d_G(i)$	4	4	3	2	3	2	2	2
$d_G(i)$	1	2	3	5	4	6	7	8
c_i	c_1	c_2	c_3	c_2	c_1	c_2	c_3	c_1

Borne pour le nombre chromatique

L'algorithme précédent ne conduit pas nécessairement à une coloration minimale ($k \geq \gamma(G)$). Le nombre chromatique est généralement inconnu, c'est pour cette raison que la borne inférieure $\gamma^-(G)$ et la borne supérieure $\gamma^+(G)$ peuvent être établi pour $\gamma(G)$:

$$\gamma^-(G) \leq \gamma(G) \leq \gamma^+(G)$$

Proposition

Si $G = (X, E)$ est un graphe simple non orienté telle que : $n =\mid X \mid$, $m =\mid E \mid$

- $\alpha(G)$: la taille d'un stable maximum dans G.
- k : la k-colloration obtenu avec l'algorithme de Powell.
- d_{max} : le degré maximum dans G.
- d_{min} : le degré minimum dans G.
- $w(G)$: la taille d'une clique maximum dans G.

Alors, le nombre chromatique du G est comprés entre $\gamma^-(G)$ et $\gamma^+(G)$ avec :

- $\gamma^+(G) = min\{n + 1 - \alpha(G), k, d_{max} + 1\}$
- $\gamma^-(G) = max\{\frac{n}{n-d_{min}}, w(G), \frac{n^2}{n^2-2m}, \frac{n}{\alpha(G)}\}$

Exemple

Pour $n = 8$, $m = 11$, $k = 3$, $\alpha(G) = 3$, $d_{max} = 4$, $d_{min} = 2$, $w(G) = 3$ on trouve :

1. $\gamma^+(G) = min\{8 + 1 - 3, 3, 5\} = 3$
2. $\gamma^-(G) = max\{\frac{8}{8-2}, 3, \frac{64}{64-42}, \frac{8}{3}\} = 3$
3. $3 = \gamma^-(G) \leq \gamma(G) \leq \gamma^+(G) = 3$

2.5.2 Coloration des arêtes d'un graphe

Une coloration des arêtes du $G = (X, E)$ consiste à déterminer la plus petite valeur de k pour laquelle il existe une k-coloration des arêtes de graphe. Pour celui la on détermine d'abord le graphe adjoint.

Graphe adjoint

On appelle graphe adjoint d'un graphe $G = (X, E)$ le graphe $G' = (X', E')$ ou $X' = E$ et E' est l'ensemble des couples non ordonnées d'arêtes de G qui ont une extremité commune.

Remarque

L'ensemble des colorations des arêtes d'un graphe G est en bijection avec l'ensemble des colorations des sommets du graphe adjoint.

Algorithme de coloration des arêtes

1. Etant donné un graphe simple $G = (X, E)$;
2. Construire le graphe adjoint $G' = (X', E')$;
3. Colorier les sommets de G', en utilisant l'algorithme de Powell et la coloration obtenue est une k- coloration des arêtes du graphe G.

Exemple

Soit $G = (X, E)$ le graphe G_2 donné par la Figure 2.23 :

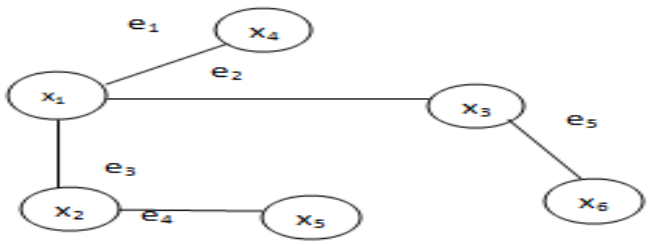


FIGURE 2.23 – G_2

On applique l'algorithme précédent, on obtient le graphe adjoint montré dans la Figure 2.24 ainsi que le nombre de coloration des arêtes possible.

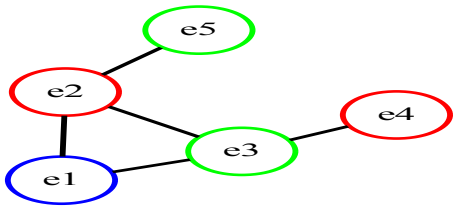


FIGURE 2.24 – Le graphe adjoint de G_2

e_i	e_1	e_2	e_3	e_4	e_5
$d(e_i)$	2	3	3	1	1
$\delta(e_i)$	3	1	2	4	5
c_i	c_3	c_1	c_2	c_1	c_2

Conclusion

Dans ce chapitre, nous avons présenté quelques problèmes de la théorie des graphes. Par la suite nous allons détailler quelques approches de résolution liée a ces problèmes, ou nous avons décrit les principes de ces méthodes.

CHAPITRE 3

QUELQUES PROBLÈMES CONCRET ET APPROCHES DE RÉSOLUTION

Dans ce chapitre, nous allons modéliser quelques situations et exemples sous forme des graphes. À l'aide des algorithmes de traitement des graphes, on cherche les solutions possibles pour chaqu'une.

3.1 Plus court chemin dans un réseau routier en Algérie

3.1.1 Plus court chemin d'une ville à 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 au 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 3.1 illustre le graphe modélisant ce problème.

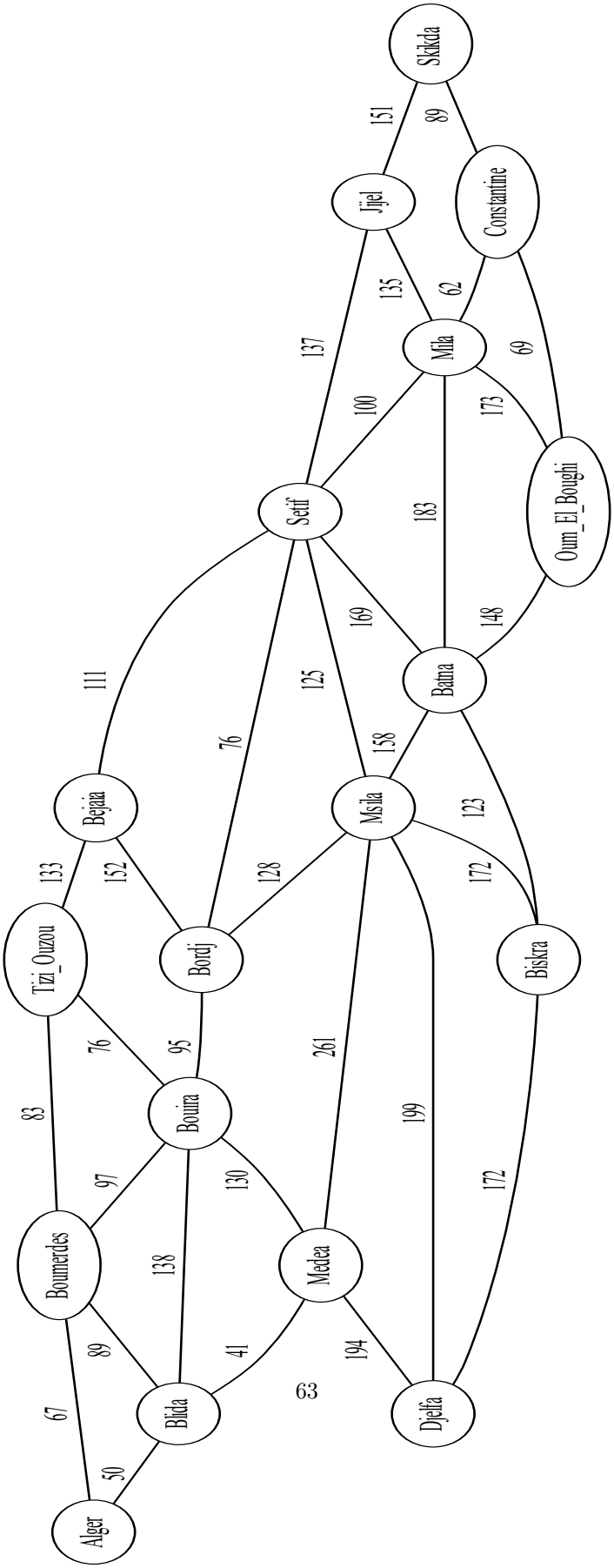


FIGURE 3.1 – Reseau routier en Algérie [7]

Pour simplifier et illustrer les différentes itérations de recherche de plus court chemin, on utilisera le graphe de la Figure 3.2 en remplaçant les noms des villes par leurs codes administratifs tels qu'ils sont indiqués dans le tableau ci-après.

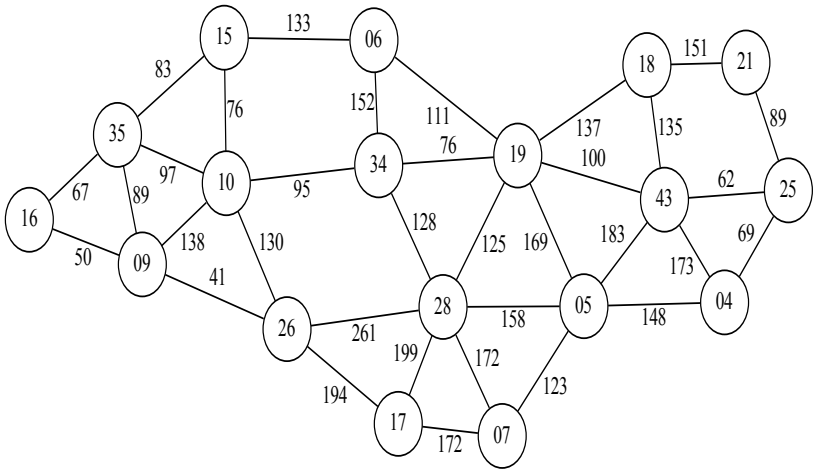
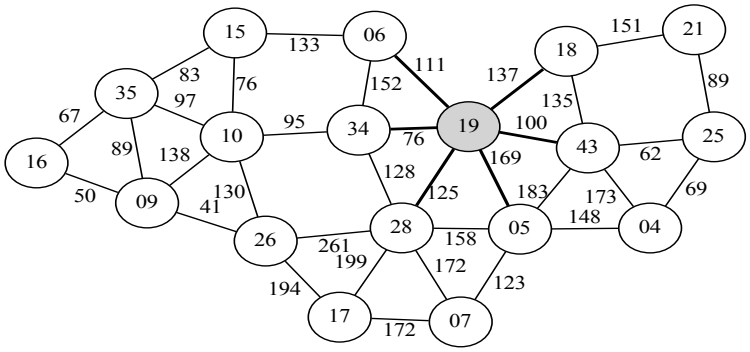


FIGURE 3.2 – Reseau routier simplifier

Code	Wilaya	Code	Wilaya
19	Sétif	34	Bordj Bou Arreridj
06	Béjaia	10	Bouira
15	Tizi Ouzou	09	Blida
35	Boumerdes	26	Medea
16	Alger	17	Djelfa
18	Jijel	07	Biskra
43	Mila	04	Oum El Boughi
05	Batna	25	Constantine
28	Msila	21	Skikda

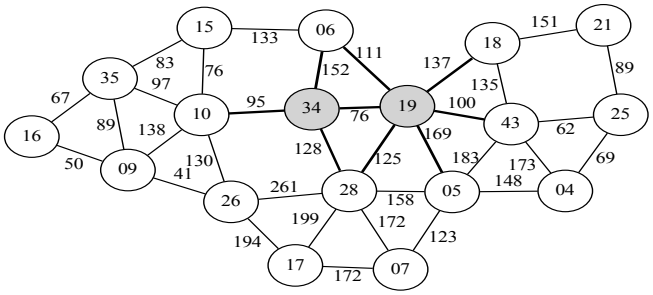
Supposant qu'un voyageur se trouve à la ville de Sétif cherche le plus court chemin entre les villes Sétif et Oum El Boughi on désigne la ville de Sétif comme ville de départ (O) et la ville de Oum El boughi comme ville d'arrivé (D), l'algorithme de Dijkstra s'arrête lorsque (D) est atteinte.

– **Itération 1**



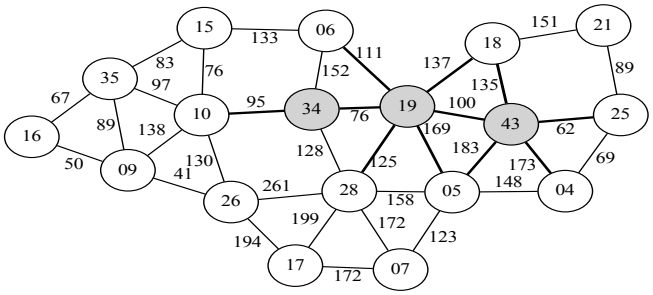
On examine les sommets voisins de sommet 19 ;
$\pi(06) = 111$,$\pi(18) = 137$,$\pi(43) = 100$,$\pi(05) = 169$,$\pi(28) = 125$,
$\pi(34) = 76$.
$min\{\pi(06), \pi(18), \pi(43), \pi(05), \pi(28), \pi(34), \} = 76 = \pi(34)$.
Le sommet 34 va être marqué.

– **Itération 2**



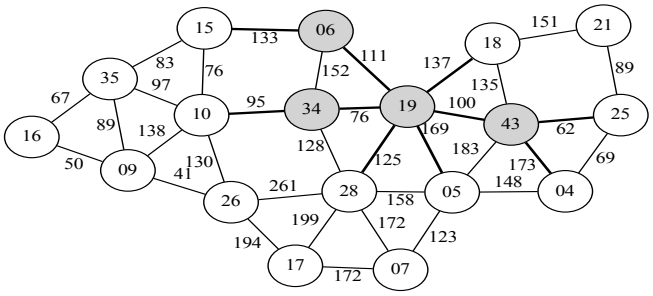
On examine les sommets voisins des sommets 19 et 34 ;
$\pi(06) = min\{111; 76 + 152\} = 111$,$\pi(18) = 137$,$\pi(43) = 100$,$\pi(05) = 169$,
$\pi(28) = min\{125; 76 + 128\} = 125$,$\pi(10) = 76 + 94 = 171$.
$min\{\pi(06), \pi(18), \pi(43), \pi(05), \pi(28), \pi(10)\} = \pi(43) = 100$.
Le sommet 43 va être marqué.

– **Itération 3**



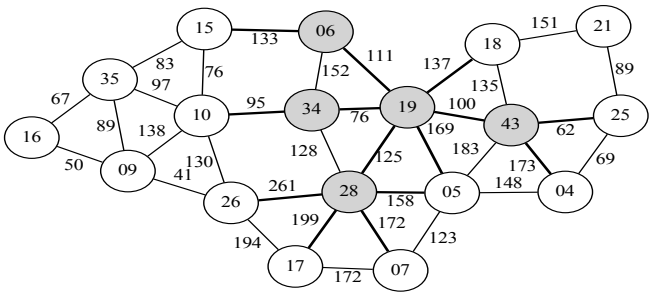
On examine les sommets voisins des sommets 19, 34 et 43 ;
$\pi(06) = 111$,$\pi(18) = min\{137; 100 + 135\} = 137$,
$\pi(05) = min\{169; 100 + 183\} = 169$,$\pi(28) = 125$,$\pi(10) = 171$,
$\pi(25) = 100 + 62 = 162$,$\pi(04) = 100 + 173 = 273$.,
$min\{\pi(06), \pi(18), \pi(25), \pi(04), \pi(05), \pi(28), \pi(10)\} = \pi(06) = 111$.
Le sommet 06 va être marqué.

– **Itération 4**



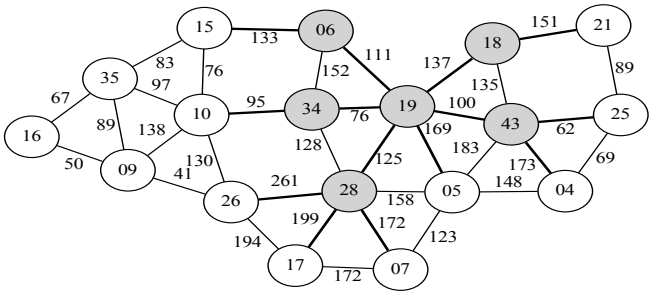
On examine les sommets voisins des sommets 19 , 34 , 43 et 06 ;
$\pi(15) = 111 + 133 = 244$,$\pi(18) = 137$, $\pi(25) = 162$,$\pi(04) = 273$,
$\pi(05) = 169$,$\pi(10) = 171$ $\pi(28) = 125$.
$min\{\pi(15), \pi(18), \pi(25), \pi(04), \pi(05), \pi(28), \pi(10)\} = \pi(28) = 125$.
Le sommet 28 va être marqué.

