République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A. Mira de Béjaia

Faculté des Science Exactes

Département de Recherche Opérationnelle

Mémoire de fin de cycle

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

Spécialité : Modélisation Mathématique et évaluation de Performance des Réseaux

Théme

Quelques techniques d'optimisation dans les réseaux

Présenté par :

M^{lle} Benmaamar Malika

M^{me} Irid Souad

Proposé et encadré par :

M^{r} KABYL K.

Devant le jury :

Président : m^{r} $S.Taouinet$

Examinateur 1 : M^{me} $L.Younsi$

Examinateur 2 : M^{r} $M.Soufit$

$***$ Promotion 2017/2018 $***$

REMERCIEMENT

Nous remercions Dieu en premier lieu, qu'il soit loué pour nous avoir donné la force et la patience nécessaires pour accomplir ce travail.

Nous tenons à remercier très sincèrement Mr Kabyl Kamal maître de conférence à l'université A.Mira de Béjaia, notre encadreur. Ce fut un grand plaisir de travailler avec lui, durant la préparation de notre mémoire. On a beaucoup appris de son expérience et de ses conseils. On lui serai reconnaissant de confiance qu'il nous a témoigné.

Nous remercions profondément les membre du jury de nous avoir fait l'honneur de juger ce travail.

A toute notre promotion Master 2018 .

N'oublions pas tous nos chers amis.

DÉDICACE

♡ *Je dèdie ce modeste travail à*

♡*Mes chers parents,c'est le moment plus que jamais de vous remercier pour tout ce que vous avez fait.*

♡*Et ma famille*

♡*mes chères copine Hanan et Soumia*

♡*ma binôme Souad*

♡ *a toutes les personnes qui étaient a côté de moi*

♡*Malika*

DÉDICACE

♡ *Je dèdie ce modeste travail à*

♡ *Mes parents : Youcef et Hannaoua*
Aucun hommage ne pourrait être à la hauteur de l'amour Dont ils ne cessent de me combler. Que dieu leurs procure bonne santé et longue vie.

♡*A celui que j'aime beaucoup et qui m'a soutenue tout au long de ce projet : mon mari Naim,*

♡*Et bien sur A mes frères Tahar, Farid et Khelaf et a mes sœurs Karima et Bahia*

♡ *et a mes belles-sœurs Lynda et Marie sans oublié ma grand-mère et mes beaux-parents.*

♡*A toute ma famille, et mes amis, A ma camarade Malika et toute la famille Irid.*

♡ *Et à tous ceux qui ont contribué de près ou de loin pour que ce projet soit possible, je vous dis merci.*

♡*Souad*

RÉSUMÉ

L'étude présentée dans ce document porte essentiellement sur les approches d'optimisation dans les réseaux, où on a mit en œuvre des algorithmes de recherche d'un plus court chemin sur un graphe pondéré connexe. Comme on a présenté une méthode pour planifier la réalisation d'un projet afin de déterminer sa durée minimale, tout en respectant les contraintes d'antériorité, et aussi la résolution d'un problème de télécommunication. Pour cela une application sur un réseau routier algérien a été proposée suivie d'un programme sous Dev-c++.

Mots clé : réseau, approches d'optimisation

Abstract

The study presented in this document focuses on optimization approaches in network. Where we have implemented search algorithms of a shorter path on a connected weighted graph. We have submitted a method to plan the realization of a project to determine it minimum duration, while respecting the constraints of anteriority, and also resolution of a telecommunication problem. For this, an application on an Algerian road network was proposed followed by a program under Dev-c++.

Keywords : Network, optimization approaches.

TABLE DES MATIÈRES

TABLE DES FIGURES

INTRODUCTION GÉNÉRALE

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

L'histoire de la théorie des graphes débuterait avec les travaux d'Euler au 18^e siècle et trouve son origine dans l'étude de certains problèmes, tels que celui des ponts de Königsberg, la marche du cavalier sur l'échiquier ou le problème du coloriage de cartes et du plus court trajet entre deux points.

En mathématiques, on retrouve les graphes dans la combinatoire, la théorie des ensembles, l'algèbre linéaire, la théorie des polyèdres, la théorie des jeux, l'algorithmique, les probabilités. . .

Les derniers travaux en théorie des graphes sont souvent effectués par des informaticiens, du fait de l'importance que revêt l'aspect algorithmique. Dans ce qui suit, nous allons donner un exemple caractéristique pour chacun des chapitres de ce document.

Le but de notre travail est la résolution d'un problème concret tout en utilisant l'optimisation dans les réseaux et pour cela on a réparti notre travail en trois chapitres :

• Dans le premier chapitre on a donné les définitions et les concepts de base de la théorie des graphes.

• Dans le deuxième chapitre on s'est intéressé aux réseaux et aux approches d'optimisation et leurs applications sur des problèmes modélisés.

• Dans le dernier chapitre nous avons abordé quelques problèmes concrets qui résument le but de notre travail à savoir les approches d'optimisations dans les réseaux programmés sous le langage Dev-c++.

CHAPITRE 1

GÉNÉRALITÉS SUR LA THÉORIE DES GRAPHES

Ce chapitre a pour but de représenter les différentes notions relatives aux graphes, qui seront utilisées dans les chapitres qui suivent.

1.1 Définitions

1.1.1 Graphe

De façon plus formelle, un graphe est défini par un couple $G = (X, E)$ tel que
- X est un ensemble fini de sommets,
- E est un sous-ensemble de couples de sommets $\{x_i, x_j\} \in X^2$.

Un graphe peut être orienté ou non-orienté :

Graphe non orienté

Les couples $(x_i, x_j) \in E$ ne sont pas orientés, c'est à dire que (x_i, x_j) est équivalent à (x_j, x_i). Une paire $\{x_i, x_j\}$ est appelée une arête et représentée graphiquement par $x_i - x_j$.

La figure 1.2 représente le graphe non orienté $G = (X; E)$ avec $X = \{1, 2, 3, 4, 5, 6\}$ et $E = \{\{1, 2\}, \{1, 5\}, \{5, 2\}, \{3, 6\}\}$.

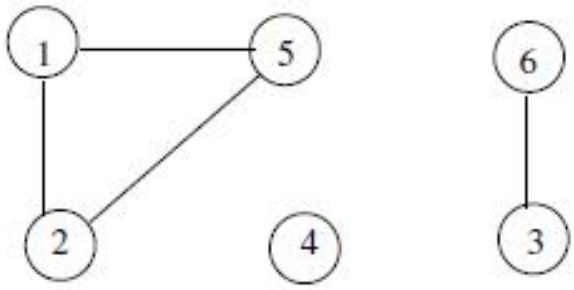


FIGURE 1.1 – Graphe non orienté

Graphe orienté

Les couples $(x_i, x_j) \in E$ sont orientés, c'est à dire que (x_i, x_j) est un couple ordonné, où x_i est le sommet initial, et x_j le sommet terminal. Un couple (x_i, x_j) est appelé un arc, et est représenté graphiquement par $x_i \to x_j$.
La figure suivante 1.1 représente un graphe orienté $G = (X, E)$ avec $X = \{1, 2, 3, 4, 5, 6\}$ et $E = \{(1, 2), (2, 4), (2, 5), (4, 1), (4, 4), (4, 5), (5, 4), (6, 3)\}$.

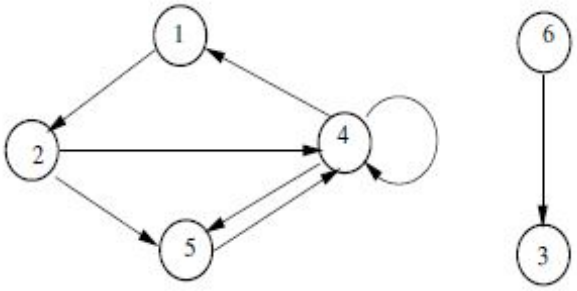


FIGURE 1.2 – Graphe orienté

Successeurs, prédécesseurs d'un sommet

a) L'ensemble des successeurs d'un sommet x_i est l'ensemble de tous les sommets ayant des arcs sortants du sommet x_i noté par $\Gamma^+(x_i)$ tel que :

$$\Gamma^+(x_i) = \{x_j \in X/(x_i, x_j) \in E\} \ ,$$

b) L'ensemble des prédécesseurs d'un sommet x_i est l'ensemble de tous les sommets ayant des arcs entrants du 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 prédécesseur ou x est un successeur qu'on note $\Gamma(x)$ avec : $\Gamma(x) = \Gamma^-(x) \cup \Gamma^+(x)$.

Dans le graphe de la figure 1.3 on a :

$\Gamma^+(4) = \{3\}$ $\Gamma^-(4) = \{1, 2\}$

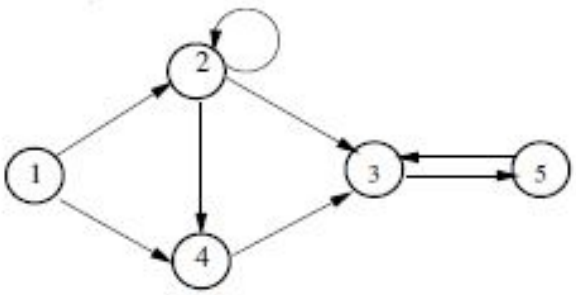


FIGURE 1.3 – Successeurs, prédécesseurs d'un sommet

Boucle

Une boucle est une arête d'un graphe ayant pour extrémités le même sommet.

Degré d'un sommet

On le note $d_G(x)$ qui est le nombre d'arêtes incidentes à x.

On définit $\delta(G) = Min_{x \in X} d_G(x)$ et $\Delta(G) = Max_{x \in X} d_G(x)$.

• $\delta(G)$: Le degré minimum de G

• $\Delta(G)$: Le degré maximum de G

Dans le cas d'un graphe orienté $G = (X, E)$,

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

• Le demi-degré intérieur d'un sommet x : est égale au nombre d'arcs ayant x comme extrémité terminal (c-à-d les arcs entrant en x) on le note $d_G^-(x)$ avec :

$d_G^-(x) = |\{e \in E / T(e) = x\}|$.