– **Itération 5**



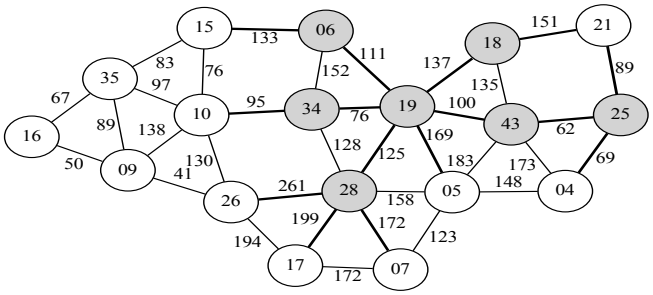
On examine les sommets voisins des sommets 19, 34, 43, 06 et 28 ;
$\pi(15) = 244$,$\pi(18) = 137$, $\pi(25) = 162$,$\pi(04) = 273$,$\pi(10) = 171$,
$\pi(07) = 125 + 172 = 297$ $\pi(26) = 125 + 261 = 386$,$\pi(17) = 125 + 199 = 324$,
$\pi(05) = min\{169; 125 + 158\} = 169$.
$min\{\pi(15), \pi(18), \pi(25), \pi(04), \pi(05), \pi(07), \pi(17), \pi(26), \pi(10)\} = \pi(18) = 137$.
Le sommet 18 va être marqué.

– **Itération 6**



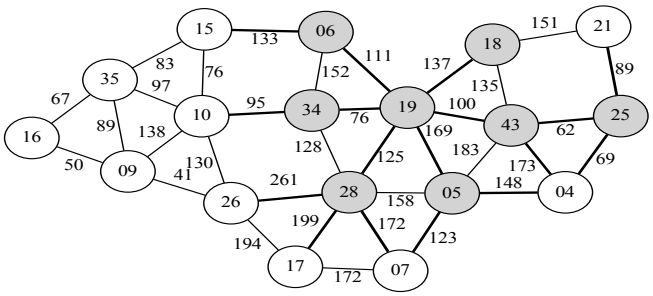
On examine les sommets voisins des sommets 19 , 34 , 43 , 06, 28 et 18 ;
$\pi(15) = 244$,$\pi(21) = 137 + 151 = 288$, $\pi(25) = 162$,$\pi(04) = 273$,
$\pi(05) = 169$,$\pi(10) = 171$, $\pi(26) = 386$,$\pi(17) = 324$,$\pi(07) = 297$.
$min\{\pi(15), \pi(21), \pi(25), \pi(04), \pi(05), \pi(07), \pi(17), \pi(26), \pi(10)\} = \pi(25) = 162$.
Le sommet 25 va être marqué.

– **Itération 7**



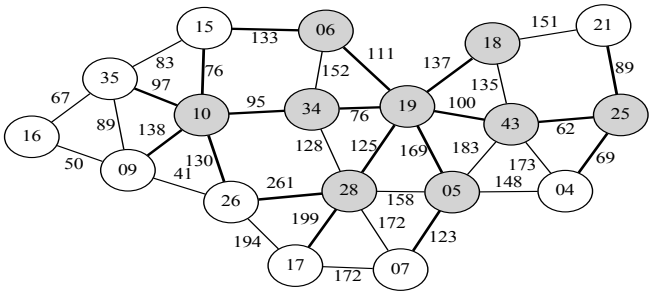
On examine les sommets voisins des sommets 19 , 34 , 43 , 06, 28, 18 et 25 ;
$\pi(15) = 244$,$\pi(21) = min\{137 + 151; 162 + 89\} = 251$,
$\pi(04) = min\{162 + 69; 100 + 173\} = 231$,$\pi(10) = 171$ $\pi(26) = 386$,
$\pi(17) = 324$,$\pi(07) = 297$,$\pi(05) = 169$
$min\{\pi(15), \pi(21), \pi(04), \pi(05), \pi(07), \pi(17), \pi(26), \pi(10)\} = \pi(05) = 169$
Le sommet 05 va être marqué.

– **Itération 8**



On examine les sommets voisins des sommets 34 , 43 , 06, 28, 05,18 et 25 ;
$\pi(15) = 244$,$\pi(21) = 251$,$\pi(26) = 386$,
$\pi(04) = min\{162 + 69; 100 + 173; 169 + 148\} = 231$,$\pi(10) = 171$
$\pi(17) = 324$,$\pi(07) = min\{169 + 123; 125 + 172\} = 292$,
$min\{\pi(15), \pi(21), \pi(04), \pi(07), \pi(17), \pi(26), \pi(10)\} = \pi(10) = 171$
Le sommet 10 va être marqué.

– **Itération 9**



On examine les sommets voisins des sommets 34 , 06, 28, 05,18 , 25 et 10 ;
$\pi(15) = min\{111 + 133; 171 + 76\} = 244$,$\pi(21) = 251$,$\pi(04) = 231$,
$\pi(07) = 292$,$\pi(17) = 324$, $\pi(26) = min\{125 + 261; 171 + 130\} = 301$,
$\pi(09) = 171 + 138 = 309$,$\pi(35) = 171 + 97 = 268$
$min\{\pi(15), \pi(21), \pi(04), \pi(07), \pi(17), \pi(26), \pi(09), \pi(35)\} = \pi(04) = 231$
Le sommet 04 va être marqué .
La ville de destination est atteinte ,on arrête l'algorithme .

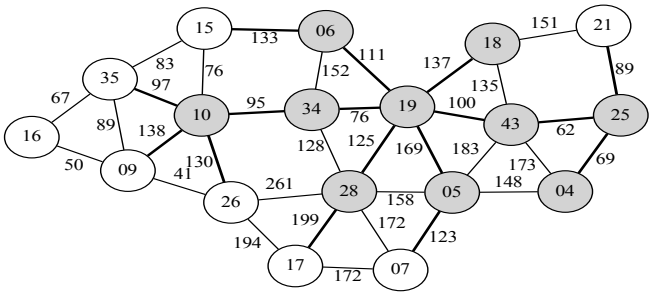


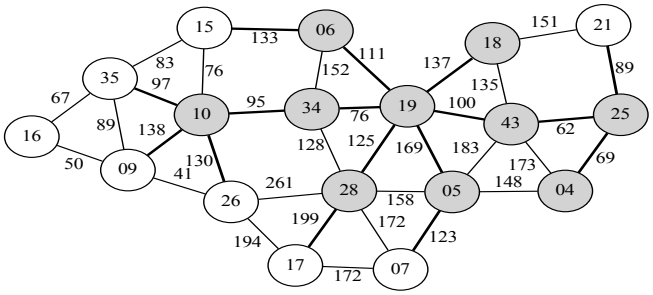
FIGURE 3.3 – Plus court chemin de ville de ""Sétif"" à " Oum El Boughi"

Le plus court chemin de ville de Sétif au ville de Oum El Boughi est :
Sétif→ Mila→ Constantine→ Oum El Boughi. Il est de valeur 231km.

3.1.2 Plus court chemin d'une ville à toutes les autres villes

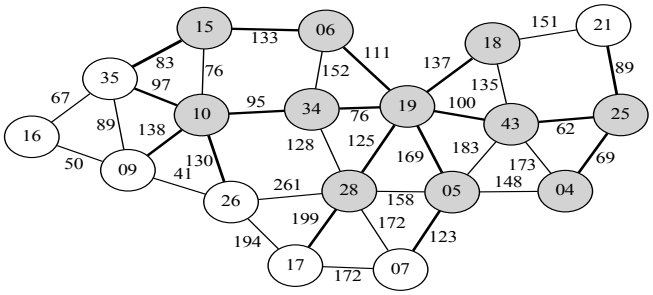
La découverte de plus court chemins d'une ville à tout les autre villes, se fait en répétant les itérations de l'algorithme de Dijkstra jusqu'à que tout les sommets soit parcourant. Pour trouver les plus courts chemins de ville de Sétif à toutes les autres villes, on répète les mêmes itérations passées jusqu'à marquer tous les sommets.

– **Itération 10**



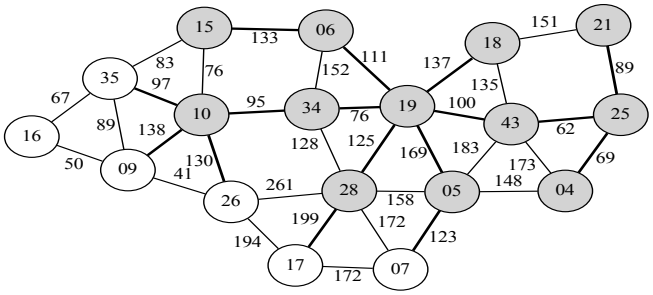
On examine les sommets voisins des sommets 06, 28, 18, 05 et 10 ;
$\pi(15) = 244$,$\pi(21) = 251$,$\pi(07) = 292$,$\pi(17) = 324$, $\pi(26) = 301$,
$\pi(09) = 309$,$\pi(35) = 268$
$min\{\pi(15), \pi(21), \pi(07), \pi(17), \pi(26), \pi(09), \pi(35)\} = \pi(15) = 244$
Le sommet 15 va être marqué.

– **Itération 11**



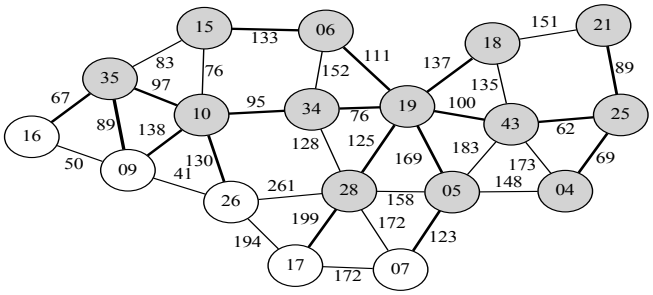
On examine les sommets voisins de sommets 28, 18, 10, 05 et 15 ;
$\pi(35) = min\{244 + 83; 171 + 97\} = 268$,$\pi(21) = 251$,$\pi(07) = 292$,
$\pi(17) = 324$, $\pi(26) = 301$,$\pi(09) = 309$,
$min\{\pi(35), \pi(21), \pi(07), \pi(17), \pi(26), \pi(09), \} = \pi(21) = 251$
Le sommet 21 va être marqué.

– **Itération 12**



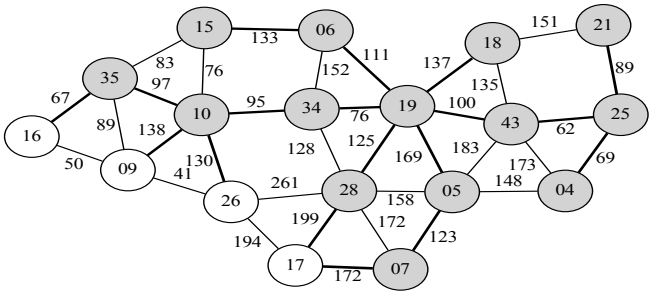
On examine les sommets voisins des sommets 15, 10, 28 et 05 ;
$\pi(35) = 268$,$\pi(07) = 292$,$\pi(17) = 324$, $\pi(26) = 301$,$\pi(09) = 309$,
$min\{\pi(35), \pi(07), \pi(17), \pi(26), \pi(09), \} = \pi(35) = 268$
Le sommet 35 va être marqué.

– **Itération 13**



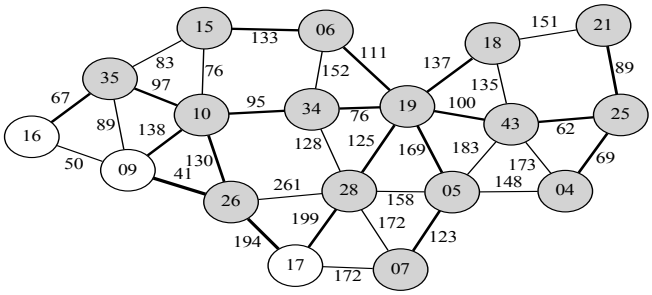
On examine les sommets voisins des sommets 35,10,28 et 05 ;
$\pi(16) = 268 + 67 = 335$,$\pi(07) = 292$,$\pi(17) = 324$, $\pi(26) = 301$,
$\pi(09) = min\{171 + 138; 268 + 89\}309$,
$min\{\pi(16), \pi(07), \pi(17), \pi(26), \pi(09), \} = \pi(07) = 292$
Le sommet 07 va être marqué.

– **Itération 14**



On examine les sommets voisins des sommets 35, 10, 07 et 28 ;
$\pi(17) = min\{125 + 199; 292 + 172\} = 324$, $\pi(16) = 335$
$\pi(26) = 301$,$\pi(09) = 309$,
$min\{\pi(17), \pi(26), \pi(09), \pi(16)\} = \pi(26) = 301$
Le sommet 26 va être marqué.

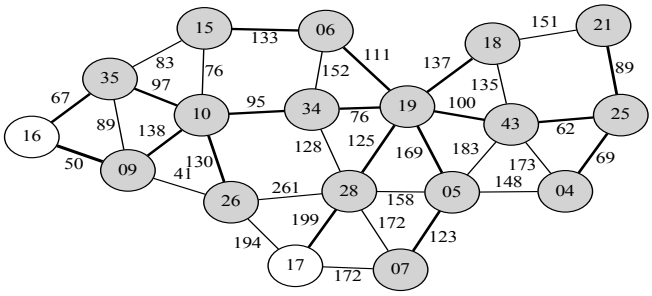
– **Itération 15**



On examine les sommets voisins des sommets 35,10, 26, 07 et 28 ;
$\pi(17) = min\{125 + 199; 301 + 194\} = 324$,$\pi(16) = 335$,
$\pi(09) = min\{301 + 41; 171 + 138\} = 309$,

$min\{\pi(17), \pi(09), \pi(16)\} = \pi(09) = 309$
Les sommet 09 va être marqué.

– **Itération 16**



On examine les sommets voisins des sommets 35, 09, 26,07 et 28 ;
$\pi(17) = 324$,$\pi(16) = min\{309 + 50; 268 + 67\} = 335$
$min\{\pi(17), \pi(16)\} = \pi(17) = 324$
Les sommets 16 et 17 vont être marqués.

– **Itération 17**

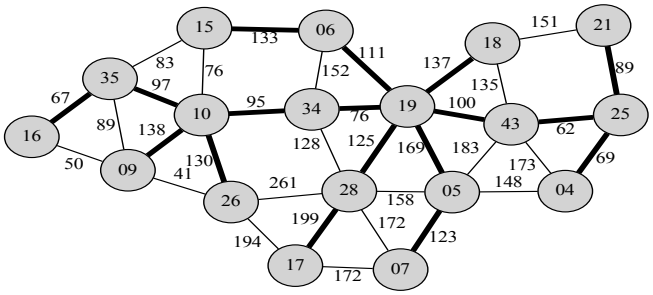


FIGURE 3.4 – plus court chemin de ville "Sétif" au toutes autre ville

Les plus courts chemins sont tels qu'ils sont illustrés dans le graphe de la Figure 3.4 précédent.

3.2 Problème de remplacement de matériel

Une location de voiture veut décider de la politique de remplacement de ses véhicules dans son parc automobile pour les 6 années prochaines. Sachant que le cout d'entretien annuel d'un véhicule croit avec son âge alors que son prix de revente diminue. La location se demande quant il est opportun de remplacer un véhicule par un autre neuf ?
Initialement on dispose d'un véhicule neuf et on veut déterminer à quelle période il faudra le remplacer par un autre neuf. On suppose d'être dans un environnement déterministe, c'est-à-dire que l'avenir est connu avec certitude au moins en ce qui concerne l'évolution des coûts.
Le coût d'achat d'un véhicule automobile est de 2 000 000. Le coût d'entretien annuel et la valeur de revente sont donnés dans le tableau ci-dessous selon l'âge du véhicule.