• Le degré d'un sommet x : c'est le nombre d'arcs ayant x comme extrémité initiale et terminal, on le note $d_G(x)$ avec : $d_G(x) = d_G^+(x) + d_G^-(x)$.

Dans le cas d'un graphe non orienté $G = (X, E)$,

Le degré est le nombre d'arêtes rattachées au sommet x (une boucle est comptée deux fois). On a la relation : $\sum_{x \in X} d_G(x) = 2|E|$.

Cas particulier

Un graphe ayant pour chaque sommet le même degré est dit graphe régulier et un sommet ayant le degré égal à 0 est dit sommet isolé.

Digraphe

Un Digraphe ou « graphe orienté » est un graphe dont les arêtes sont orientées et dont le couple est défini comme suit : (Départ, Arrivée). Par conséquent, le couple (x, y) est différent du couple (y, x) puisque leur orientation est différente.

Graphe complémentaire

Soit $G = (X, E)$ un graphe simple. Le graphe $\bar{G} = (X, \bar{E})$ est le graphe complémentaire de G si : $\forall e \in E \Leftrightarrow e \notin \bar{E}$.
La figure 1.4 montre un graphe G et son complémentaire $\bar{G}$:

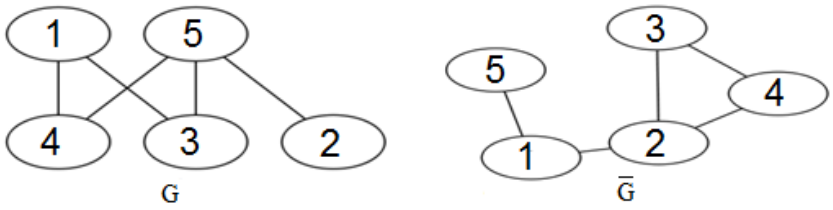


FIGURE 1.4 – Graphe G et son complémentaire

Graphe simple

Un graphe est dit simple lorsque, pour l'ensemble des sommets du graphe, il y a au plus une arête entre deux sommets et pas de boucle.

Sous-graphe, graphe partiel

Un graphe partiel de G est un graphe de la forme $G' = (X, E')$ avec $E' \subset E$. En d'autre terme, c'est un graphe qui a les mêmes sommets que G mais dont on a enlevé certains arcs.

Un sous graphe de G est un graphe H ayant pour sommets un sous ensemble X' des sommets de G et en ne conservent que les arcs/arêtes joignant les sommets de X' ce qui s'écrit :

$$H = (X', E') \text{ avec } X' \subset X \text{ et } E' = \{(x, y) \in E | x, y \in X'\}$$

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

La figure 1.5 montre un graphe G et son graphe partiel G'

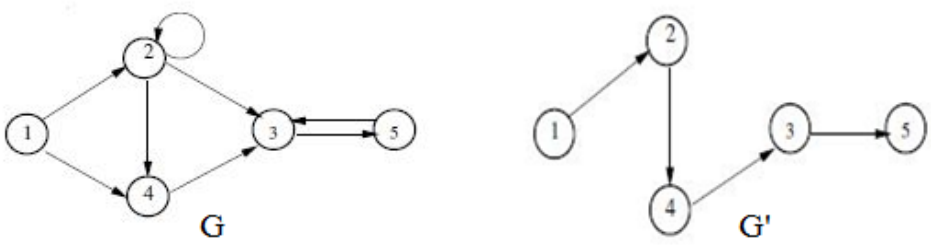


FIGURE 1.5 – Graphe G et son graphe partiel G'

La figure 1.6 montre un graphe G et son sous-graphe H induit par les sommets $X = \{1, 2, 3, 5\}$

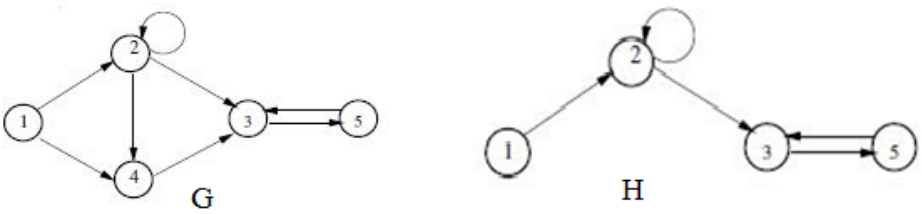


FIGURE 1.6 – Graphe G et son sous-graphe H

Stable

Soit G un graphe, un ensemble stable de G est un sous-ensemble S de sommets tel que deux sommets de S ne sont jamais voisins. Un stable est un ensemble de sommets de G deux à deux non-adjacents.

Les sommets $\{A, C, D, F\}$ forment un stable dans le graphe de la figure 1.7

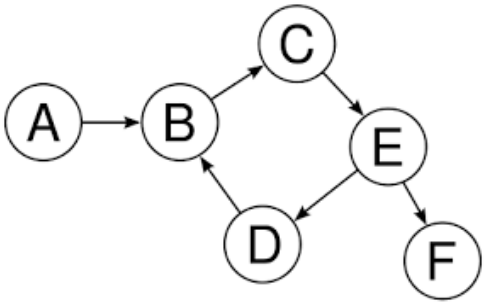


FIGURE 1.7 – Stable

Clique

Soit $G = (X, E)$ un graphe simple tel que X un ensemble de sommets de G deux à deux adjacents. Cet ensemble est dit clique de n sommets et noté k_n . Donc un graphe G est une clique si $d_G(x) = n - 1$ pour tous ses sommets telle que $|X| = n$.

Le graphe de la figure 1.8 montre une clique à 5 sommets, notée K_5

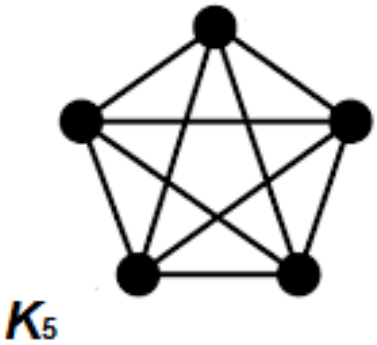


FIGURE 1.8 – Clique

1.2 Chaînes, cycles, chemins et circuits

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

• Le premier et le dernier sommet sont appelés extrémités de la chaîne.

• La longueur d'une chaîne est le nombre d'arcs ou d'arêtes ou d'arcs qui la composent.

• Une chaîne qui n'utilise pas deux fois le même sommet est dite
« élémentaire ».

• Une chaîne qui n'utilise pas deux fois le même arête est dite « Simple ».

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

• Un chemin est dit simple s'il passe une et une seul fois par ses arcs est 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 longueur du plus court chemin de x_i à x_j .

c) Cycle : Un cycle est une chaîne dans les deux extrémités sons confondues.

d) Circuit : Est un chemin simple dont les deux extrémités sont confondues.

1.3 Connexité et forte connexité

Connexité

Soit $G = (X, E)$ un graphe tel que pour tout couple de sommets (x, y) il existe une chaine reliant x et y :

$\forall x, y \in X; \exists x_i \in X; i = 0, \ldots, k; c = (x_0, \ldots, x_k)$ chaine G et $[x_0 = x$ et $x_k = y]$.

On appelle graphe fortement connexe un graphe dont toute pair de sommets (x, y) il existe un chemin de x vers y et un autre chemin de y vers x

La figure 1.9 montre un graphe à trois composantes connexes.

FIGURE 1.9 – Graphe ayant trois composantes connexes

• Un graphe $G = (X, E)$ est dit connexe si tous ses sommets ont deux à deux la relation de connexité. Autrement dit, si G contient une seule composante connexe.
Un graphe est connexe $\Leftrightarrow$ il possède une seule composante connexe.

Remarque

pour trouver les composantes connexes d'un graphe est assez facile mais pour trouver les composantes fortement connexes est plus difficile.

On pourra utiliser l'algorithme suivant :

Algorithme de recherche de composante fortement connexe (cfc)

Soit G un graphe orienté, un moyen de vérifier si ce graphe est fortement connexe et d'appliquer un algorithme de marquage suivant, qui permet de déterminer touts ses composantes fortement connexe.

• **Le principe :**

L'idée de cet algorithme est de parcourir le graphe à partir d'un sommet x dans le sens direct (en suivant les flèches des arcs) et dans le sens opposé (en suivant les flèches des arcs en sens inverse), le premier ensemble obtenu regroupe les sommets accessibles à partir de x , le deuxième regroupe les sommets qui peuvent atteindre x.

l'intersection de ces deux ensembles forment l'ensemble des sommets qui à la fois peuvent atteindre x et sont accessible à partir de x . on obtient ainsi la composante fortement connexe qui contient le sommet x .

- **Énonce :**

Données : un graphe orienté $G = (X, U)$.
Résultat : le nombre k de composantes fortement connexes de (G) ainsi que la liste $\{C_1, C_2, \ldots, C_k\}$ de ces composantes fortement connexes.
0) Initialisation : $k = 0, W = X$.

1) Comporte les étapes suivantes :

1.1) Choisir un sommet de W et le marquer d'un signe (+) et (-).
1.2) Marquer tous les successeurs directs et indirects de x avec (+).
1.3) Marquer tous les prédécesseurs direct et indirects de x avec (-).
1.4) Poser $k = k + 1$ et C_k l'ensemble des sommets marques avec (+) et (-).
1.5) Retirer de W les sommets de C_k et effacer toutes les marques :
On pose $W = W - C_k$.
1.6) On test si $W = \phi$.
Si oui, terminer et aller à (2).
Sinon, retourner à (1, 1).

2) Le nombre de composantes fortement connexes de (G) est k. Chaque ensemble C_k, $i = i, \ldots, k$, correspond aux sommets d'une composante fortement connexe de (G).

Application

Soit le graphe $G = (V, E)$ suivant :

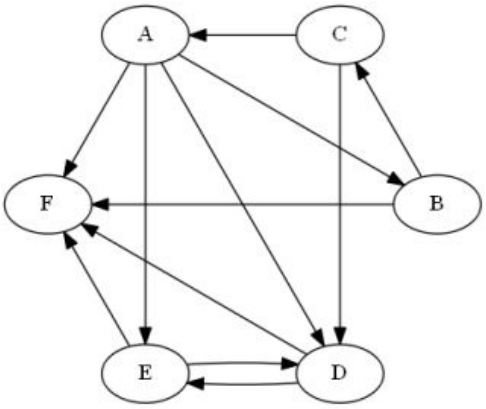


FIGURE 1.10 – Graphe G

G est connexe. Il a donc une seule composante connexe : l'ensemble de tous ses sommets. Nous allons d'abord trouver la composante fortement connexe qui contient le sommet A . Rappel de l'algorithme :

• marquer par $+$ et $-$ le sommet A.

• marquer par $+$ chaque successeur non marqué $+$ d'un sommet marqué $+$.

• marquer par $-$ chaque prédécesseur non marqué $-$ d'un sommet marqué $-$.

• quand on ne peut plus marquer, les sommets marqués $+$ et $-$ constituent la composante fortement connexe contenant A est connexe.

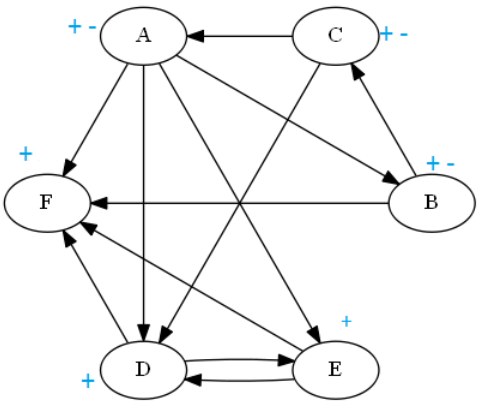


FIGURE 1.11 – Graphe G marqué

La composante fortement connexe qui contient A est $C_1 = \{A, B, C\}$. On retire les sommets A, B et C et les arcs qui leur sont adjacents, puis on recommence par exemple avec le sommet D. On trouve que la composante fortement connexe qui contient D est $C_2 = \{D, E\}$. On retire ces sommets. Il n'y a plus que le sommet F. Ainsi, la troisième composante fortement connexe est $C_3 = \{F\}$.
Le graphe réduit de G est donne par la figure 1.12

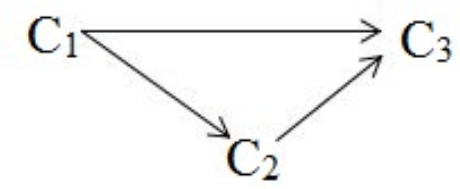


FIGURE 1.12 – Graphe G réduit

1.4 Coloration d'un graphe

Colorier un graphe consiste à affecter une couleur à chaque sommet (resp. arête) de sorte que deux sommets (resp. arêtes) adjacents ne portent pas la même couleur. Le nombre minimum de couleurs nécessaires pour colorier les sommets [resp les arête de G] d'un graphe G est appelé le nombre chromatique de G, [resp l'indice chromatique de G] et noté $\gamma(G)$ [resp : $\gamma(G)$].

• Le nombre chromatique d'un graphe G est borné par $\gamma(G) \leq \Delta(G) + 1$

1) coloration simple

Soit G 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 différente au moins à un seul nombre de sommets coloriés par une autre couleur.

2) Coloration K-équitable

Soit $G = (V, E)$ un graphe donné, une coloration k-équitable est une coloration des sommets du graphe G qui vérifient :

• Le graphe G est k-coloriable.

• $\forall(i,j) \in 1,\ldots,k$, le nombre de sommets coloriés par la couleur i est égale au nombre de sommets coloriés par la couleur j.

1.4.1 Algorithme de coloration [4]

Algorithme de welsh-powell ou algorithme glouton

a) Ranger les sommets par ordre décroissant de leurs degrés ;

b) Choisir une couleur ;

c) Affecter cette couleur au premier sommet de la liste non encore coloré ;

d) Suivre la liste en attribuant cette même couleur à tout sommet.

Qui n'est pas encore coloré et qui n'est pas adjacent à un sommet coloré avec cette couleur.

Continuer jusqu'à ce que la liste soit finie.

Si tous les sommets ne sont pas colorés, choisir une couleur qui n'est pas encore utilisée et recommencer les étapes c et d ;

Continuer tant que chaque sommet n'est pas coloré.

Considérons le graphe G de la figure 1.13.

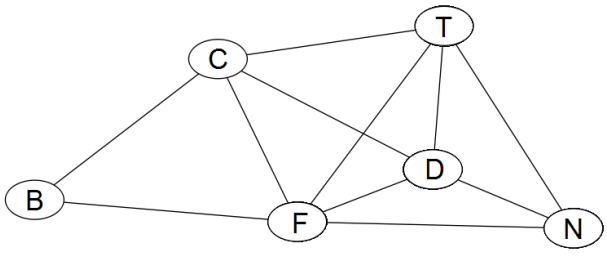


FIGURE 1.13 – Graphe G

Les couleurs attribuées aux sommets de G sont données dans le tableau ci-dessus.

Sommet	F	C	D	T	N	B
Degré	5	4	4	4	3	2
Couleur	C_1	C_2	C_3	C_4	C_2	C_3

Le nombre chromatique de ce graphe est donc égal à 4

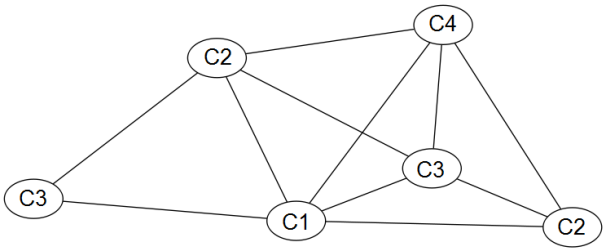


FIGURE 1.14 – Graphe colorie de G

1.5 Quelques opérations sur les graphes

1.5.1 Morphisme de graphe

Un morphisme de graphe ou homomorphisme de graphe est une application entre deux graphes qui respecte la structure de ces graphes. Autrement dit l'image d'un graphe G dans un graphe H doit respecter les relations d'adjacence présentes dans G. Soit $G = (X, E)$ et $H = (T, B)$ deux graphes orientés. On dit qu'une application φ de X dans T réalise un morphisme du graphe G vers le graphe H si elle vérifie la propriété suivant :
$(x, y) \in E \Rightarrow (\varphi(x), \varphi(y)) \in B$

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

1.5.2 Isomorphisme de graphe

L'isomorphisme de graphe est une relation univoque des nœuds entre deux graphes qui respecte leurs attributs et ceux des arêtes.

Soient $G = (X, E)$ et $H = (X', E')$ deux graphes. S'il existe une relation univoque m :

$X \to X'$ telle que $(x_1; x_2) \in E$, $(m(x_1), m(x_2)) \in E'$, alors G et H sont isomorphes.

1.5.3 Homomorphisme de graphe

Par conséquent, un homomorphisme est un isomorphisme lorsque la relation m est bijective. Les deux notation sont souvent confondues. un homomorphisme de G vers G' permet qu'il y ait une arête entre deux sommets dans G' faisant partie de la relation, sans qu'il y en ait entre leurs antécédents dans G. Dans le cas de l'isomorphisme, une telle situation est impossible.

$G = (V(G), E(G))$, graphe non orienté.
Un homomorphisme de G vers H est une application $c : V(G) \to V(H)$, telle que $uv \in E(G) \Rightarrow c(u)c(v) \in E(H)$.

Notation :

- $G \to H$: il existe un homomorphisme de G vers H.
- $G' \to H$: il n'existe aucun homomorphisme de G vers H.

Remarque

- si H contient une boucle, alors $G \to H$ pour tout G.
- si G contient une boucle alors $G \to H$ si et seulement si H contient une boucle.

On s'intéresse aux graphes sans boucles

- si $G \to H$ et $H \to J$ alors $G \to J$ (composition).
- si G sous-graphe de H, alors $G \to H$ (inclusion).

1.6 Couplage

Soit G un graphe simple. Un couplage C de G est un sous-graphe partiel 1-régulier de G. On peut aussi dire qu'un couplage est un ensemble d'arêtes deux à deux non-adjacentes. Un sommet v est dit saturé par un couplage C si v est l'extrémité d'une arête de C. Dans le cas contraire, v est dit insaturé.
Un couplage maximum est un couplage contenant le plus grand nombre possible d'arêtes. Un graphe peut posséder plusieurs couplages maximum.
Un couplage parfait est un couplage où chaque sommet du graphe est saturé.

Les arêtes en gras montre un couplage maximum dans le graphe de la figure 1.15

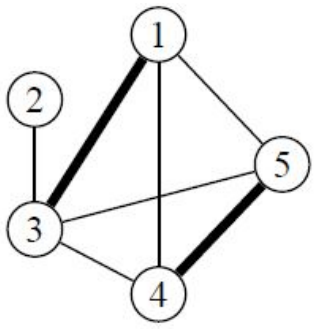


FIGURE 1.15 – Couplage

Les arêtes en gras montre un couplage maximum et parfait dans le graphe de la figure 1.16

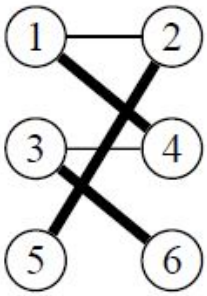


FIGURE 1.16 – Un couplage maximum et parfait

1.7 Transversal

Une couverture par sommets ou transversale d'un graphe G est un ensemble minimum de sommets pour couvrir toutes les arêtes. Il est considéré comme l'un des problèmes algorithmiques classiques.

1.8 Point d'articulation

Un point d'articulation d'un graphe est un sommet dont la suppression augmente le nombre de composantes connexes.

• Un isthme est une arête dont la suppression augmente le nombre de composantes connexes.

• Un ensemble d'articulation $A \subset X$ d'un graphe connexe $G = (X, E)$ est un ensemble de sommets tel que le sous-graphe G' déduit de G par suppression des sommets de A ne soit plus connexe.

1.9 Quelques types des graphes

1.9.1 Graphe complet

Un graphe complet est un graphe simple dont tous les sommets sont deux à deux adjacents, c'est-à-dire que tout couple de sommets disjoints est relié par une arête.
Pour tous entier naturel non nul n, on note k_n le graphe complet d'ordre n. Le nombre d'arêtes du graphe complet k_n est égal à $n(n-1)/2$.