Age	1	2	3	4	5	6
Revente	1400000	1200000	800000	600000	400000	200000
Coût d'entretien	60000	100000	160000	240000	320000	440000

Une politique étant une suite de décision sur les 6 années prochaines, donc on cherche à déterminer la politique de coût minimale. Ce problème donne lieu au même type de modélisation que celui de la recherche d'un itinéraire optimale d'ou il est possible de le représenter par un problème de plus court chemin dans un graphe.

3.2.1 Modélisation du problème de remplacement de véhicule

A la fin de chaque année, on examine ce qu'on fait du véhicule, on le garde ou on le change. Si on décide de le garder, on décide à priori du nombre de nombre de période pendant lequel le conservera telle que on connait les conséquences en terme des coûts (achat, entretien, revente) . Les différentes étapes sont représentées par un sommet et a chaque arc de ce graphe on associe un nombre réel correspondant aux conséquences financières de la décision pour l'année : on suppose pour simplifier que le prix d'acquisition PA ne varie pas et on connaît le coût d'entretien

annuel CE_i ainsi que la valeur de revente RV_i d'un véhicule en fonction de son âge.

Par exemple, pour l'arc(0,1) le coût total annuel est de $PA+CE_1-RV_1 = 660000$: prix d'achat plus coût d'entretien de la première année moins valeur de revente au bout d'un an. Pour l'arc (0,2) le coût total annuel est de $PA+CE_1+CE_2-RV_2 = 960000$, il est le coût de remplacement d'un véhicule gardé 2 ans dans la location. Pour l'arc (0,3) le coût total annuel est de $PA+CE_1+CE_2+CE_3-RV_3 = 1520000$, il est le coût de remplacement d'un véhicule gardé 3 ans dans la location. Pour l'arc (0,4) le coût total annuel est de $PA+CE_1+CE_2+CE_3+CE_4-RV_4 = 1960000$ il est le coût de remplacement d'un véhicule gardé 4 ans dans la location. Pour l'arc (0,5) le coût total annuel est de $PA+CE_1+CE_2+CE_3+CE_4+CE_5-RV_5 = 2480000$ il est le coût de remplacement d'un véhicule gardé 5 ans dans la location. Pour l'arc (0,6) le coût total annuel est de $PA+CE_1+CE_2+CE_3+CE_4+CE_5+CE_6-RV_6 = 3120000$ il est le coût de remplacement d'un véhicule gardé 6 ans dans la location. Les différents coûts et décisions sont données dans la Figure 3.5 avec les coûts sont multipliés par 10^4).

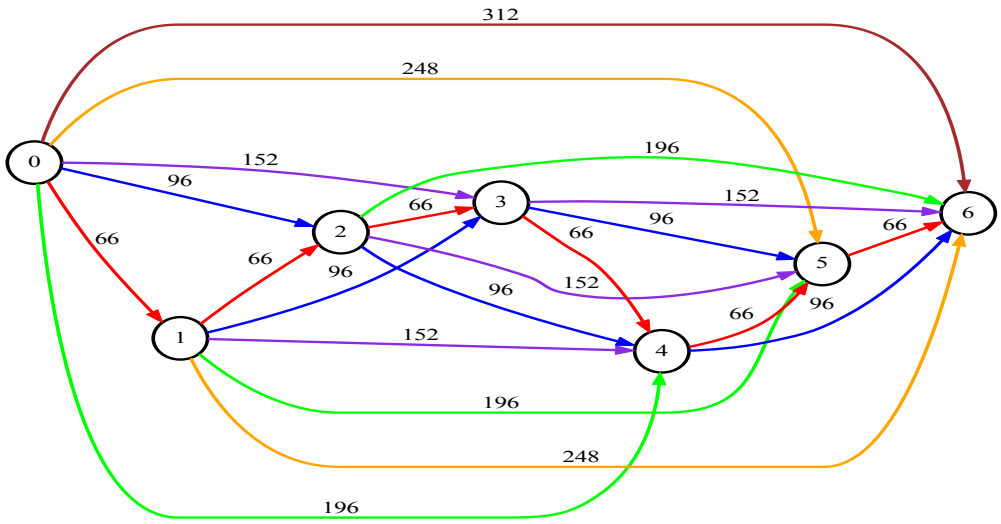


FIGURE 3.5 – graphe remplacement de véhicule

La recherche de la meilleure stratégie est modélisée par la recherche du plus

court chemin dans ce graphe du sommet 0 au sommet 6.
Notre graphe n'admet pas de circuit et notre problème est un problème de plus court chemin avec origine unique, donc on peut lui appliquer l'algorithme de Bellman-Ford.

- **Itération 1**
  On pose s=$\{0\}$(la racine), $\pi(0) = 0$, $C = \{0\}$,$A = \emptyset$,
  On examine les sommets dont $0 \in$ à leur ensembles des prédécesseurs :
  $\pi(1) = 66$,$\pi(2) = 96$,$\pi(3) = 152$,$\pi(4) = 196$,$\pi(5) = 248$,$\pi(6) = 312$
  $min\{\pi(1), \pi(2), \pi(3), \pi(4), \pi(5), \pi(6), \} = 66 = \pi(1)$.
  D'ou $A = \{(0, 1)\}$.
- **Itération 2**
  On examine les sommets dont $\{0, 1\} \in$ à leur ensembles des prédécesseurs :
  $\pi(2) = min\{66 + 66; 96\} = 96$,$\pi(3) = min\{66 + 96; 152\} = 152$,
  $\pi(4) = min\{66 + 152; 196\} = 196$, $\pi(5) = min\{66 + 196; 248\} = 248$,
  $\pi(6) = min\{66 + 248; 312\} = 312$
  $min\{\pi(2), \pi(3), \pi(4), \pi(5), \pi(6), \} = 96 = \pi(2)$.
  D'ou $A = \{(0, 1); (0, 2)\}$.
- **Itération 3**
  On examine les sommets dont $\{0, 1, 2\} \in$ à leur ensembles des prédécesseurs :
  $\pi(3) = min\{66 + 96; 152\} = 152$,$\pi(4) = min\{66 + 152; 96 + 96; 196\} = 192$,
  $\pi(5) = min\{96 + 152; 66 + 196; 248\} = 248$,
  $\pi(6) = min\{312; 66 + 248; 96 + 196\} = 292$
  $min\{\pi(3), \pi(4), \pi(5), \pi(6), \} = 152 = \pi(3)$.
  D'ou $A = \{(0, 1); (0, 2); (0, 3)\}$.
- **Itération 4**
  On examine les sommets dont $\{0, 1, 2, 3\} \in$ à leur ensembles des prédécesseurs :
  $\pi(4) = min\{66 + 152; 96 + 96; 196; 152 + 66\} = 192$,
  $\pi(5) = min\{66 + 196; 96 + 152; 66 + 196; 248\} = 248$,
  $\pi(6) = min\{312; 66 + 248; 96 + 196; 152 + 152\} = 292$
  $min\{\pi(4), \pi(5), \pi(6), \} = 192 = \pi(4)$.
  D'ou $A = \{(0, 1); (0, 2); (0, 3); (2, 4)\}$.

– **Itération 5**

On examine les sommets dont $\{0, 1, 2, 3, 4\} \in$ à leur ensembles des prédécesseurs :

$\pi(5) = min\{96 + 96 + 66; 96 + 152; 66 + 196; 248\} = 248$,

$\pi(6) = min\{312; 66 + 248; 96 + 196; 152 + 152; 96 + 96 + 96\} = 288$

$min\{\pi(5), \pi(6), \} = 248 = \pi(5)$.

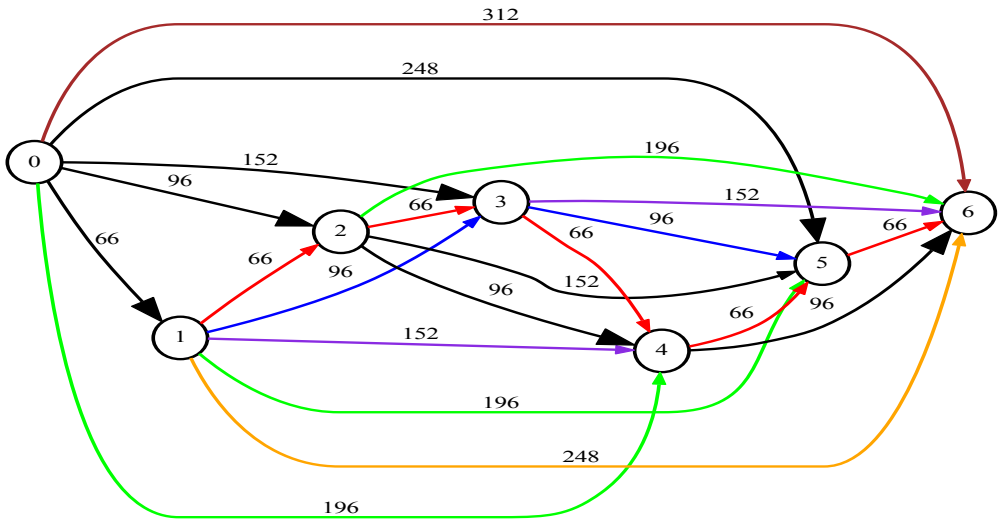
D'ou $A = \{(0, 1); (0, 2); (0, 3); (2, 4); (0, 5)\}$.

– **Itération 6**

On examine les sommets dont $\{0, 1, 2, 3, 4, 5\} \in$ à leur ensembles des prédécesseurs :

$\pi(6) = min\{312; 66 + 248; 96 + 196; 152 + 152; 96 + 96 + 96\} = 288$.

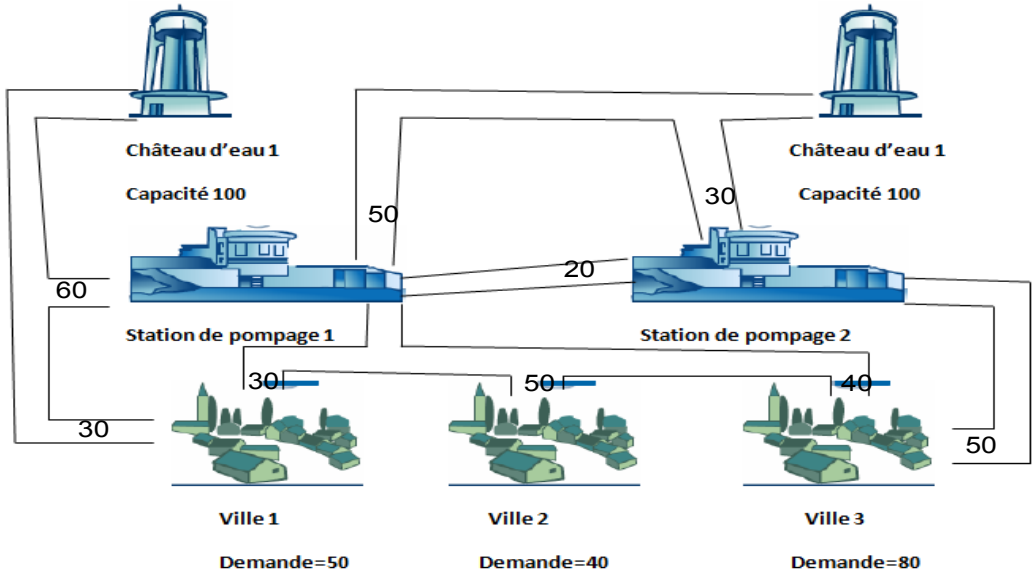
D'ou $A = \{(0, 1); (0, 2); (0, 3); (2, 4); (0, 5); (4, 6)\}$.



Le plus court chemin de 0 à 6 est $0 \to 2 \to 4 \to 6$ et de valeur 2 880 000DA. Donc la meilleur politique de remplacement des voitures pour cette location est de remplacer ses voitures chaque 2 ans car cette politique lui coutera moins chère.

3.3 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 exprimé 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 suivant le long des canalisations.



Un premier problème est de déterminer s'il est possible de satisfaire à travers ce réseau la demande des 3 villes et comment ? Pour résoudre ce problème il faut dans un premier temps le modéliser.

Modélisation du problème de distribution d'eau par 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, C1) et (s, C2) avec une

capacité supérieure ou égale à 100. La conservation des flux au sommet C1 permet de traduire qu'il ne peut pas partir de C1 une quantité supérieure à 100. Il en est de même pour C2. 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 3.6 :

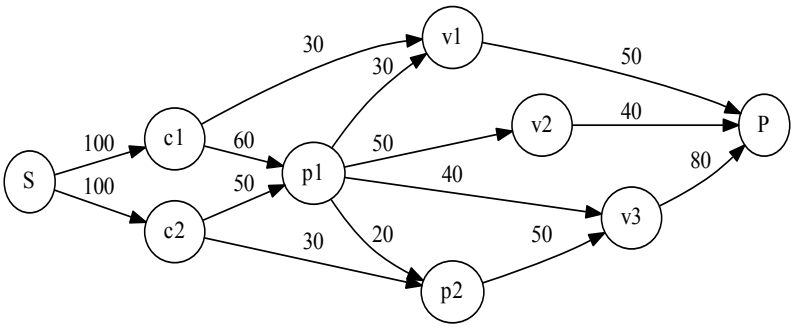


FIGURE 3.6 – Réseau de canalisation d'eau

Résolution du problème de flot maximal

Pour résoudre ce problème ont lui applique l'algorithme de Ford-Fulkerson dont les itérations sont illustrées dans les graphes suivants :

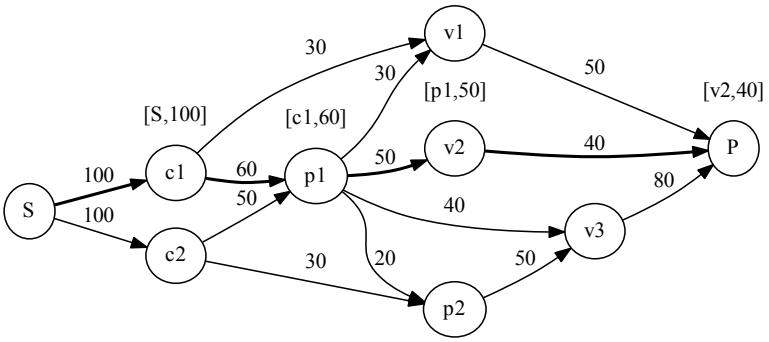


FIGURE 3.7 – Itération 1

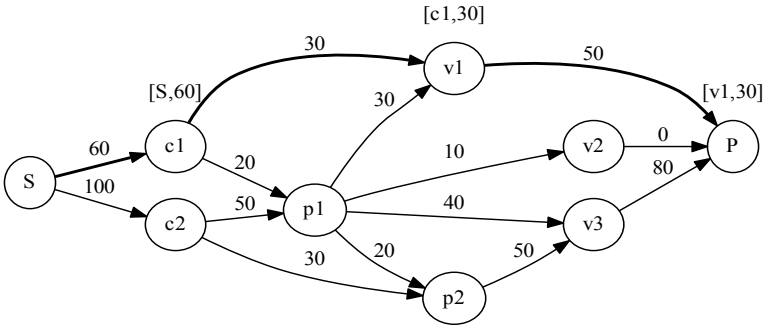


FIGURE 3.8 – Itération 2

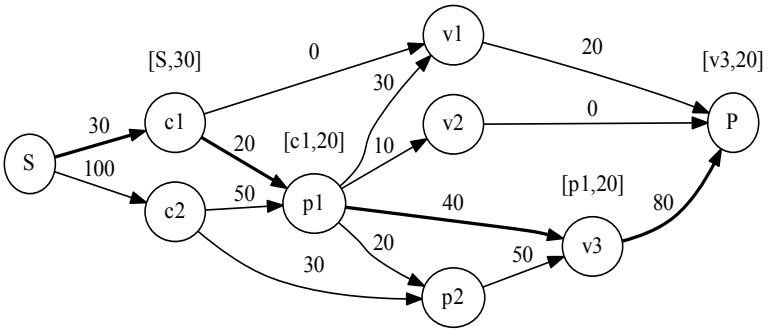


FIGURE 3.9 – Itération 3

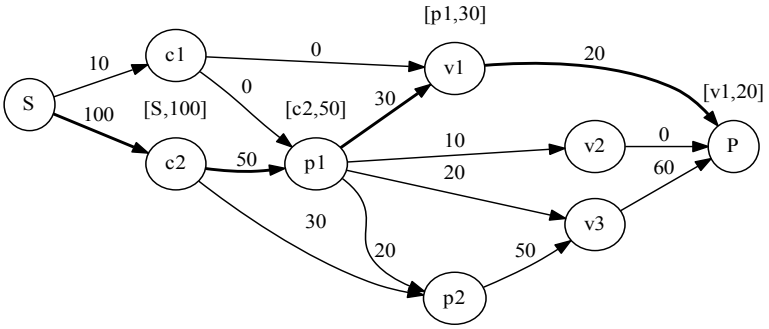


FIGURE 3.10 – Itération 4

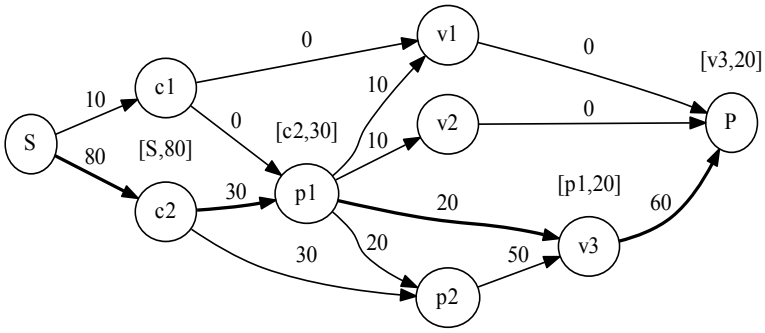


FIGURE 3.11 – Itération 5

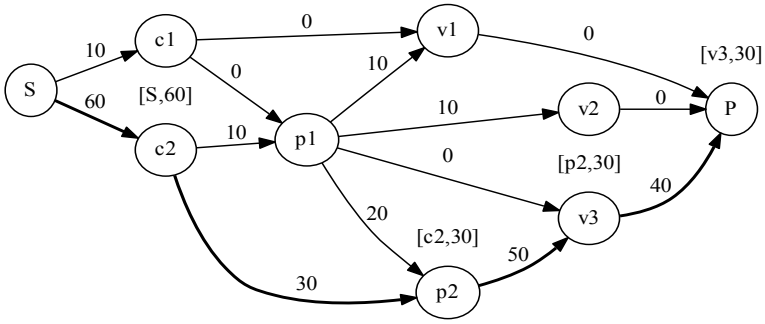


FIGURE 3.12 – Itération 6

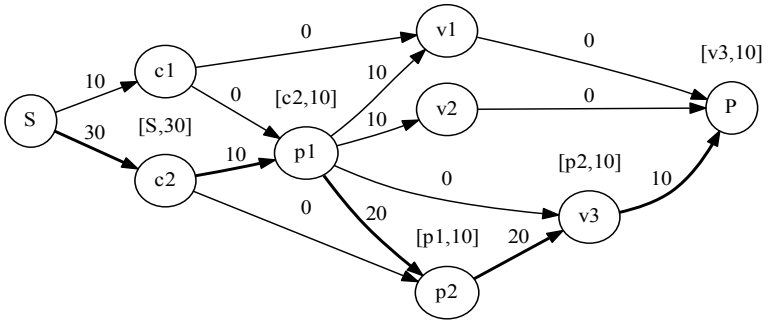


FIGURE 3.13 – Itération 7

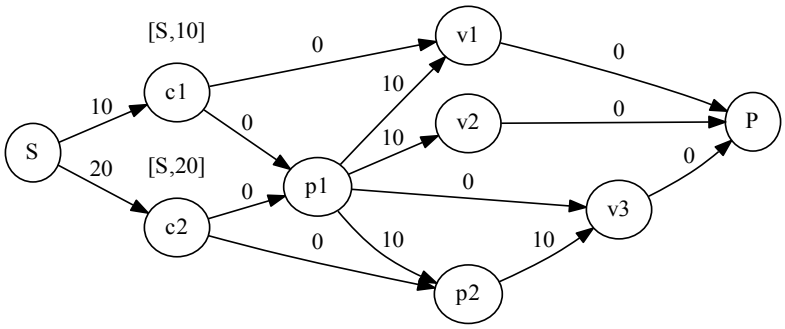


FIGURE 3.14 – Itération 8

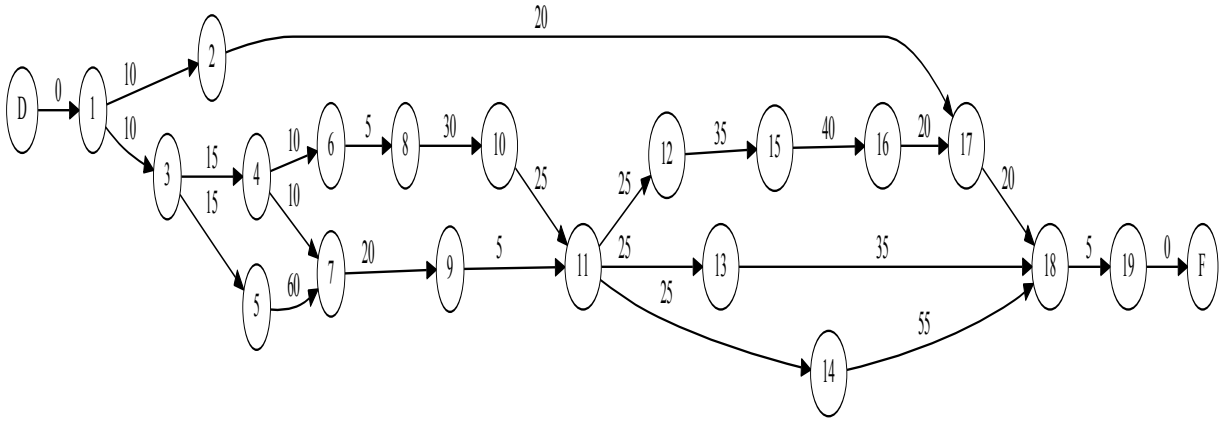
Le puits n'est pas marqué, d'ou l'algorithme s'arrête. Le flot maximal égale : $40+30+20+20+20+30+10=170$. Les résultats trouvés montrent qu'à travers ce réseau, il est possible de satisfaire les demandes des 3 villes. On remarque à travers la dernière itération que la procédure de marquage est arrêté dans c1 et c2 et que les canalisations de p1 et p2 aux 3 villes n'est pas nulle, donc on peut proposer aux services d'eau des 3 villes, qu'il est possible d'augmenter la quantité d'eau qui leur fournée en finançant des travaux pour augmenter la capacité des canalisations de (c1-p1) de 10 et (c2-p1) de 20 en milliers de m^3.

3.4 Ordonnancement des tâches d'un projet

Le problème central de l'ordonnancement s'énoncé comme suit : étant donné un projet constitué de n tâches de durée d'exécution fixes et soumises à des contraintes de l'antériorité, on cherche à déterminer un calandrier d'exécution ou un ordonnancement qui minimise la durée de réalisation totale du projet. Considérant une réalisation industrielle comporte un certain nombre d'opérations définies par le tableau suivant :

Tâche	Durée (jours)	Tâches précédentes
1	10	0
2	20	1
3	15	1
4	10	3
5	60	3
6	5	4
7	20	4, 5
8	30	6
9	5	7
10	25	8
11	25	9, 10
12	35	11
13	35	11
14	55	11
15	40	12
16	20	15
17	20	16
18	5	13, 14, 17

Ce problème peut être modélisé sous forme d'un graphe telle que, les sommets représentent les tâches et que deux tâches successives sont reliés par un arc valué de nombre de jours de réalisation de cette tâche. Le graphe est le suivant :



Les dates au plus tôt et au plus tard sont calculé en utilisant la méthode Pert et ils son montrées dans la Figure 3.15 :

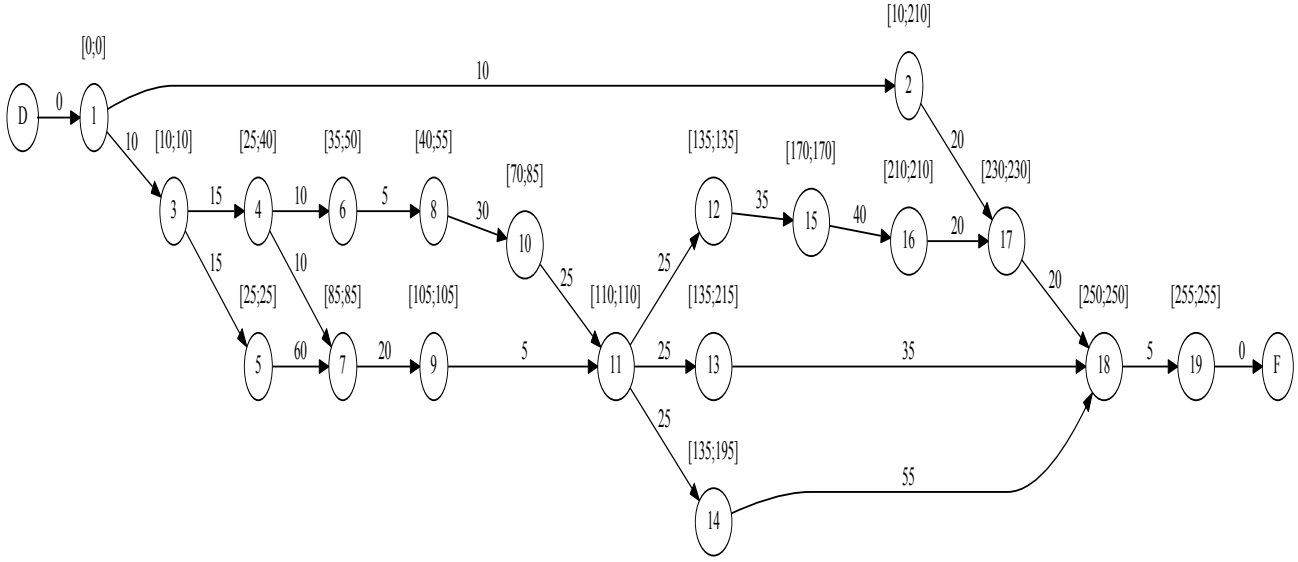


FIGURE 3.15 – Les dates aux plus tôt et aux plus tard

Le calcul de la marge totale a donnée les résultats illustrés dans la Figure 3.16, d'ou la définition de chemin critique qui est précisé en gras.

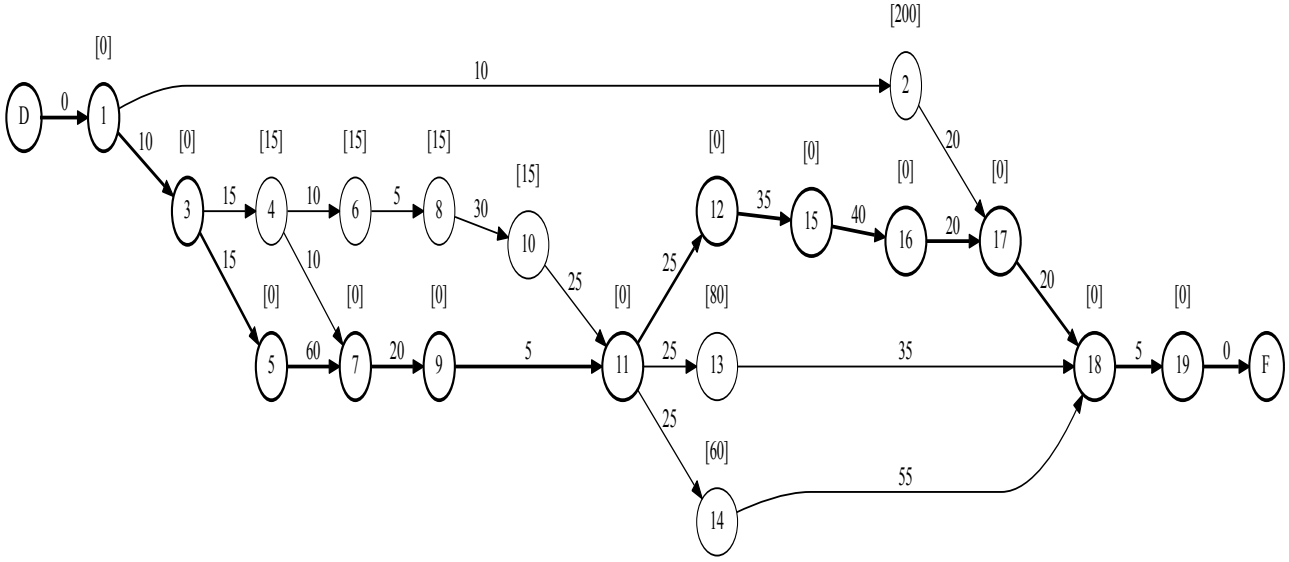


FIGURE 3.16 – les marge totales et chemin critique

Le chemin critique de projet est :
$0 \to 1 \to 3 \to 5 \to 7 \to 9 \to 11 \to 12 \to 15 \to 16 \to 17 \to 18 \to 19$.
La date de réalisation de projet est alors, 255 jours.

3.5 Stockage et transport des produits chimiques

Dans le domaine de la production ; pour atteindre un objectif ; on doit totalement respecter certaines conditions. L'une des plus importantes est celle qui concerne les produits chimiques et de ne pas confondre entre les différentes variétés. Soient disons un ensemble de N produits chimiques de poids connus a_i doit être conditionné dans des boites. Chaque boite peut recevoir 2 objets dont le poids
total n'excède pas B(on suppose que le poids maximal B est de 20 unités) et pour lesquels aucune incompatibilité n'existe. En effet, du fait que ces produits soient

chimiques des précautions d'usage interdisent que certains d'entre eux soient assemblés ensemble. On désire connaitre le nombre minimal de boites à utiliser pour conditionner ces objets.

On considère 6 objets dont les poids sont donnés dans le tableau suivant :

Objet	1	2	3	4	5	6
Poids	9	10	8	12	7	11

Les objets 1, 3 et 5 sont incompatibles entre eux et les objets 2, 4 et 6 sont également incompatibles entre eux.

Modélisation du problème :

Soit a_i et a_j : représentent les poids des produits chimiques. Le nombre de boite minimale nécessaire pour tous les déférents produits chimiques représente un sous ensemble d'arêtes donc on le représente par un vecteur booléen :

$$x_{ij} = \begin{cases} 1 & \text{si } (\ a_i + a_j \leq B) \\ 0 & \text{sinon} \end{cases} \tag{3.1}$$

Dans notre problème on cherche à minimiser le nombre de boites :

$$\sum_{i=1}^{N} \sum_{j=1}^{N} x_{i,j} \rightarrow min$$

La construction du graphe $G = (V, E)$ dont les sommets représentent les différents produits chimiques numéroté de 1 à 6, une arête relie deux de ces sommets lorsque deux produits sont compatibles et leur poids ne dépasse pas $B = 20$.