La figure 1.17 montre un graphe complet à 5 sommets.

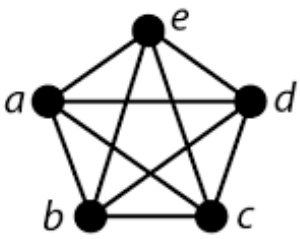


FIGURE 1.17 – Graphe complet à 5 sommets.

1.9.2 Graphe biparti

Un graphe G est dit biparti s'il existe une partition de son ensemble de sommets en deux sous-ensembles U et V telle que chaque arête de G ait une extrémité dans U et l'autre dans V, deux sommets de même ensemble ne soient pas adjacents.
Un graphe biparti permet notamment de représenter une relation binaire.
$G = (X, E)$ est biparti si et seulement si $\exists U \subseteq X, V \subseteq X$ tels que
• $U \cup V = X$.
• $U \cap V = \varnothing$.
• $G[U]$ et $G[V]$ sont deux stables.

$\trianglerighteq$ Un graphe G est dit graphe biparti complet, si chaque sommet de U est adjacent à tous les sommets de V et vice versa. Si de plus le graphe G est simple, alors G est un graphe simple biparti-complet, un tel graphe est noté $k_{m,n}$. Clairement $k_{m,n}$ a $m * n$ arêtes avec $|U| = m$ et $|V| = n$. $\trianglerighteq$ Si $|U| = |V|$ on dit que G est biparti équilibré.

Le graphe biparti complet $k_{3,2}$ et donné par la figure 1.18.

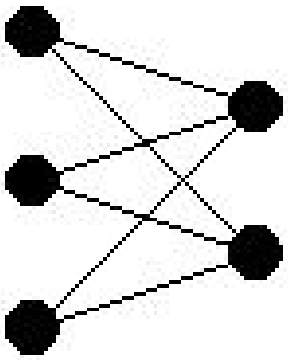


FIGURE 1.18 – Graphe biparti complet

1.9.3 Graphe planaire

Un graphe planaire est un graphe qui a la particularité de pouvoir se représenter sur un plan sans qu'aucune arête n'en croise une autre. Autrement dit, ces graphes sont précisément ceux que l'on peut plonger dans le plan.

1.9.4 Graphe valué ou pondéré

On dit qu'un graphe est valué, si on peut affecter des valeurs aux arêtes, les valeurs représentent, soient : coût, probabilité...etc.

1.9.5 Arbre

Un arbre est un graphe $G = (X, E)$ connexe sans cycle. D'après la définition, un arbre est un graphe simple sans boucles, ayant $(n-1)$ arcs.
On appellera quand ce la existe :

- Racine de l'arbre le seul sommet de G qui n'a pas de prédécesseur.
- Feuilles de l'arbre les sommets qui n'ont pas de successeur.
- Nœud de l'arbre tous les autres sommets.
- Branche de l'arbre tout chemin de la racine vers une feuille.
- Descendant de x les successeurs de x.
- Ascendant de x le prédécesseur de x.

Lorsque chaque sommet a au plus deux successeurs on parle d'arbre binaire.

L'arbre de la figure 1.19, admet une racine donnée par 1, trois feuilles données par 4, 5 et 6.

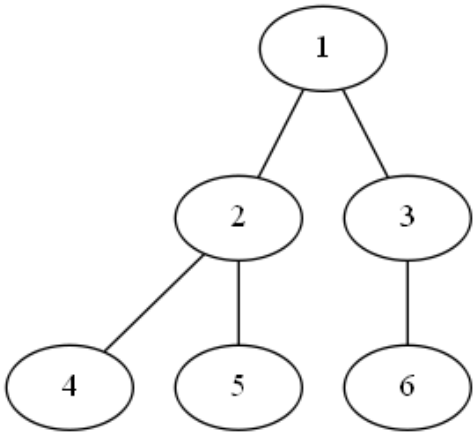


FIGURE 1.19 – Arbre

1.9.6 Arborescence

Un arborescence est un graphe $G = (X, E)$ 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.

La figure 1.20 montre une arborescence.

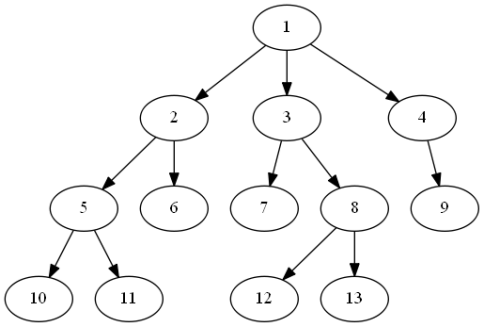


FIGURE 1.20 – Arborescence

Conclusion

Ce premier chapitre était consacré à la définition de certaines généralités sur la théorie des graphes, nous allons voir maintenant quelques approches d'optimisation dans les graphes connexes (ou bien réseaux).

CHAPITRE 2

QUELQUES APPROCHES D'OPTIMISATION DANS LES RÉSEAUX

Introduction

Les graphes permettent de manipuler plus facilement des objets et leurs relation avec une représentation graphique naturelle. Beaucoup de problèmes peuvent être modéliser en utilisant des graphes, Les problèmes de cheminement dans les graphes, en particulier la recherche de plus court chemin, comptent parmi les problèmes les plus anciens de la théorie des graphes et les plus importants par leurs application. Dans ce chapitre nous allons présenté brièvement quelques problèmes d'optimisations pour les quelles nous donnons quelque algorithmes de résolutions pour chaque problème. Parmi les problèmes nous citons : Problème du plus court chemin, problème de l'arbre couvrant de poids minimum et problème d'ordonnancement.

2.1 Définition d'un réseau

Un réseau est un graphe valué $G = (X, E, C)$ dans lequel il y a un sommet s (source) tel que $d_G^-(s) = 0$ et un sommet puits (p) tel que $d_G^+(p) = 0$, pour $e \in E$ la valeur $C(e)$ est appelé la capacité de l'arc e.

2.2 Problème de plus court chemin

L'objectif de ce problème est de déterminer un chemin entre des sommets d'un graphe qui minimise une certaine fonction. Il existe de nombreuses variantes de ce problème. On suppose donner un graphe fini, orienté ou non selon les cas ; de plus, chaque arc ou arête possède une valeur qui peut être un poids ou une longueur. Un chemin le plus court entre deux nœuds donnés est un chemin qui minimise la somme des valeurs des arcs traversés. Pour calculer un plus court chemin, il existe de nombreux algorithmes et se diffèrent selon les propriétés des graphes :

- G est sans circuit et suivant le problème considéré :
- Recherche d'un plus court chemin d'un sommet à un autre.
- Recherche d'un plus court chemin d'un sommet à tous les autres sommets.
- Recherche d'un plus court chemin entre tous les couples de sommets.

Dans de nombreux cas, il existe des algorithmes de complexité en temps polynomiale, comme l'algorithme de Dijkstra dans des graphes avec poids positifs.

2.2.1 Algorithme de Djikstra

Cet algorithme détermine les plus courts chemins d'un point s à tous les autres points d'un réseau $R = (V; E; C)$. Il suppose que les longueurs sur les arcs sont positives ou nulles. L'idée de cet algorithme est de partager les nœuds en deux groupes : ceux dont on connaît la distance la plus courte au point s (ensemble S) et ceux dont on ne connaît pas cette distance (ensemble S'). On part avec tous les nœuds dans S'. Tous les nœuds ont une distance infinie ($\lambda(x) = +\infty$) avec le point s, excepté le point s lui-même qui a une distance nulle ($\lambda(s) = 0$). A chaque itération, on choisit le nœud x qui a la plus petite distance au point s. Ce nœud est déplacé dans S. Ensuite, pour chaque successeur y de x, on regarde si la distance est la plus courte entre s et y et elle ne peut pas être améliorée en passant par x, si c'est le cas, $p(y)$ est modifié. Ensuite, on recommence avec un autre nœud.

Début

Donnés : $R = (V; E; C)$ un réseau avec $c_{ij} > 0$, $(i, j) \in E$.

Sorti : λ_j : le plus court chemin entre s et j , $j \in V$.

Soit S : la liste des nœuds marqué, arcs, nœuds auxquels on a calculé la plus courte distance entre s et j.

a) Initialisation

Poser $S = \{s\}$ et

$$\lambda_j = \begin{cases} 0 & si \quad j = s \\ \infty & si \quad \bar{S} \in V \setminus S \end{cases}$$

b) Choix du nœud a marqué

• Sélectionner $k \in \bar{S}$ tel que $\lambda = min\{\lambda_j\}$ avec $j \in \bar{S}$

• Poser $\bar{S} = \bar{S} \setminus \{k\}$.

Si $\bar{S} = \emptyset$ fin de l'algorithme.

Si non allé à (a).

c) Mise a jour des distances

Pour tout $j \in \Gamma^+(k)$ avec $j \in \bar{S}$ faire $\lambda_j = min\{\lambda_j, \lambda_j + C_k\}$.

Retourner à (b)

fin ;

Justification

Pour qu'un nœud x ait une distance $\lambda(x)$ différente de $+\infty$, il faut qu'il ait un de ses prédécesseurs dans S. De plus, $\lambda(x)$ est la plus petite distance de s à x en empruntant uniquement les nœuds de S. Supposons que $\lambda(x)$ soit la plus petite distance de tout l'ensemble S'. Supposons également que le plus court chemin pour aller de s à x passe

par t et y. Cela signifie que y est le prédécesseur de x. Il faudra donc que $\lambda(y) \leq \lambda(x)$. Cependant, si le chemin le plus court pour aller de s à x passe par t et y alors le chemin le plus court pour aller de s à y passe par t. Ce qui signifie que y connaît sa plus courte distance à s. Malheureusement, elle est supérieure à $\lambda(x)$ ce qui contredit l'hypothèse que $\lambda(x)$ est la plus courte distance de S'.

Appliquons l'algorithme de djikstra au graphe de la figure 2.1.