Supposons que notre graphe sera de la forme illustré dans la Figure 3.17

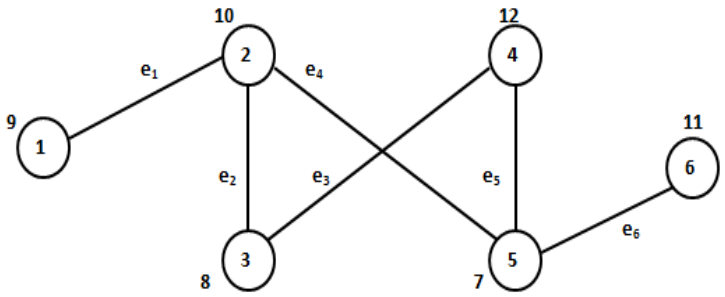


FIGURE 3.17 – Compatibilité des produits chimiques

Résolution du problème :

Pour résoudre ce type des problèmes, on utilise la coloration des arêtes. Donc on va construire d'abord le graphe adjoint donné dans la Figure 3.18

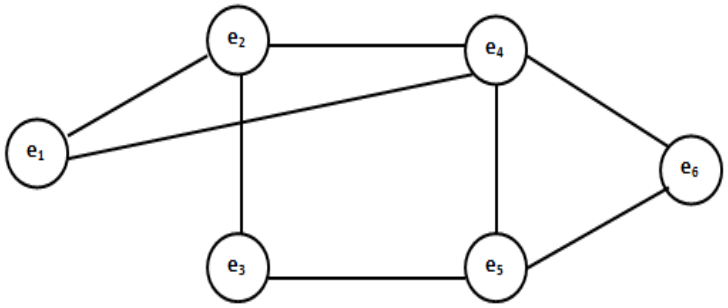


FIGURE 3.18 – Compatibilité des produits chimiques (graphe adjoint)

Puis, on applique l'algorithme de Powell vu dans le chapitre précédent, afin d'obtenir le nombre minimal de couleurs nécessaire pour colorier un graphe qu'étant donné.

e_i	e_1	e_2	e_3	e_4	e_5	e_6
$d(e_i)$	2	3	2	4	3	2
$d(e_i) \searrow$	4	3	5	1	2	6
c_i	c_2	c_3	c_1	c_1	c_2	c_3

Le nombre minimal nécessaire pour colorier les arêtes de graphe précédent égale à 3 comme la Figure 3.19 montre.

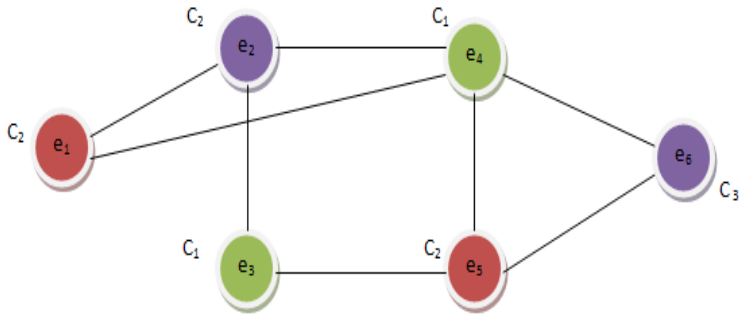


FIGURE 3.19 – Compatibilité des produits chimiques (coloration des sommets du graphe adjoint)

On revient au problème original, on aura la coloration des arêtes donnée par la Figure 3.20

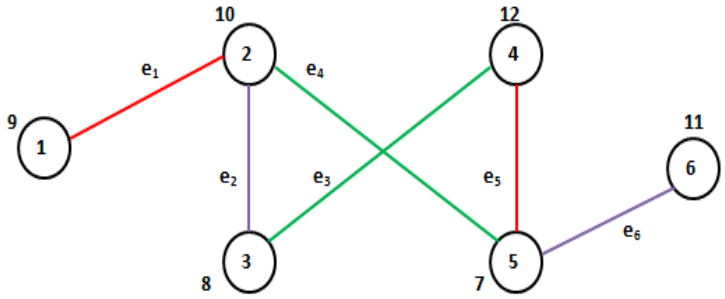


FIGURE 3.20 – Compatibilité des produits chimiques (coloration des arêtes du graphe G)

En interprète les résultats trouvés comme suit :
Les produits qui se passent sur les arcs de la même couleur ne peuvent pas être rangés et transportés dans une même boite sans risques d'explosion. En effet, le fait que ces produits ayant la même couleur donc ils sont non adjacent c-à-d qu'ils sont incompatibles entre eux. Donc la seule possibilité parmi les solutions réalisables trouvés et lesquelles vérifiées les deux contraintes précédentes ainsi repend à notre objectif dans le but est de réaliser le transport des produits sans risque et au moindre coût c -à-d en utilisant le moins de boites possible est que le produit 1 sera ensemble avec le produit 2, puis 3 et 4 ensembles et la même chose pour les deux derniers produits (5 et 6). À à ce moment là, on est arrivé à répondre sur la question qui concerne le nombre de boite minimale nécessaire pour stocker ainsi transporter les produits chimiques.

Conclusion

Dans ce chapitre nous présentons certains problèmes concrets que nous avons modélisés ainsi que sa résolution à l'aide des méthodes vu dans le chapitre précédent.

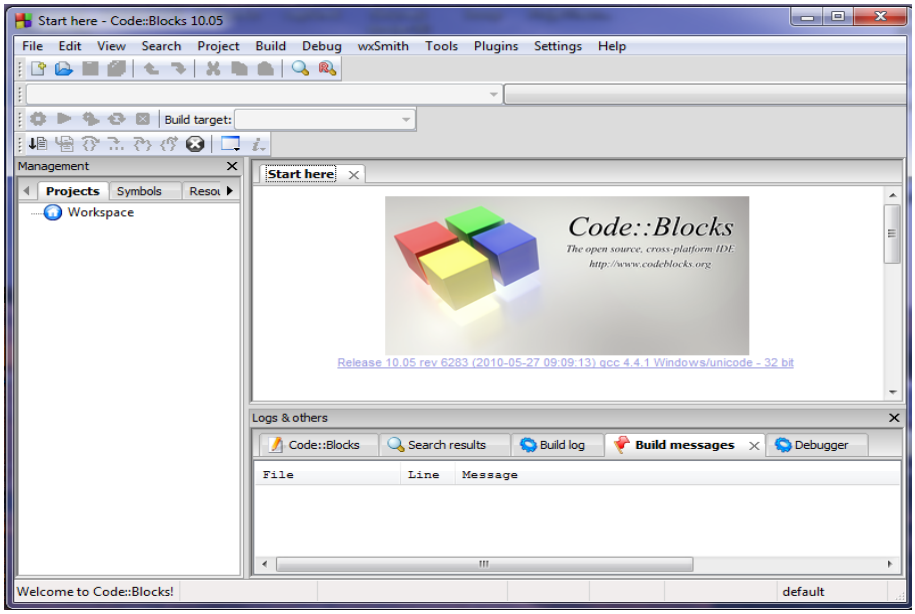
CHAPITRE 4

APPLICATION

Ce chapitre porte sur le retraitement des exemples vus dans le chapitre 3 en utilisant les applications programmé sous Code Block.

4.1 Introduction au Code Block

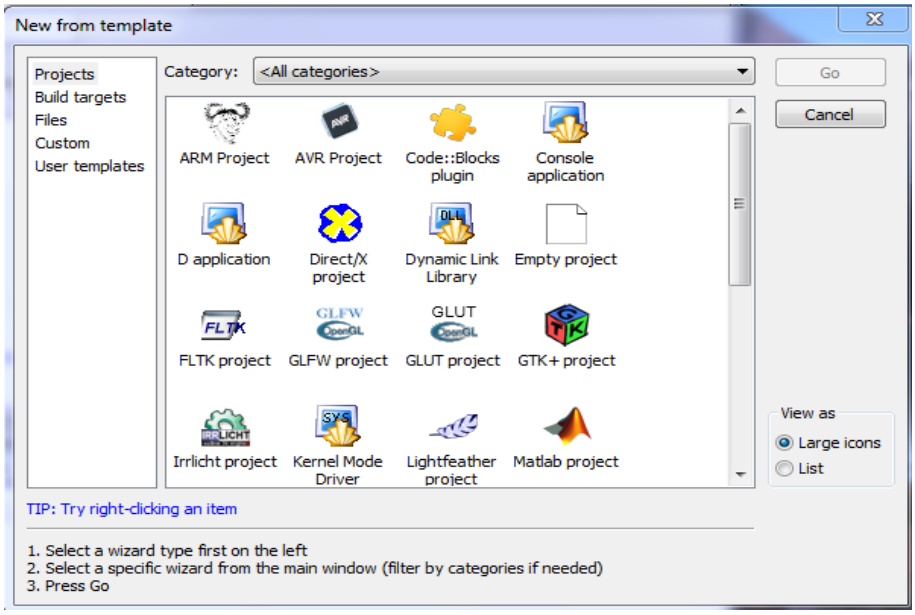
Code Blocks est un environnement de développement intégré libre et multiplateforme. Il est écrit en C++ grâce à la bibliothèque wxWidgets. Code : :Blocks est orienté C et C++, mais il supporte d'autres langages comme FORTRAN ou le D. Code Blocks est développé pour Linux, Windows et Mac OS X. Des utilisateurs indiquent avoir réussi à compiler le code source sous FreeBSD. L'illustration ci-dessous montre l'apparence de la fenêtre de l'interface utilisateur de Code-Blocks.



Dans CodeBlocks, les sources et les paramètres d'un processus de génération sont stockés dans un fichier projet $< name > .cbp$. Les sources en C/C++ et les fichiers d'entêtes correspondants (ou headers) sont les composants typiques d'un projet. La façon la plus simple de créer un projet est de passer par la commande "Fichier"→ "Projet" et de choisir un assistant. Vous pouvez alors ajouter des fichiers au projet via le menu de contexte 'Ajouter des fichiers' de la fenêtre de gestion. CodeBlocks gère les fichiers de projets en catégories qui dépendent de l'extension des fichiers. Les catégories suivantes sont prédéfinies :

- **Sources** contient les fichiers sources dont l'extension est $*.c; *.cpp$;.
- **ASM Sources** contient les fichiers sources dont l'extension est $*.s; *.S; *.ss; *.asm$.
- **Headers** contient, entre autres, les fichiers dont l'extension est $*.h$;. Ressources contient les fichiers pour paramétrer l'aspect des fenêtres des wxWidgets avec les extensions $*.res; *.xrc$;. Ces types de fichiers sont affichés dans l'onglet 'Ressources' de la fenêtre de Gestion.

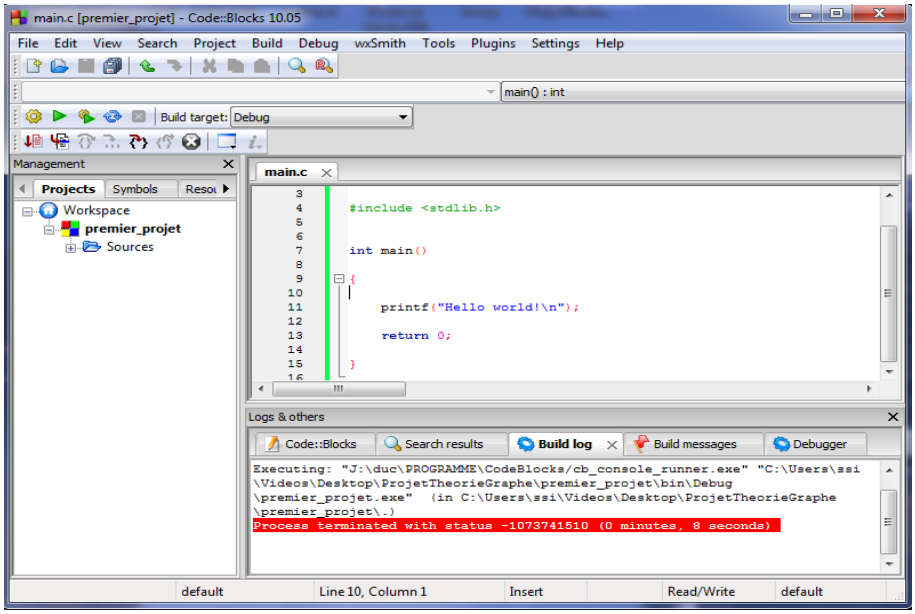
Pour créer un nouveau projet on click sur le menu File → New → Project... Une boîte de dialogue s'ouvre alors pour vous permettre de choisir le type de projet que vous souhaitez créer.



Code Blocks place automatiquement tous les fichiers du projet dans un répertoire qui porte le nom du projet.

Construire et executer un projet

Vous pouvez éditer le fichier main. En allant à gauche dans la fenêtre Management dans Sources → main.c. Ce fichier comporte la fonction main, fonction principale du programme.



4.2 Plus court chemin dans un réseau routier en Algérie

Le problème a traité est celui du chapitre 3 qui montré dans la Figure 4.1 :

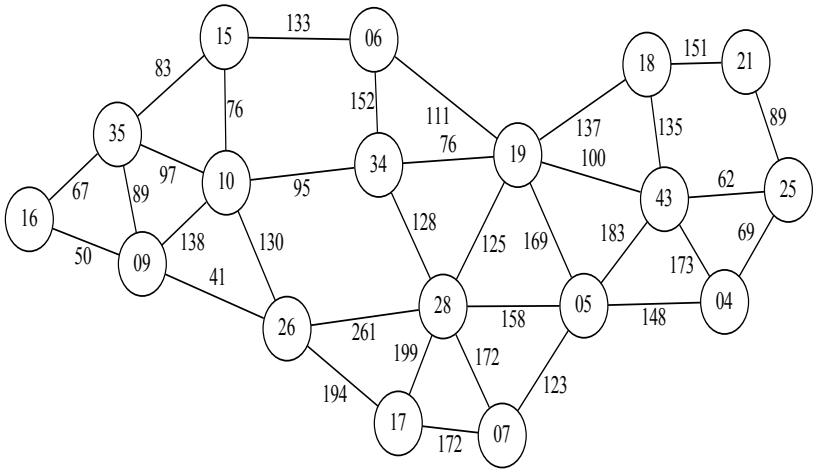


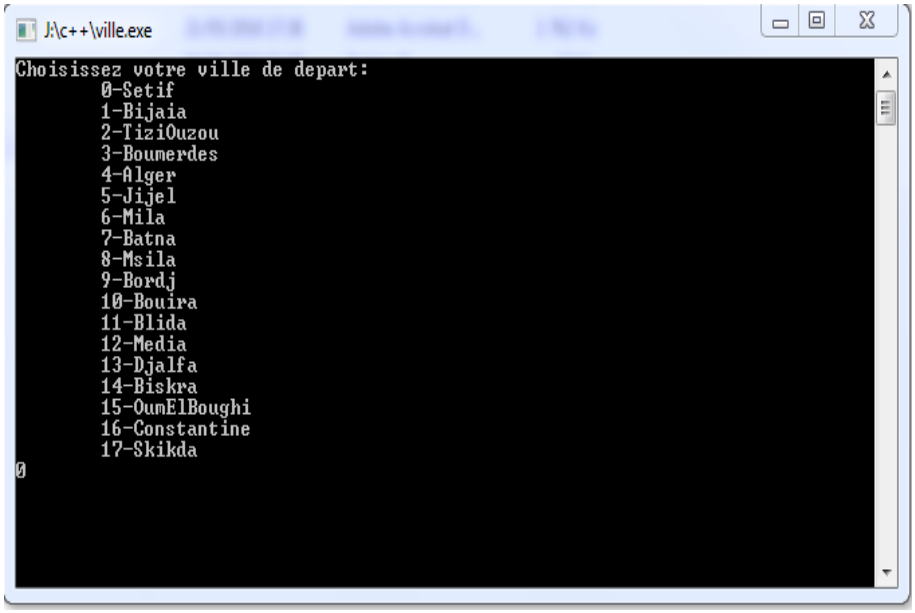
FIGURE 4.1 – Réseau routier

La matrice de capacité de graphe est illustrée dans le tableau de la Figure 4.2 :

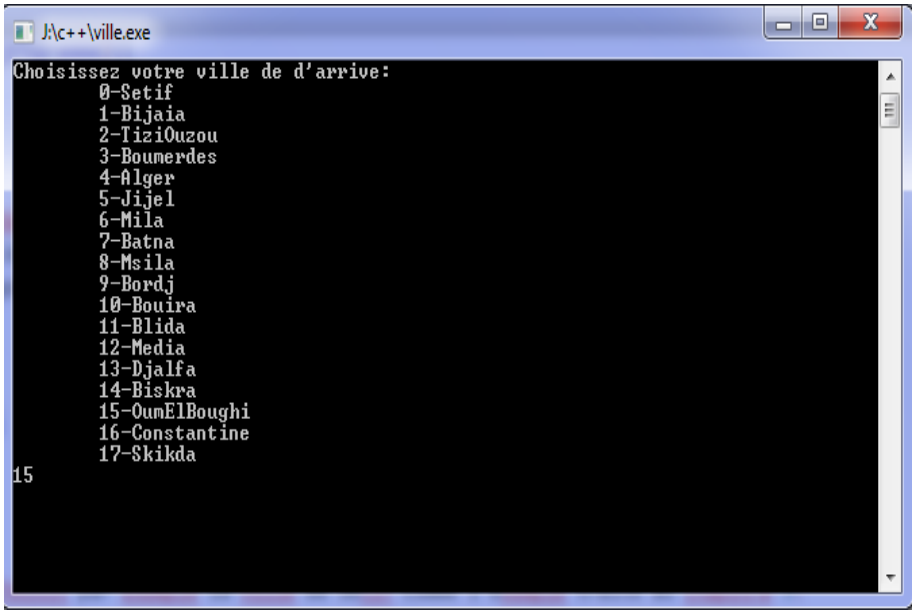
		19	06	15	35	16	18	43	05	28	34	10	09	26	17	07	04	25	21
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
19	0	*	111	-	-	-	137	100	169	125	76	-	-	-	-	-	-	-	-
06	1	111	*	133	-	-	-	-	-	-	152	-	-	-	-	-	-	-	-
15	2	-	133	*	83	-	-	-	-	-	-	76	-	-	-	-	-	-	-
35	3	-	-	83	*	67	-	-	-	-	-	97	89	-	-	-	-	-	-
16	4	-	-	-	67	*	-	-	-	-	-	-	50	-	-	-	-	-	-
18	5	137	-	-	-	-	*	135	-	-	-	-	-	-	-	-	-	-	151
43	6	100	-	-	-	-	135	*	183	-	-	-	-	-	-	-	173	62	-
05	7	169	-	-	-	-	-	183	*	158	-	-	-	-	-	123	148	-	-
28	8	125	-	-	-	-	-	-	158	*	128	-	-	261	199	172	-	-	-
34	9	76	152	-	-	-	-	-	-	128	*	95	-	-	-	-	-	-	-
10	10	-	-	76	97	-	-	-	-	-	95	*	138	130	-	-	-	-	-
09	11	-	-	-	89	50	-	-	-	-	-	138	*	41	-	-	-	-	-
26	12	-	-	-	-	-	-	-	-	261	-	130	41	*	194	-	-	-	-
17	13	-	-	-	-	-	-	-	-	199	-	-	-	194	*	172	-	-	-
07	14	-	-	-	-	-	-	-	123	172	-	-	-	-	172	*	-	-	-
04	15	-	-	-	-	-	-	173	148	-	-	-	-	-	-	-	*	69	-
25	16	-	-	-	-	-	-	62	-	-	-	-	-	-	-	-	69	*	89
21	17	-	-	-	-	-	151	-	-	-	-	-	-	-	-	-	-	89	*

FIGURE 4.2 – Matrice de capacité du réseau routier

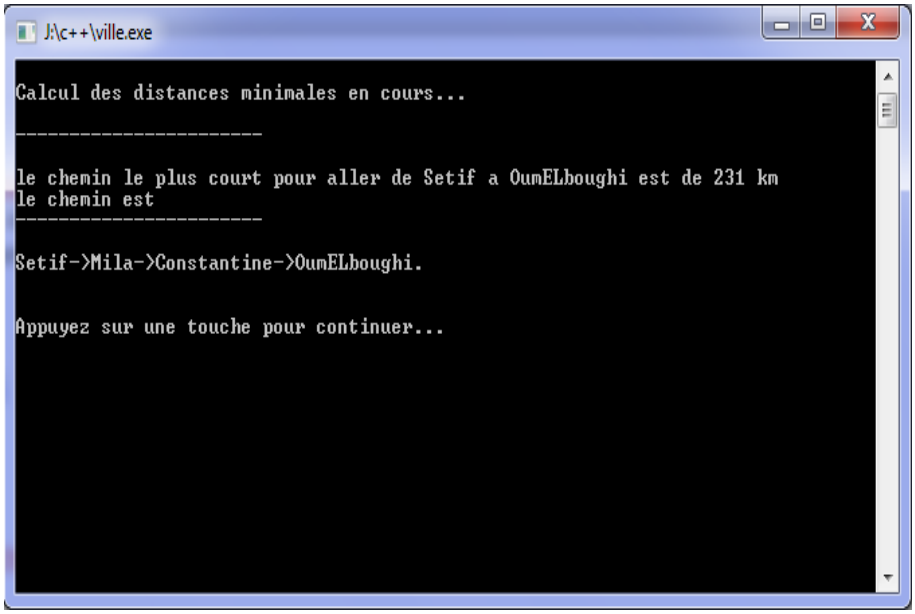
L'application est programmée en suivant les étapes de l'algorithme Dijkistra.



L'application demande votre ville de départ choisit par exemple la ville de Sétif comme l'exemple traité au chapitre 3.

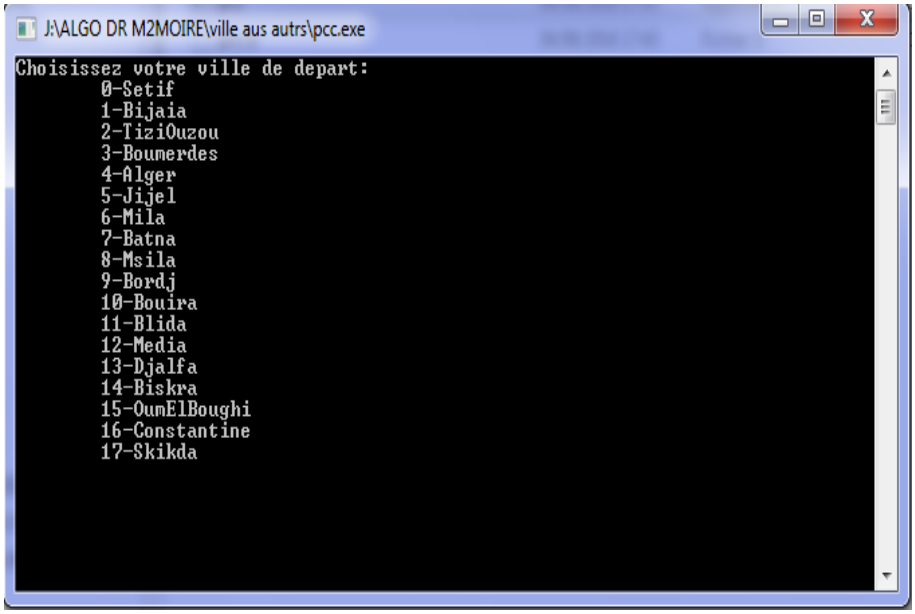


La ville d'arrivé est Oum El Boughi.

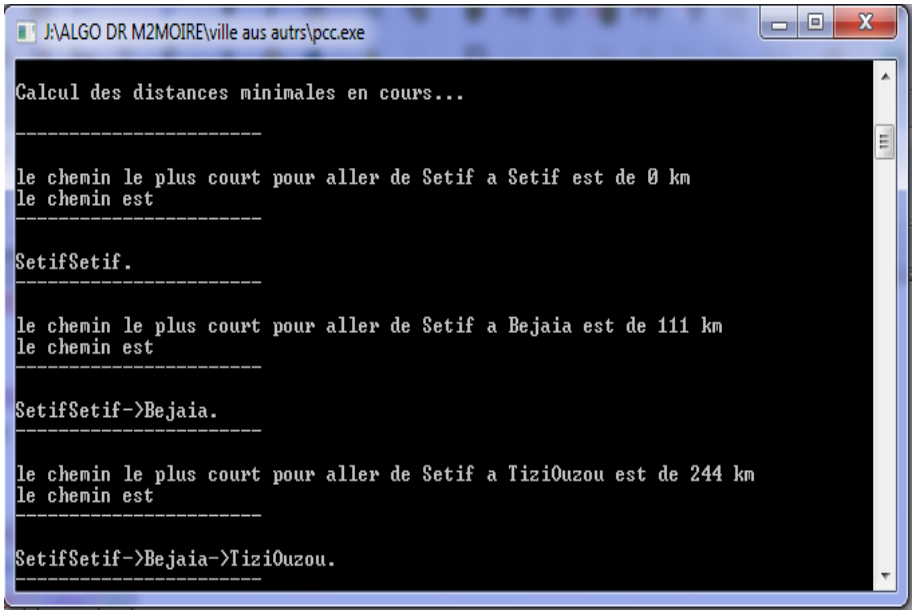


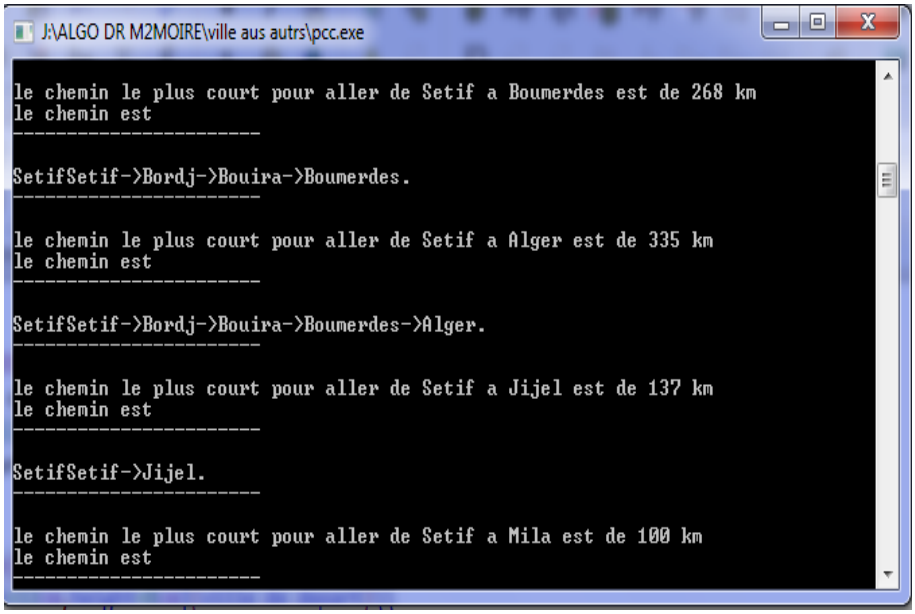
Le plus court chemin est alors : Sétif→ Mila→ Constantine→ Oum El Boughi.

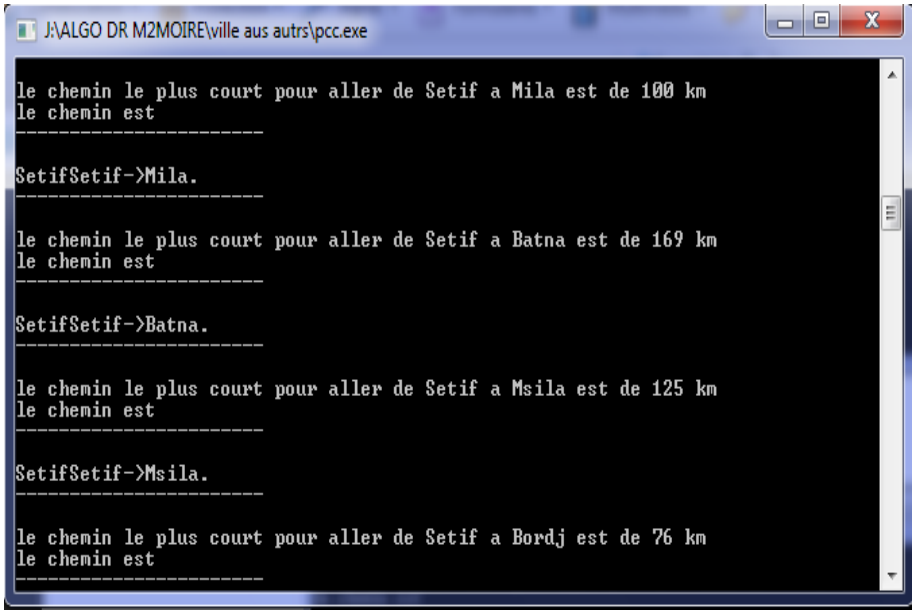
Les plus courts chemins d'une ville à toute les autres villes

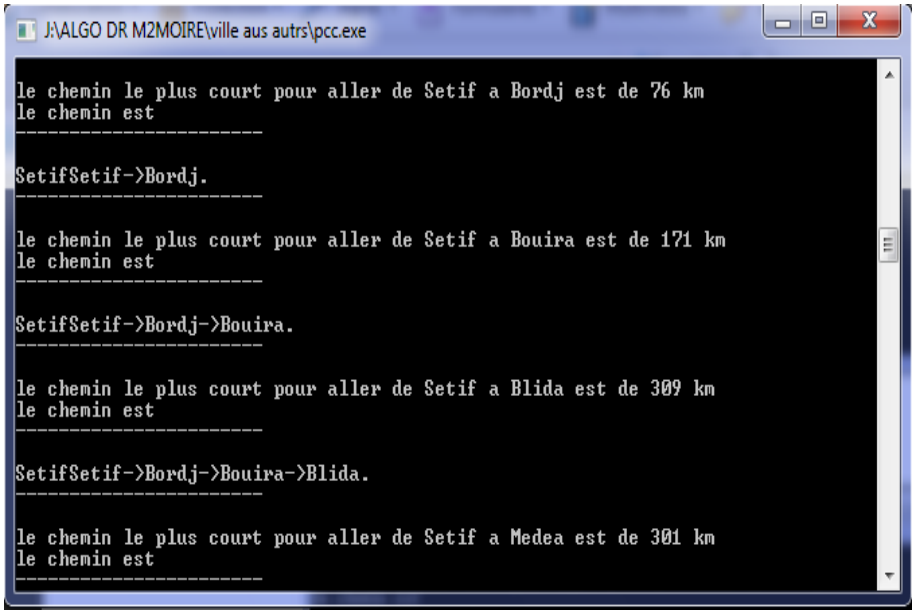


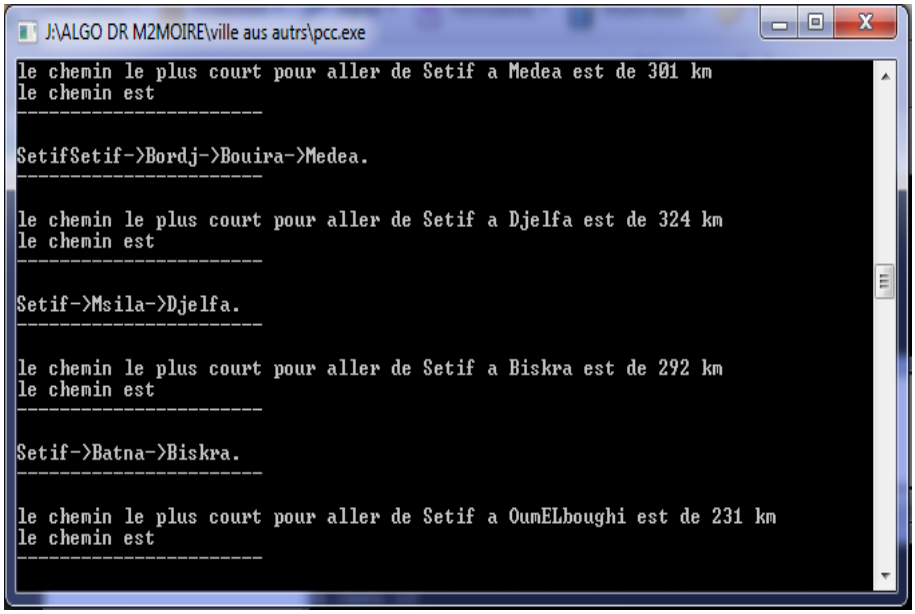
En choisissant la ville de Sétif comme ville de départ, les plus courts chemins vont être illustrés dans les Figures de l'application suivante :

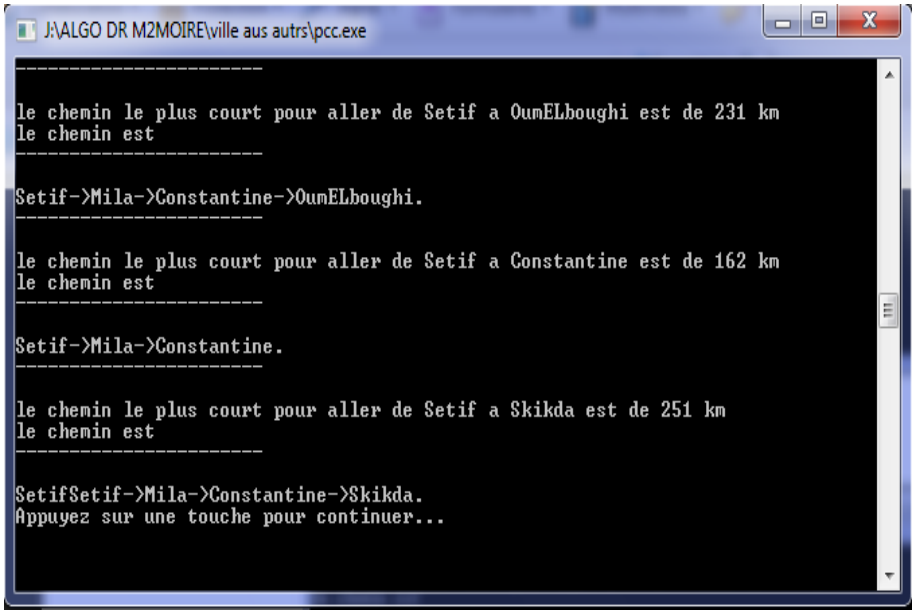

J:\ALGO DR M2MOIRE\ville aus autrs\pcc.exe
Calcul des distances minimales en cours...
le chemin le plus court pour aller de Setif a Setif est de 0 km
le chemin est
SetifSetif.
le chemin le plus court pour aller de Setif a Bejaia est de 111 km
le chemin est
SetifSetif->Bejaia.
le chemin le plus court pour aller de Setif a TiziOuzou est de 244 km
le chemin est
SetifSetif->Bejaia->TiziOuzou.