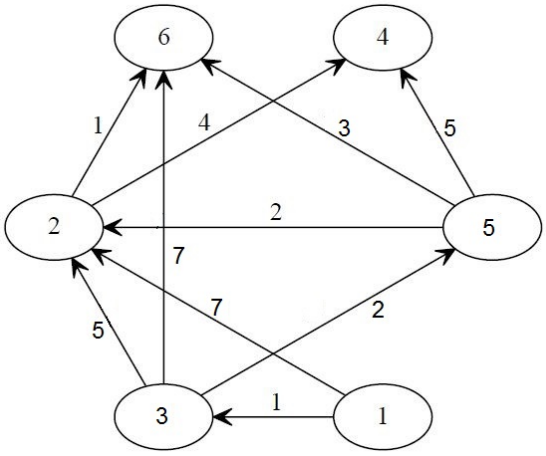


FIGURE 2.1 – Réseau

Itération 1

$S = \{1\}, \bar{S} = V \backslash S, k = 1$

J	1	2	3	4	5	6
λ_j	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$

Mise a jour des distances : $\Gamma^+(1) = \{2, 3\}$

J	1	2	3	4	5	6
λ_j	0	7	1	$+\infty$	$+\infty$	$+\infty$

Itération 2

$k = 3, \Gamma^+(3) = \{5, 6, 2\}$.

J	1	2	3	4	5	6
λ_j	0	7	1	$+\infty$	3	8

Itération 3

$K = 5, \Gamma^+(5) = \{4, 2, 6\}$.

J	1	2	3	4	5	6
λ_j	0	5	1	8	3	8

Itération 4

$K = 2, \Gamma^+(2) = \{4, 6\}$.

J	1	2	3	4	5	6
λ_j	0	5	1	8	3	6

Itération 5

$K = 6, \Gamma^+(6) = \emptyset$.

J	1	2	3	4	5	6
λ_j	0	5	1	8	3	6

Itération 6

$K = 4, \Gamma^+(4) = \emptyset, \bar{S} = \emptyset$.alors on arrête l'algorithme.

Les λ_j sont les plus courte distances.

L'arborescence des plus courts distances, issue du sommet 1 dans le réseau est la suivante.

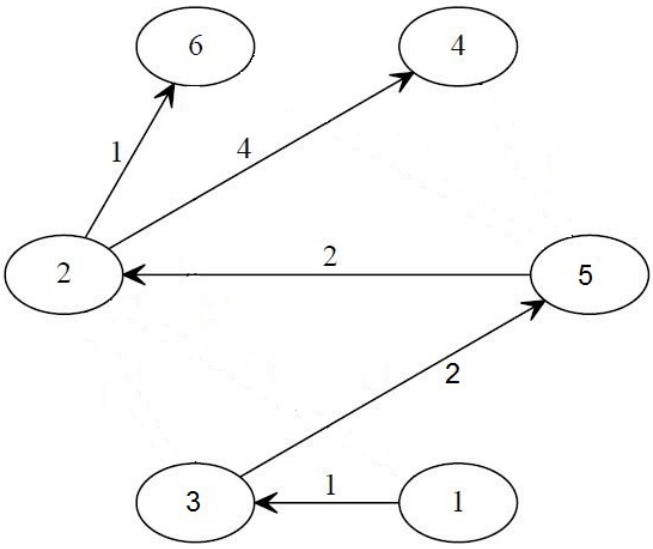


FIGURE 2.2 – L'arborescence des plus courts chemins

2.3 Algorithme de Bellman

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

Son principe

L'idée de l'algorithme de Bellman, est de calculer de proche en proche, l'arborescence des plus courtes distances, issue du sommet s a un sommet donnée p.

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

Données : $R = (S, A, V)$, r racine sans circuit.
Résultats : π, A, C : sous ensemble accessible a partir du sommet S.

a) Initialisation

$\pi(S) = 0, A = \emptyset, c = \{1\}$.

b) Procédure de calcul

- Chercher un sommet $j \in \bar{C}$ tel que $\Gamma_j^- \in C$.
- Calculer $\pi(j) = Min\{\pi(i) + V_{ij}\}$, $i \in \Gamma_j^-$.
- Poser $C = C \cup \{j\}$.

c) Test d'arrêt

– si :$|C| = |S|$ terminer.

– Si non : retourner en (b).

Appliquons l'algorithme de Bellman au graphe de la figure 2.3

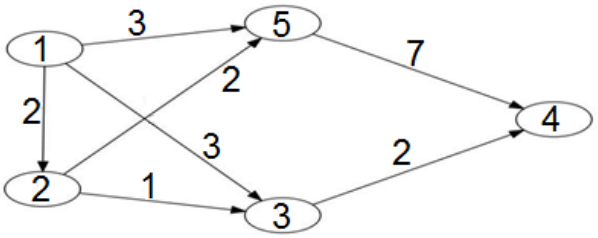


FIGURE 2.3 – le réseau $R = (S, A, V)$

a) Initialisation

Soit 1 un sommet de X, $\pi(1) = 0, c = \{1\}, A = \emptyset$.

Itération 1

b) Procédure de calcul

- $2 \in \bar{C}$ tel que $\Gamma_2^- \in C$
- $\Pi(2) = MIN\{0 + 2\} = 2$ d'où $a = (1, 2)$.
- $C = \{1, 2\}$, $A = A \cup \{(1, 2)\} = \{1, 2\}$.

c- $|C| \neq |S|$

Itération 2

b) Procédure de calcul

• $3 \in \bar{C}$ tel que $\Gamma_3^- \in C$, $5 \in \bar{C}$ tel que $\Gamma_5^- \in C$.

• $\Pi(3) = MIN\{\pi(1) + v(1,3)\,;\, \pi(2) + v(2,3)\} = MIN\{0+3, 2+1\} = 3$ d'où $a = (1,3)$.

$\Pi(5) = MIN\{\pi(1) + v(1,5)\,;\, \pi(2) + v(2,5)\} = MIN\{0+3; 2+2\} = 1$ d'où $a = (1,5)$.

• $C = C \cup \{3,5\} = \{1,2,3,5\}$, $A = A \cup \{(1,3),(1,5)\} = \{(1,2),(1,3),(1,5)\}$

c- $|C| \neq |S|$.

Itération 3

b) Procédure de calcul

• $4 \in \bar{C}$ tel que $\Gamma_4^- \in C$.

• $\Pi(4) = MIN\{\pi(3) + v(3,4); \pi(5) + v(3,5)\} = MIN\{3+2; 1+7\} = 5$ d'où $a = (3,5)$.

• $C = C \cup \{4\} = \{1,2,3,4,5\}$; $A = A \cup \{(3,4)\} = \{(1,2),(1,3),(1,5),(3,4)\}$

c) Test d'arrêt

$|C| = |S|$; terminer.

On obtient donc l'arborescence des plus courtes distances suivante :

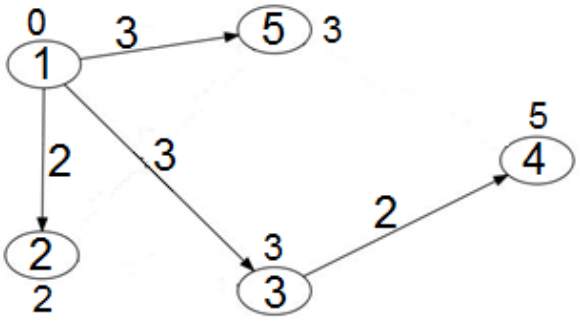


FIGURE 2.4 – l'arborescence obtenue

2.4 Arbre couvrant minimum

Définition

Les graphes sont un outil de représentation puissant et l'arbre couvrant minimum peut s'interpréter de différentes manières selon ce que représente le graphe. De manière générale si on considère un réseau où un ensemble d'objets doivent être reliés entre eux (par exemple un réseau électrique et des habitations), l'arbre couvrant minimum est la façon de construire un tel réseau en minimisant un coût représenté par le poids des arêtes (par exemple la longueur totale de câble utilisée pour construire un réseau électrique).

2.4.1 Algorithme de kruskal

Son principe

L'algorithme construit un arbre couvrant minimum en sélectionnant des arêtes par poids croissant. Plus précisément, l'algorithme considère toutes les arêtes du graphe par poids croissant (en pratique, on trie d'abord les arêtes du graphe par poids croissant) et pour chacune d'elle, il la sélectionne si elle ne crée pas un cycle.

Algorithme

Début

Données : $G = (V, E, C)$ un graphe.
Résultat : $T = (V, F)$ arbre couvrant de poids minimum.
Trier les arêtes de E par ordre de poids croissants :
$C(e_1) \leq C(e_2)...C(e_m)$
$F = \emptyset$
pour $i = 1$ à $|E|$ faire
Si l'ajout de e_j à F ne crée pas de cycle alors
$F \leftarrow F \cup \{e_j\}$ retourner $T = (V, F)$.
Fin si ;

Fin pour ; Si $||E|| = n - 1$ **Fin.**

Soit $G = (X, E, C)$ un graphe connexe. On veut donner un arbre couvrant de poids minimum, pour le projet d'installation de lignes téléphonique. Appliquons l'algorithme de Kruskal pour le graphe de la figure 2.5.

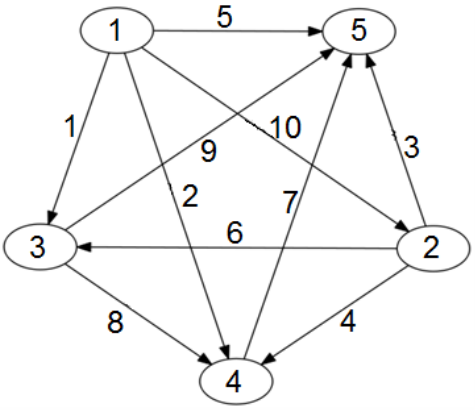


FIGURE 2.5 – Graphe connexe

Initialisation

On trie les arêtes de E par poids croissants :

I	1	2	3	4	5	6	7	8	9	10
u_i	$(1,3)$	$(1,4)$	$(2,5)$	$(2,4)$	$(1,5)$	$(2,3)$	$(4,5)$	$(3,4)$	$(3,5)$	$(1,2)$
$W(u_i)$	1	2	3	4	5	6	7	8	9	10

Soit $E = \emptyset$; $i = 1; m = 10$.