J:\ALGO DR M2MOIRE\ville aus autrs\pcc.exe
le chemin le plus court pour aller de Setif a Boumerdes est de 268 km
le chemin est
SetifSetif->Bordj->Bouira->Boumerdes.
le chemin le plus court pour aller de Setif a Alger est de 335 km
le chemin est
SetifSetif->Bordj->Bouira->Boumerdes->Alger.
le chemin le plus court pour aller de Setif a Jijel est de 137 km
le chemin est
SetifSetif->Jijel.
le chemin le plus court pour aller de Setif a Mila est de 100 km
le chemin est


J:\ALGO DR M2MOIRE\ville aus autrs\pcc.exe
le chemin le plus court pour aller de Setif a Mila est de 100 km
le chemin est
SetifSetif->Mila.
le chemin le plus court pour aller de Setif a Batna est de 169 km
le chemin est
SetifSetif->Batna.
le chemin le plus court pour aller de Setif a Msila est de 125 km
le chemin est
SetifSetif->Msila.
le chemin le plus court pour aller de Setif a Bordj est de 76 km
le chemin est

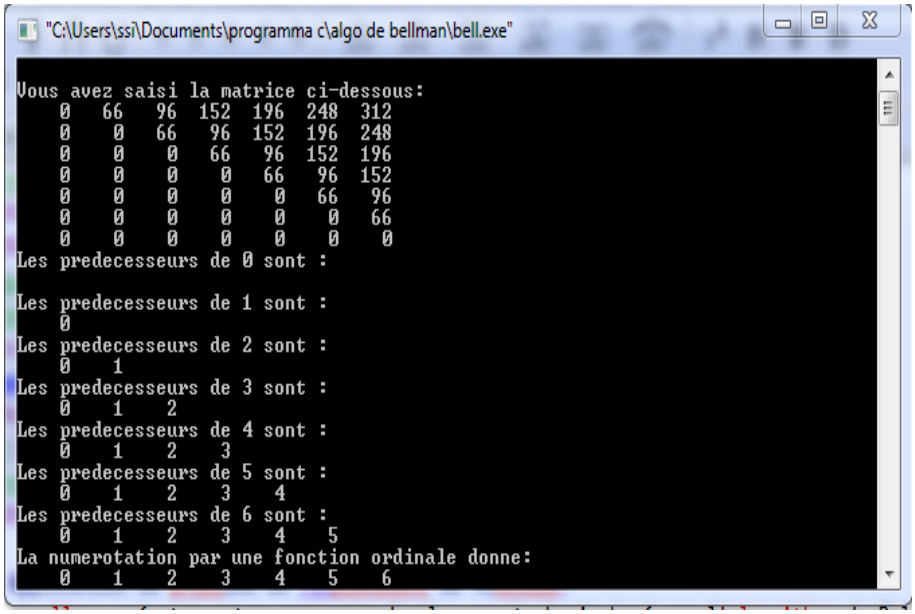

J:\ALGO DR M2MOIRE\ville aus autrs\pcc.exe
le chemin le plus court pour aller de Setif a Bordj est de 76 km
le chemin est
SetifSetif->Bordj.
le chemin le plus court pour aller de Setif a Bouira est de 171 km
le chemin est
SetifSetif->Bordj->Bouira.
le chemin le plus court pour aller de Setif a Blida est de 309 km
le chemin est
SetifSetif->Bordj->Bouira->Blida.
le chemin le plus court pour aller de Setif a Medea est de 301 km
le chemin est

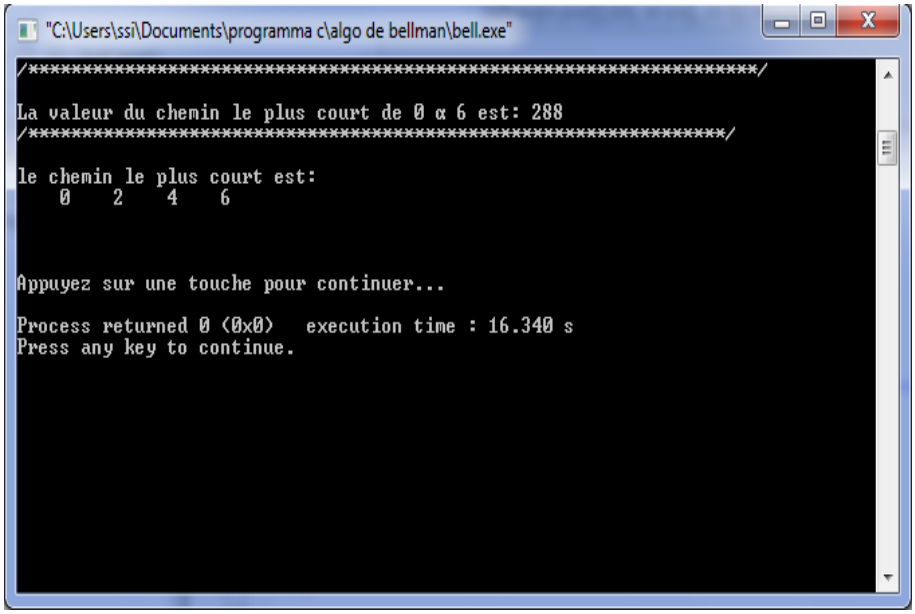

J:\ALGO DR M2MOIRE\ville aus autrs\pcc.exe
le chemin le plus court pour aller de Setif a Medea est de 301 km
le chemin est
SetifSetif->Bordj->Bouira->Medea.
le chemin le plus court pour aller de Setif a Djelfa est de 324 km
le chemin est
Setif->Msila->Djelfa.
le chemin le plus court pour aller de Setif a Biskra est de 292 km
le chemin est
Setif->Batna->Biskra.
le chemin le plus court pour aller de Setif a OumELboughi est de 231 km
le chemin est


J:\ALGO DR M2MOIRE\ville aus autrs\pcc.exe
le chemin le plus court pour aller de Setif a OumELboughi est de 231 km
le chemin est
Setif->Mila->Constantine->OumELboughi.
le chemin le plus court pour aller de Setif a Constantine est de 162 km
le chemin est
Setif->Mila->Constantine.
le chemin le plus court pour aller de Setif a Skikda est de 251 km
le chemin est
SetifSetif->Mila->Constantine->Skikda.
Appuyez sur une touche pour continuer...

4.3 Application au problème de remplacement de véhicule

Nous allons exécuter notre programme de plus court chemin basé sur l'algorithme de Bellman-Ford sur ce problème. Les résultats sont données dans les Figures suivantes :





Le plus court chemins de source 0 au puit 6 est $0 \rightarrow 2 \rightarrow 4 \rightarrow 6$ et de valeur 2880 000 DA, comme l'indique le résultat précédent.

4.4 Application au problème de canalisation d'eau dans 3 villes

Le problème de canalisation d'eau de chapitre 3 a donnée naissance au graphe de la Figure 4.3 :

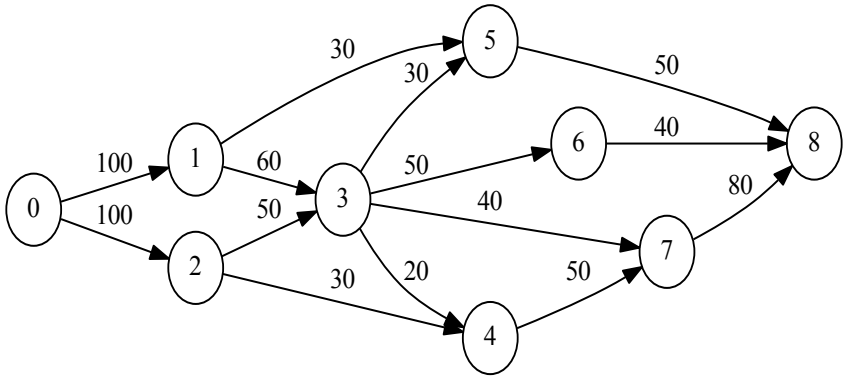


FIGURE 4.3 – Problème de canalisation d'eau dans 3 villes

La matrice de capacité de graphe est les résultats sont donnés dans la Figure 4.4 :

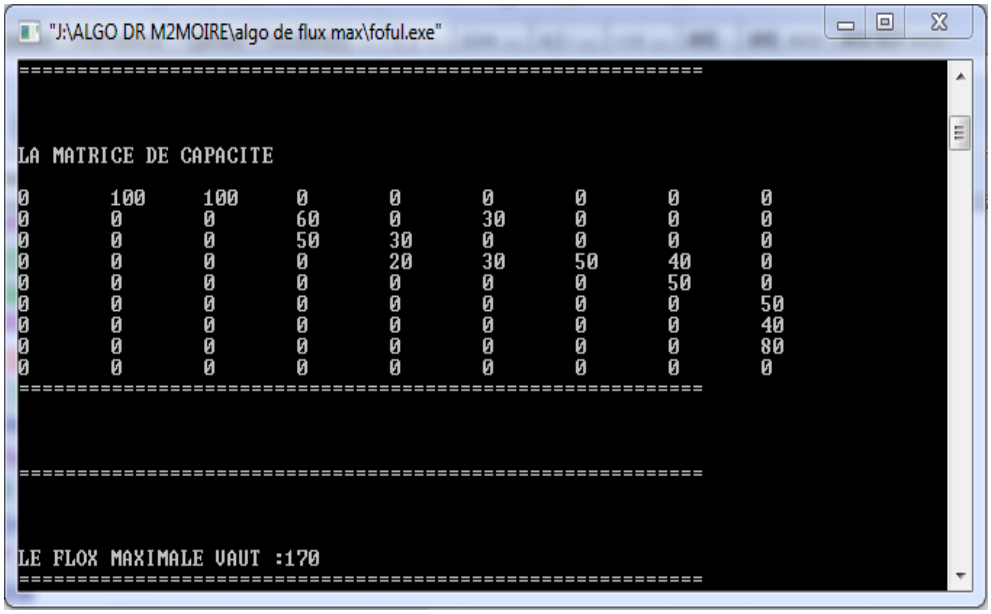


FIGURE 4.4 – Résultats relatifs au problème de canalisation d'eau

4.5 Application au problème d'ordonnancement des tâches d'un projet

Le problème traité au chapitre 3 dont la représentation graphique donné par la Figure 4.5 :

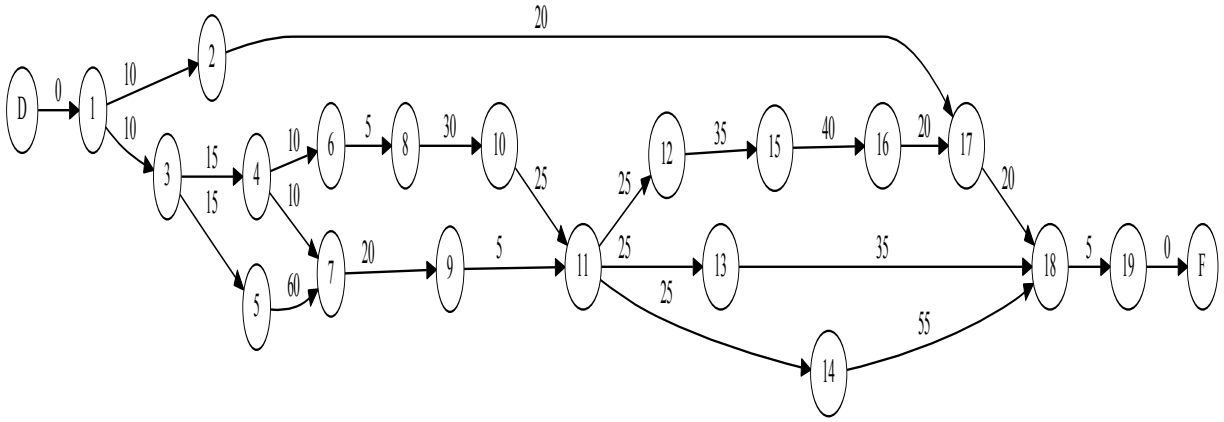
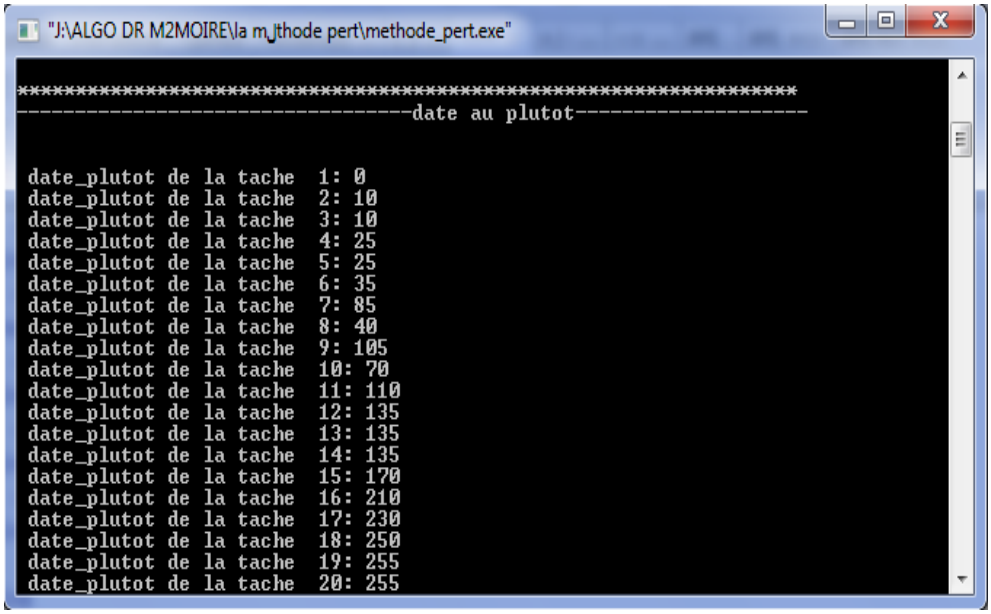