Itération 1

On a : $e_1 = (1,3)$; soit $E = E \cup \{e_1\} = \{(1,3)\}$ ne crée pas de cycle dans le graphe (X, E), alors on pose $E = \{(1,3)\}$
$||E|| = 1$, on a $i \neq m$ alors on pose $i = i + 1 = 2$.

Itération 2

On a : $e_1 = (1,4)$; soit $E = E \cup \{e_2\} = \{(1,3),(1,4)\}$ ne crée pas de cycle dans le graphe (X,E), alors on pose $E = \{(1,3),(1,4)\}$.

$||E|| = 2$, on a $i \neq m$ alors on pose $i = i + 1 = 3$.

Itération 3

On a : $e_1 = (2,5)$; soit $E = E \cup \{e_3\} = \{(1,3),(1,4),(2,5)\}$ ne crée pas de cycle dans le graphe (X,E), alors on pose $E = \{(1,3),(1,4),(2,5)\}$.
$||E|| = 3$, on a $i \neq m$ alors on pose $i = i + 1 = 4$.

Itération 4

On a : $e_1 = (2,4)$; soit $E = E \cup \{e_4\} = \{(1,3),(1,4),(2,5),(2,4)\}$ ne crée pas de cycle dans le graphe (X,E), alors on pose $\{(1,3),(1,4),(2,5),(2,4)\}$.
$||E|| = 4 = n - 1$ Terminer.

L'arbre de poids minimum est représenté par le graphe $A' = (V,E')$ Le coût total de l'installation est :
$C(e_1) + C(e_2) + C(e_3) + C(e_4) = 1 + 2 + 3 + 4 = 10$

L'arbre couvrante de poids minimum obtenue est donnée par la figure 2.6.

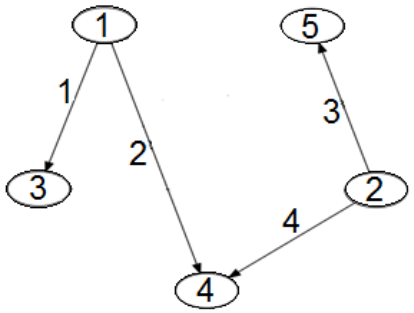


FIGURE 2.6 – Arbre couvrant de poids minimum

2.4.2 Algorithme de PRIM

L'algorithme de Prim, est un algorithme qui permet de trouver un arbre couvrant minimal dans un graphe connexe valué et non orienté. En d'autre termes, cet algorithme trouve un sous-ensemble d'arrêtes formant un arbre sur l'ensemble des sommets du graphe initial, et tel que la somme des poids de ces arrêtes soit minimal. Si le graphe n'est pas connexe, alors l'algorithme ne déterminera l'arbre couvrant minimal que d'une composante connexe du graphe.il a été conçu en 1957 par Robert C.Prim.

Principe de l'algorithme de PRIM

L'algorithme consiste à choisir arbitrairement un sommet et à faire croitre un arbre à partir de ce sommet. Chaque augmentation se fait de la manière la plus économique possible.

Algorithme

Données : soit $G = (V, E, C)$ un graphe connexe.
Résultat : $T(V, E)$ arbre couvrant de poids minimum.

Soit I l'ensemble de nœuds déjà inclus dans l'arbre et NI l'ensemble de nœuds non encore inclus.

a) Initialisation
$I = \emptyset$, $NI =$ l'ensemble de tous les nœuds.

Début

1. Mettre le nœud de départ dans I ;
2. Si T est connexe alors aller a (5) ;
3. Trouver un nœud de NI dont la distance à un nœud quelconque a de I est minimal ; On relie le nœud a à b et faire $I = I \cup b$, $NI = NI/\{b\}$;
4. Aller à (3) ;

Si $||I|| = n - 1$

5. Fin.

Appliquons l'algorithme de Prim pour le graphe de la figure 2.7

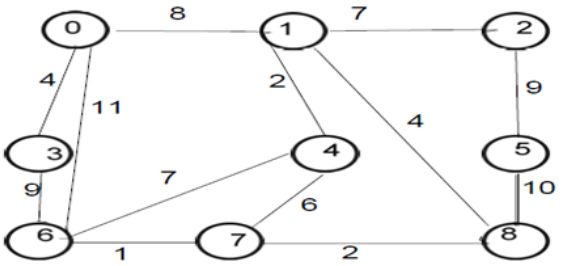


FIGURE 2.7 – Graphe connexe

Initialisation

Soit $I = \emptyset$;

$NI = \{(0,1),(0,3),(0,6),(1,2),(1,4),(1,8),(2,5),(3,6),(4,6),(4,7),(5,8),(6,7),(7,8)\}$;

Commençant par le sommet 0.

e_i	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}	e_{13}
(i,j)	$(0,1)$	$(0,3)$	$(0,6)$	$(1,2)$	$(1,4)$	$(1,8)$	$(2,5)$	$(3,6)$	$(4,6)$	$(4,7)$	$(5,8)$	$(7,6)$	$(8,7)$

Itération 1

On a $e_2 = (0,3)$ avec $C(0,3) = 4$ (on choisit l'arrête de faible poids) ;

Soit $I = I \cup \{e_2\} = \{(0,3)\}$;

Le graphe G n'est pas connexe alors :

• $I = \{(0,3)\}$;

• $NI = \{(0,1),(0,6),(1,2),(1,4),(1,8),(2,5),(3,6),(4,6),(4,7),(5,8),(6,7),(7,8)\}$;

$|I| = 1$ on continue ;

Itération 2

On a $e_1 = (0,1)$ avec $C(0,1) = 8$;

Soit $I = I \cup \{e_1\} = \{(0,1)\}$;

Le graphe G n'est pas connexe alors :

• $I = \{(0,3),(0,1)\}$;

• $NI = \{(0,6),(1,2),(1,4),(1,8),(2,5),(3,6),(4,6),(4,7),(5,8),(6,7),(7,8)\}$;

$|I| = 2$ on continue ;

Itération 3

On a $e_5 = (1,4)$ avec $C(1,4) = 2$;

Soit $I = I \cup e_5 = \{(1,4)\}$;

Le graphe G n'est pas connexe alors :

• $I = \{(0,3),(0,1),(1,4)\}$;

• $NI = \{(0,6),(1,2),(1,8),(2,5),(3,6),(4,6),(4,7),(5,8),(6,7),(7,8)\}$;

$|I| = 3$ on continue ;

Itération 4

On a $e_6 = (1,8)$ avec $C(1,8) = 4$;

Soit $I = I \cup e_6 = \{(1,8)\}$;

Le graphe G n'est pas connexe alors :

• $I = \{(0,3),(0,1),(1,4),(1,8)\}$;

• $NI = \{(0,6),(1,2),(2,5),(3,6),(4,6),(4,7),(5,8),(6,7),(7,8)\}$;

$|I| = 4$ on continue ;

Itération 5

On a $e_{13} = (8,7)$ avec $C(8,7) = 2$;

Soit $I = I \cup \{e_{13}\} = \{(8,7)\}$;

Le graphe G n'est pas connexe alors :

• $I = \{(0,3),(0,1),(1,4),(1,8),(8,7)\}$;

• $NI = \{(0,6),(1,2),(2,5),(3,6),(4,6),(4,7),(5,8),(6,7)\}$;

$|I| = 5$ on continue ;

Itération 6

On a $e_{12} = (7, 6)$ avec $C(7, 6) = 1$;
Soit $I = I \cup \{e_{12}\} = \{(7, 6)\}$;

Le graphe G n'est pas connexe alors :
- $I = \{(0, 3), (0, 1), (1, 4), (1, 8), (8, 7), (7, 6)\}$;
- $NI = \{(0, 6), (1, 2), (2, 5), (3, 6), (4, 6), (4, 7), (5, 8)\}$;

$|I| = 6$ on continue ;

Itération 7

On a $e_4 = (1, 2)$ avec $C(1, 2) = 7$;
Soit $I = I \cup \{e_4\} = \{(1, 2)\}$;

Le graphe G n'est pas connexe alors :
- $I = \{(0, 3), (0, 1), (1, 4), (1, 8), (8, 7), (7, 6), (1, 2)\}$;
- $NI = \{(0, 6), (2, 5), (3, 6), (4, 6), (4, 7), (5, 8)\}$;

$|I| = 7$ on continue ;

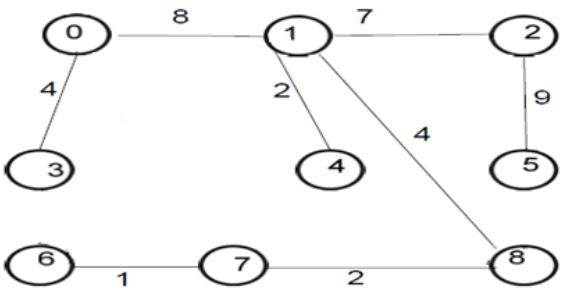
Itération 8

On a $e_7 = (2, 5)$ avec $C(2, 5) = 9$;
Soit $I = I \cup \{e_7\} = \{(2, 5)\}$;

Le graphe G est connexe alors :
- $I = \{(0, 3), (0, 1), (1, 4), (1, 8), (8, 7), (7, 6), (1, 2), (2, 5)\}$;
- $NI = \{(0, 6), (2, 5), (3, 6), (4, 6), (4, 7), (5, 8)\}$;

$|I| = 8 = (n - 1)$ on arrête ;

L'arbre couvrant de poids minimum est :



Le poids total est :

$C(e_2)+C(e_1)+C(e_5)+C(e_6)+C(e_{13})+(e_{12})+(e_4)+(e_7) = 4+8+2+4+2+1+7+9 = 37.$

2.5 Problème d'ordonnancement

Définition

Les problèmes d'ordonnancement sont apparus au départ dans la planification de grands projets. Le but était de gagner du temps sur leurs réalisations. De tels projets sont constitués de nombreuses étapes, également appelées tâches. Des relations temporelles existent entre ces dernières.

Les critères d'optimisation sont les suivantes :

– la date d'achèvement du projet : on va essayer de terminer le plus tôt possible.

– l'utilisation des ressources : on va essayer de lisser l'utilisation des ressources, (personnels, matériels, financiers. . .).

On a alors les types de contraintes suivants :

1. Contraintes potentielles :

– Contrainte d'antériorité : la tâche i ne peut commencer que quand j est achevée).

– Localisation temporelle : la tâche i ne peut commencer avant une date d où doit être terminée avant la date d_0.

2. Contraintes cumulatives (sur les moyens) : respect les moyens disponibles.

3. Contraintes disjonctives : les tâches i et j ne peuvent être réalisées simultanément (Pour des raisons de sécurité, par exemple).

Le but est donc de déterminer un ordre et un calendrier d'exécution des tâches, qui minimise la durée totale de réalisation du projet.
Les taches : une tache est l'activité élémentaire caractérisée par :
– Sa durée d_i.
– Sa date de début t_i et sa date de fin f_i, la date de fin est égale à $t_i + d_i$.
– Éventuellement les ressources nécessaire à son accomplissement.
Les taches sont supposées non-préemptives : une fois que l'exécution a commencé, elles se poursuivent sans interruption jusqu'à la fin. Il existe entre les tâches d'un projet les relations de précédence, qui signifient par exemple qu'une tâche ne peut commencer avant l'achèvement de certaines autres tâches.

2.5.1 Représentation du réseau PERT

La méthode PERT est une méthode de gestion de projet visant à prévoir les propriétés d'un projet en termes de temps, délais et coûts.
Son principe est de découper un projet en un ensemble d'actions appelées tâches et de les représenter sous forme graphique selon un graphe de dépendances.
Grâce à la chronologie et l'interdépendance de chacune des tâches, on structure ainsi l'ensemble du projet et on peut alors planifier la réalisation de chacune des tâches les unes par rapport aux autres, afin de minimiser les délais, ainsi que réduire l'impact des retards lors de l'exécution des différentes tâches.

2.5.2 Diagramme de Gant

On peut visualiser 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 possède ainsi pour toutes les tâches, on saura à tout moment quel est l'état réel d'avancement du projet.

On considère le problème d'ordonnancement suivant :

Tâches	Tâches antérieures	Durée
A	−−	2
B	−−	3
C	A	2
D	A, B	4
E	C	4
F	C	3
G	E, D	4
H	F, G	2

Ainsi la représentation des tâches précédentes est donnée dans la figure suivante :

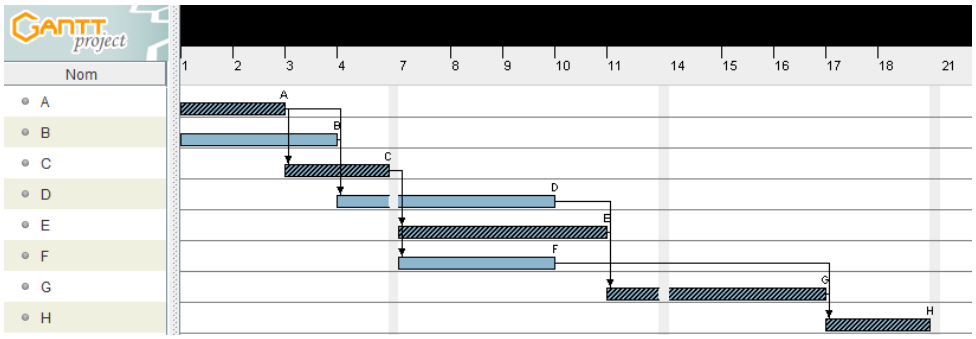


FIGURE 2.8 – Un exemple de diagramme de Gantt

2.5.3 Méthode CPM [8]

Cette méthode consiste à calculer pour chaque tâche du projet :

La date de début au plus tôt : noté par t_i est la date pour laquelle on peut commencé sa réalisation. Elle correspond à la date pour lesquelles toutes les tâches antérieur 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.

$t_i - max_{j \in p^-(i)}\{t_d + t_j\}$ avec :

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

La date de début au plus tard : noté 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 $T_i - min_{j \in p^+(i)}\{T_j - d^-\}$.

La figure suivante représente le réseau CPM de l'exemple précédent :

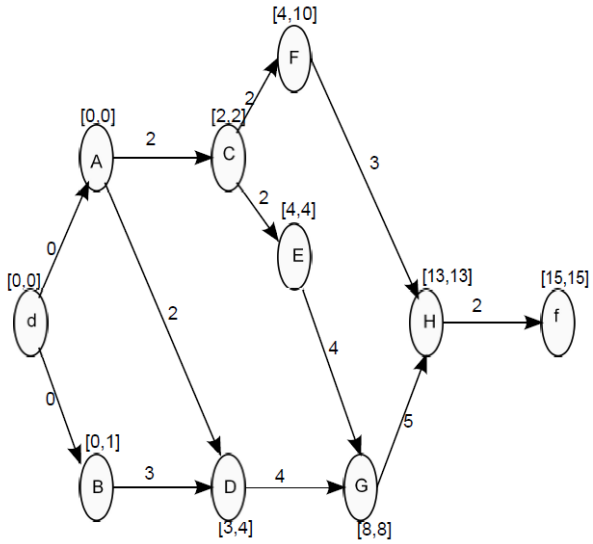


FIGURE 2.9 – Le réseau CPM

Le chemin critique est composé des tâches suivantes $C = \{A, C, E, G, H\}$.

2.5.4 Méthode PERT [8]

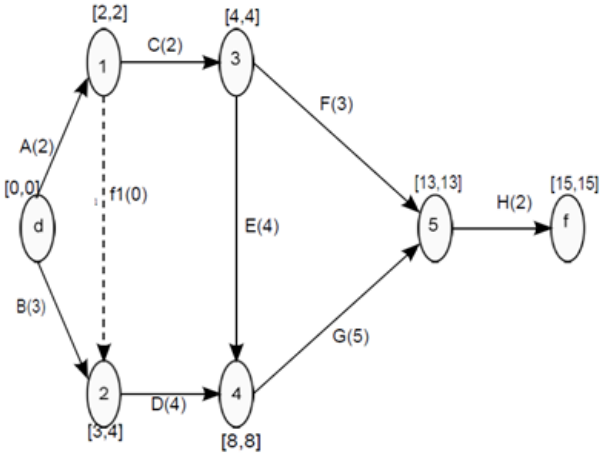
Pour cette méthode, on calcule les dates au plus tôt et au plus tard des évènements tel que la tâche (i, j) de durée d_{ij}, ou i, j représentent respectivement l'événement initial et terminal de cette tâche, avec la date de début au plus tôt d'un évènement i noté t_i est donné par :

$$\begin{cases} t_d & - & 0 \\ t_j & - & max_{j \in p^-(i)}\{t_i + d_{ij}\} \end{cases}$$

On calcule la date de début au plus tard d'un événement i notée T_i :

$$\begin{cases} T_f & - & t_f \\ T_i & - & min_{j \in p^+(i)}\{T_j + d_{ij}\} \end{cases}$$

Dans le graphe suivant, la durée minimale de réalisation du projet est de 15 jours, ainsi que le chemin critique est : $C = \{A, C, E, G, H\}$.



2.6 Problème de flot maximum

Le problème de flot maximum consiste à transporter la quantité maximale possible d'une origine vers une destination données, sans dépasser les capacités des arcs.

Soit un réseau de transport de données noté $R = (V, E, C), s, p, C$ avec :

- $G = (V, E)$ un graphe orienté.
- $s \in V$ appelé sommet source.
- $p \in V$ appelé sommet puits.
- $C : E \to N$ fonction capacité (à chaque arc e_{ij} on associe une capacité $c_{ij} > 0$).
- Un $s - p$ flot (f) est réalisable dans R s'il satisfait les contraintes suivantes :
- Contrainte de capacité :

$$0 \leq f_{ij} \leq C_{ij} \qquad \forall (i, j) \in E.$$

Contrainte de conservation de flot

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

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

La valeur du flot est :

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

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

Coupe minimum : la coupe dans le graphe $G = (V, E)$ définie par le sous-ensemble non vide de nœuds S, est l'ensemble d'arcs noté $\delta^+(S)$ sortant de S.

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

- 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 capacité de ses arcs.

Théorème 1. (Dualité faible)

La capacité d'une s-p coupe est supérieure ou égale à la valeur d'un s-p flot.[7]

Théorème 2. (Dualité forte)

La valeur d'un s-p flot maximal est égale à la capacité d'une s-p coupe minimale.[7]

2.6.1 Algorithme de Ford-Fulkerson

Idée de l'algorithme est de trouver un chemin augmentant et augmenter le flot sur ce chemin.

1) Initialisation

• Partir d'un flot réalisable quelconque ($f = 0$).

• Marquer la source d'un signe $(., +\infty)$.

2) Examination d'un sommet

• Pour tout sommet j tel que $e = (i, j) \in A$ avec $\{f(e) < C(e)$, i marqué et j non marqué $\}$ faire marqué j d'un signe $\{+i, min[\Delta, C(e) - f(e)]\}$

Δ : c'est la quantité de flots qui arrive au sommet i

3) Test d'optimalité

Si le sommet puits est marqué aller à l'étape 4

Sinon voir s'il existe d'autre sommet marqué mais non examiné alors retourner à l'étape 2.

Sinon : il n'existe plus de sommet marqué mais non examiné. Terminer, le flot courant est maximum.

4) Actualisation des flots

• Identifier la chaine augmentant joignant la source s au puits p.

• Actualiser le flot courant le long du chemin augmentant.

• Effacer toute marque et retourner à l'étape 1.