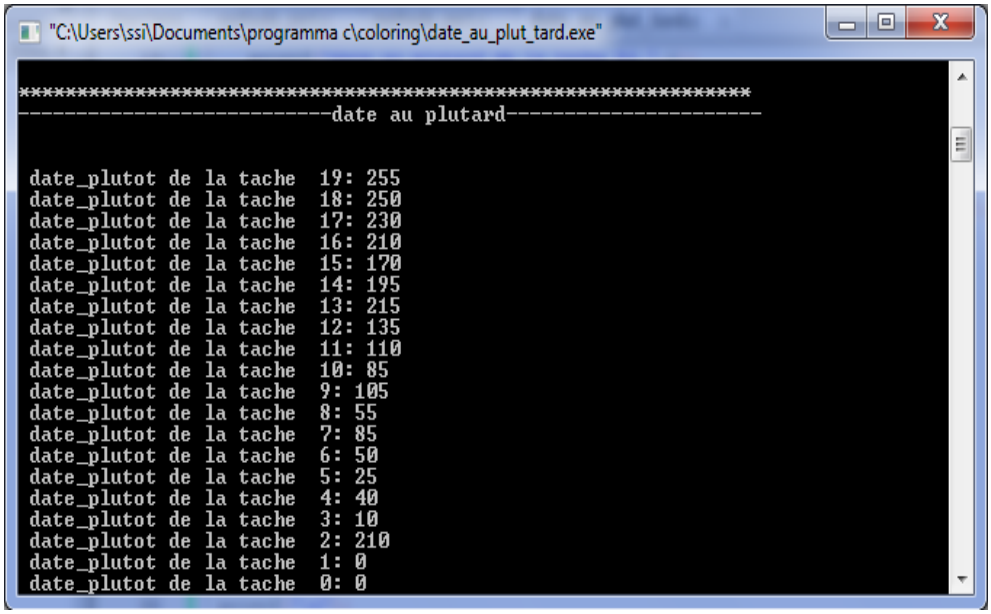
FIGURE 4.5 – L'ordonnancement des tâches d'un projet

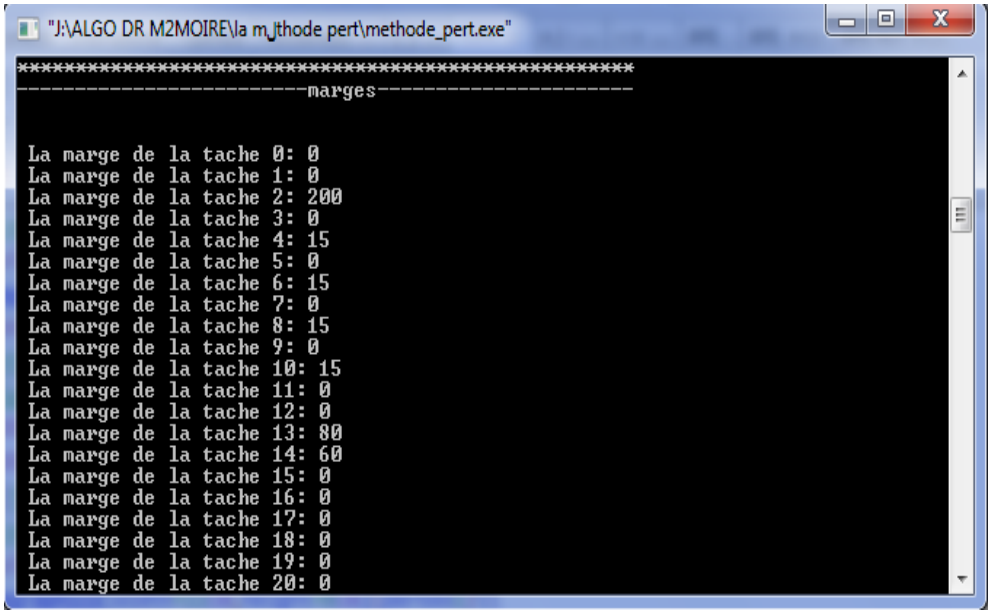
Donc le problème est composé de 20 tâches et de la matrice de capacité suivante :

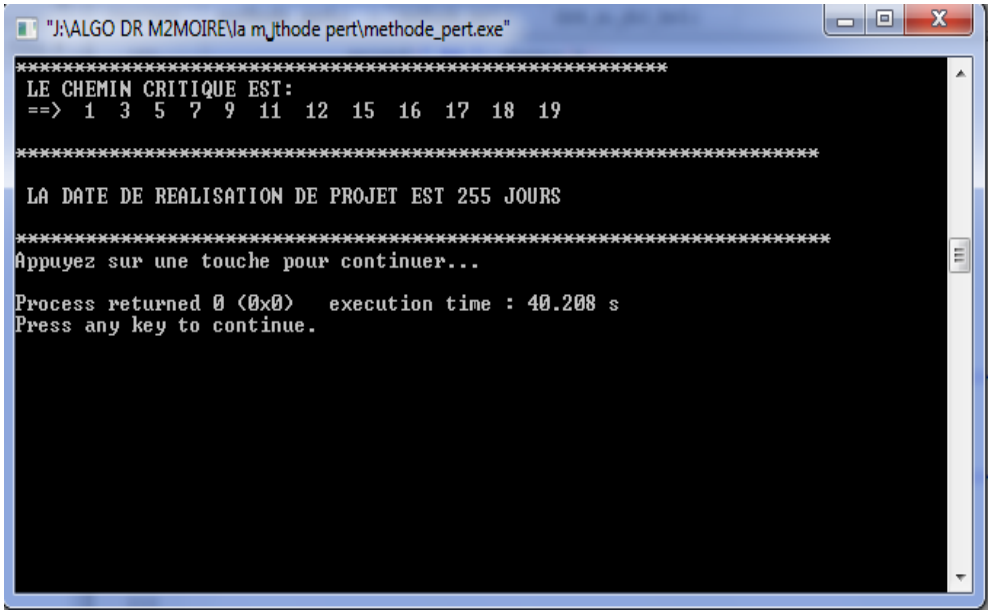
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	*	10	10	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2	-	*	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	20	-1	-1
3	-	-	*	15	15	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
4	-	-	-	*	-1	10	10	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
5	-	-	-	-	*	-1	60	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
6	-	-	-	-	-	*	-1	5	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
7	-	-	-	-	-	-	*	-1	20	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
8	-	-	-	-	-	-	-	*	-1	30	-1	-1	-1	-1	-1	-1	-1	-1	-1
9	-	-	-	-	-	-	-	-	*	-1	5	-1	-1	-1	-1	-1	-1	-1	-1
10	-	-	-	-	-	-	-	-	-	*	25	-1	-1	-1	-1	-1	-1	-1	-1
11	-	-	-	-	-	-	-	-	-	-	*	25	25	25	-1	-1	-1	-1	-1
12	-	-	-	-	-	-	-	-	-	-	-	*	-1	-1	35	-1	-1	-1	-1
13	-	-	-	-	-	-	-	-	-	-	-	-	*	-1	-1	-1	-1	35	-1
14	-	-	-	-	-	-	-	-	-	-	-	-	-	*	-1	-1	-1	55	-1
15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*	40	-1	-1	-1
16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*	20	-1	-1
17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*	20	-1
18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*	5

L'application à ce modèle nous donne les résultats suivantes :


"J:\ALGO DR M2MOIRE\la m_thode pert\methode_pert.exe"
****
----------date au plutot----------
date_plutot de la tache 1: 0
date_plutot de la tache 2: 10
date_plutot de la tache 3: 10
date_plutot de la tache 4: 25
date_plutot de la tache 5: 25
date_plutot de la tache 6: 35
date_plutot de la tache 7: 85
date_plutot de la tache 8: 40
date_plutot de la tache 9: 105
date_plutot de la tache 10: 70
date_plutot de la tache 11: 110
date_plutot de la tache 12: 135
date_plutot de la tache 13: 135
date_plutot de la tache 14: 135
date_plutot de la tache 15: 170
date_plutot de la tache 16: 210
date_plutot de la tache 17: 230
date_plutot de la tache 18: 250
date_plutot de la tache 19: 255
date_plutot de la tache 20: 255

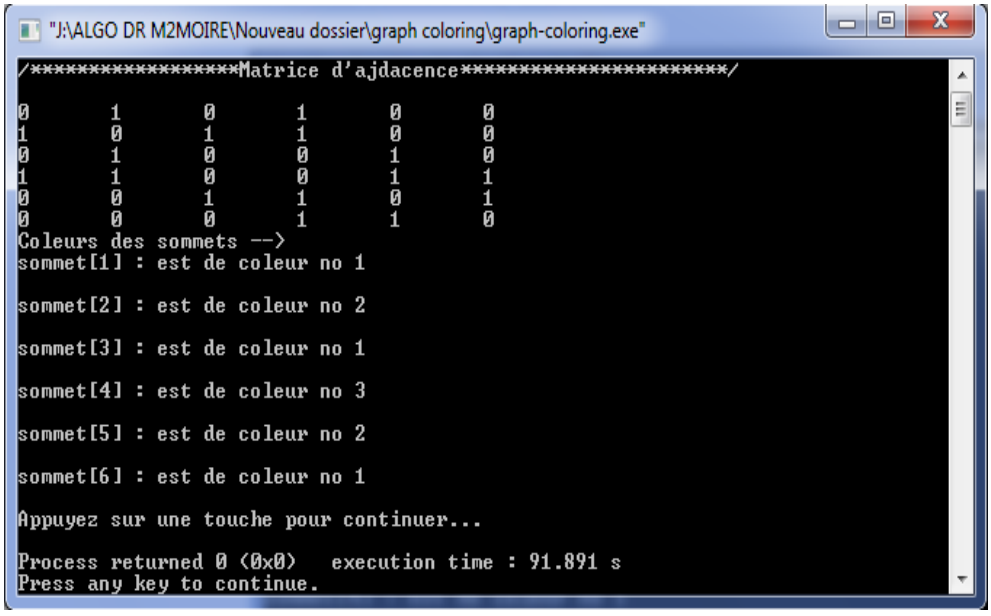

"C:\Users\ssi\Documents\programma c\coloring\date_au_plut_tard.exe"
****
----------date au plutard----------
date_plutot de la tache 19: 255
date_plutot de la tache 18: 250
date_plutot de la tache 17: 230
date_plutot de la tache 16: 210
date_plutot de la tache 15: 170
date_plutot de la tache 14: 195
date_plutot de la tache 13: 215
date_plutot de la tache 12: 135
date_plutot de la tache 11: 110
date_plutot de la tache 10: 85
date_plutot de la tache 9: 105
date_plutot de la tache 8: 55
date_plutot de la tache 7: 85
date_plutot de la tache 6: 50
date_plutot de la tache 5: 25
date_plutot de la tache 4: 40
date_plutot de la tache 3: 10
date_plutot de la tache 2: 210
date_plutot de la tache 1: 0
date_plutot de la tache 0: 0


"J:\ALGO DR M2MOIRE\la m_thode pert\methode_pert.exe"
****
----------marges----------
La marge de la tache 0: 0
La marge de la tache 1: 0
La marge de la tache 2: 200
La marge de la tache 3: 0
La marge de la tache 4: 15
La marge de la tache 5: 0
La marge de la tache 6: 15
La marge de la tache 7: 0
La marge de la tache 8: 15
La marge de la tache 9: 0
La marge de la tache 10: 15
La marge de la tache 11: 0
La marge de la tache 12: 0
La marge de la tache 13: 80
La marge de la tache 14: 60
La marge de la tache 15: 0
La marge de la tache 16: 0
La marge de la tache 17: 0
La marge de la tache 18: 0
La marge de la tache 19: 0
La marge de la tache 20: 0


"J:\ALGO DR M2MOIRE\la m_thode pert\methode_pert.exe"
****
LE CHEMIN CRITIQUE EST:
==> 1 3 5 7 9 11 12 15 16 17 18 19
****
LA DATE DE REALISATION DE PROJET EST 255 JOURS
****
Appuyez sur une touche pour continuer...
Process returned 0 (0x0) execution time : 40.208 s
Press any key to continue.

4.6 Application sur le problème de stockage et de transport des produits chimiques :

Pour la résolution de ce problème, on lui applique la méthode de (Back-Tracking) coloration, dont on introduit la matrice d'ajdacence puis le programme nous donne les différents coloration des sommets comme il sont indiquer dans la Figure suivant :



La solution optimale de problème est 3 boite dont on arrange les objet (1 ;2) ensemble, (3 ;4)
ensemble, (5 ;6) ensemble, ce qui interprète la couleur ayant un nombre maximale d'arête de même couleur qui la couleur 3, les arête sont e_1,e_3 et e_3, telle que e_1 relié les sommets (1, 2),e_3 relié les sommets (3, 4) et e_6 relié les sommets (5, 6).

Conclusion

Dans ce chapitre, nous intéressons au partie réalisation ou nous avons donné les résultats obtenus relatif aux problèmes modélisant précédemment sous langage C.

CONCLUSION GÉNÉRALE

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

L'importance de la théorie des graphes vient aussi 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 a donné au lecteur un certain nombre d'outil (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 utilise l'optimisation en théorie des graphes, pour cela on a été intéressé pour la résolution de cinq problèmes.

Le premier consiste à résoudre un problème de réseau routier en Algérie, ou 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 prolème a été résoulu à l'aide de l'algorithme de Dijkstra.

Le deuxième problème sert à résoudre le problème de remplacement de véhicules dans une agence de location de voitures, dont on cherche la meilleure politique de remplacement des véhicules dans son parc automobile qu'est modéliser sous forme de problème de plus court chemin en utilisant pour sa résolution l'algorithme de Bellman-Ford.

Le troixième problème conserne le problème de canalisation d'eau dans 3 villes là on cherche à determiner s'il est possible de satisfaire à travers ce réseau la demande d'eau de trois villes. Ce problème est modéliser sous forme d'un problème de flot maximal dont on a utilisé l'algorithme de Ford-Fulkersen pour sa résolution.

Le quatrième traite l'ordonnancement des tâches d'un projet, résoulu avec la méthode pert.

Dans le quatrième, on a appliqué l'algorithme de coloration "Powell" pour trouvé le nombre minimale de boites necéssaires pour stocker ainsi transrpoter des produits chimiques, susceptibles d'avoir des intéractions entre-eux.

Finalement, notre travail consiste à étudier et implémenter de façon plus approfondie les
algorithmes les plus adapté sous language C, 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 Béjaia*, (2010/2011).

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

[3] Franck Butelle. Contribution à l'algorithme distribuée de controle : Arbre couvrants avec et sans contrantes. *Université de Paris*, Mars.

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

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

[6] C.Simon F. Madelaire. Cour sur les graphes. *IUT Informatique 2 G1*, (2007/2008).

[7] Tableau kilometrage algerie. Recherche google. *Site internet*, (2016).

[8] Pierre Lopez. Cour des graphes. *LAAS-CNRS*, Novemre.

[9] M^{me}.*Amarouche*. Cour théorie des graphes. *Université A.mira Béjaia*, (2011/2012).

[10] 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).

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

[12] M^r. Taouinat. *Cour Théorie des graphes avancée.* Université A.mira Béjaia, 2012/2013.

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-Ford, Kruskal...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.

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 ie the Bellman- Ford algorithm, Kruskal ... etc for solving concrete problems , the limits of these algorithms used in the context of optimization.

Keywords : Graph, Optimisation, Path, Coloration.

Annexe

GrapheViz

Le dessin des graphes sur ce mémoire est fait à l'aide de logiciel Graphviz. Ce dernier est un logiciel de visualisation de graphes. Il permet de représenter des données structurées sous la forme de diagramme de graphes. Graphviz est un logiciel de diffusion libre disponible sur le site *http* : *//graphviz.org/*
Une documentation détaillée sur le logiciel ainsi que sur son utilisation en tant que bibliothèque sont données en détails sur le site *http* : *//graphviz.org/* .
L'illustration ci-dessous montre l'apparence de la fenêtre de l'interface utilisateur de Graphviz ainsi qu'un exemple.