On applique l'algorithme de Ford-Fulkerson sur le réseau $R = (V, E, C)$ de la figure 2.10

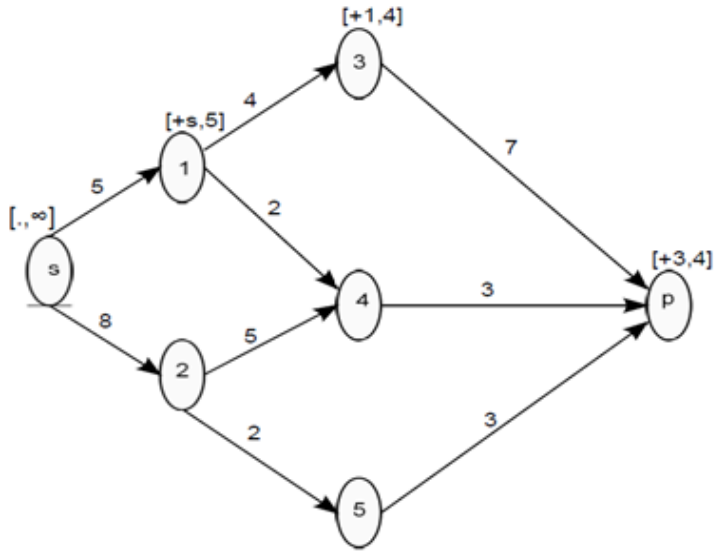


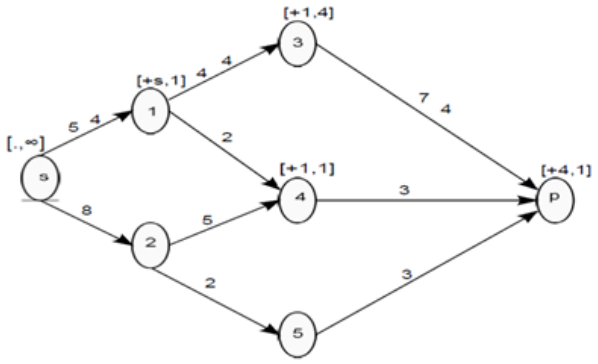
FIGURE 2.10 – Le réseau R

$C = \{s, 1, 3, p\}$

V(f) =4.

La première itération :

Est donnée par la figure suivant.

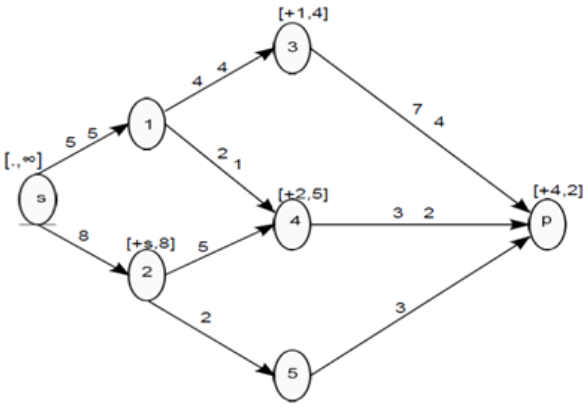


$C = \{s, 1, 4, p\}$

V(f)= 5.

La deuxième itération :

Est donnée par la figure suivant.

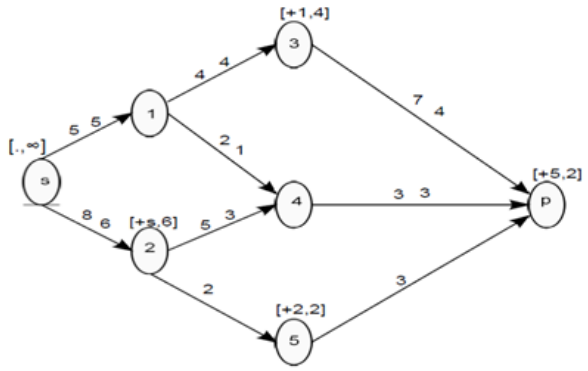


$C = \{s, 2, 4, p\}$

V(f)= 7.

Troisième itération :

Est donnée par la figure suivant.

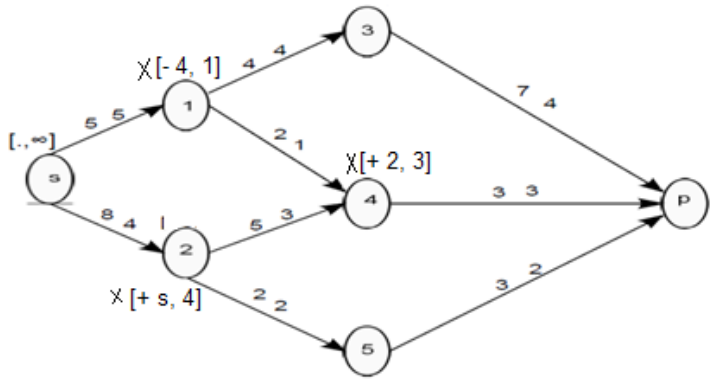


$C = \{s, 2, 5, p\}$

V(f)=9.

Quatrième itération :

Est donnée par la figure suivant.



v(f)=9.

La coupe minimum est $X' = \{s, 1, 2, 4\}$ et la capacité est : $4 + 3 + 2 = 9$

Conclusion

Nous avons vu dans ce chapitre les différents problèmes d'optimisations dans les réseaux, nous allons nous intéresser maintenant aux diverses applications. Ou bien l'implémentation de quelques algorithmes sous le logiciel $DevC^{++}$.

CHAPITRE 3

APPLICATIONS

Il existe plusieurs problèmes pour lesquels on peut utiliser des algorithmes afin d'optimiser une certaine propriété. On va considérer les cas suivants :

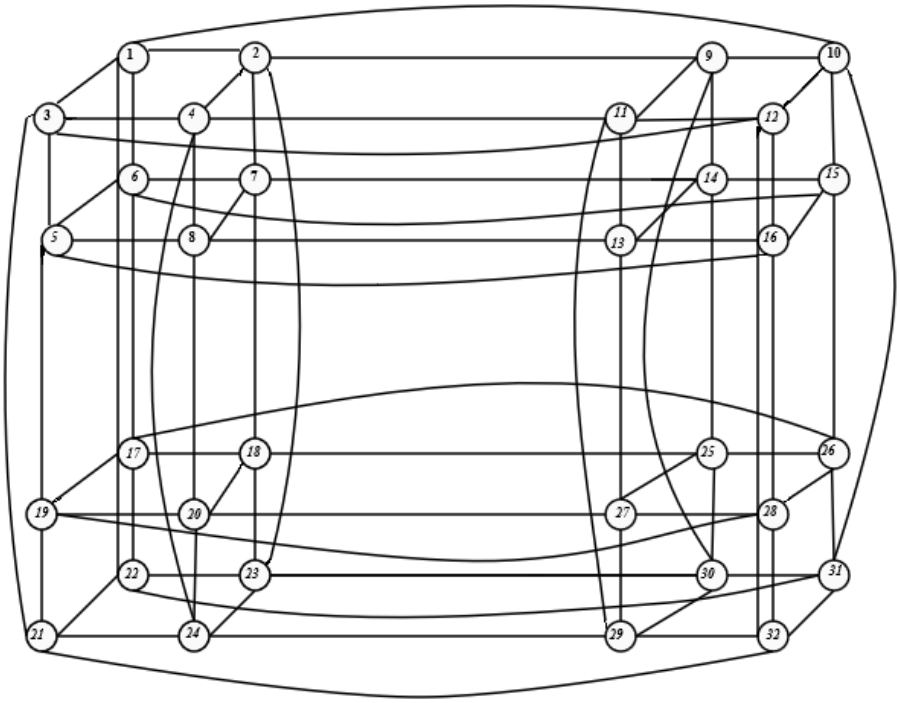
3.1 Réseaux téléphoniques

3.1.1 Présentation de la problématique

Soit une entreprise de télécommunication qui veut tester la confidentialité des services qu'elle offre, c'est-a-dire trouver la probabilité qu'un message ou un appel entre ses abonnés ne soit pas intercepter par d'autres personnes étrangères.

3.1.2 Modélisation du problème sous forme d'un réseau

Pour véhiculer de l'information entre un groupe de personnes, on modélise sous forme d'un graphe G à n sommets, tel que chaque sommet représente une personne et chaque arête représente la communication entre deux personnes.
Les séquences des messages transmis forment un arbre couvrant dans G qui minimise la fonction $(1 - \prod (1 - p_{ij}))$ tel que p_{ij} est la probabilité d'interception d'un message transmis entre x_i et x_j.
Soit un graphe $G = (X, E)$ tel que $X = x_1, x_2, ..., x_n$ et $\{x_i, x_j \in E(G)\}$ si une communication entre x_i et x_j est possible (x_i à x_j et vice versa).



Application de l'algorithme de Prim

Initialisation

On considère l'ordonnancement des arêtes du graphe selon les probabilités croissantes dans le tableau ci-dessous :

i	1	2	3	4	5	6	7	8	9	10
e_i	$(8, 20)$	$(19, 21)$	$(22, 31)$	$(4, 24)$	$(3, 12)$	$(22, 23)$	$(1, 10)$	$(12, 32)$	$(3, 21)$	$(17, 26)$
$p(e_i)$	0.05	0.05	0.06	0.06	0.07	0.07	0.09	0.09	0.09	0.09

i	11	12	13	14	15	16	17	18	19	20
e_i	$(29, 30)$	$(16, 28)$	$(15, 26)$	$(11, 13)$	$(14, 25)$	$(13, 27)$	$(17, 22)$	$(20, 24)$	$(21, 24)$	$(9, 10)$
$p(e_i)$	0.09	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.11

i	21	22	23	24	25	26	27	28	29	30
e_i	$(18, 23)$	$(5, 16)$	$(11, 29)$	$(5, 19)$	$(27, 28)$	$(1, 3)$	$(2, 7)$	$(7, 14)$	$(5, 6)$	$(2, 9)$
$p(e_i)$	0.11	0.12	0.13	0.13	0.13	0.14	0.14	0.15	0.15	0.16

i	31	32	33	34	35	36	37	38	39	40
e_i	$(17, 18)$	$(15, 16)$	$(19, 28)$	$(25, 30)$	$(27, 29)$	$(31, 32)$	$(9, 30)$	$(6, 15)$	$(14, 15)$	$(1, 6)$
$p(e_i)$	0.16	0.17	0.17	0.17	0.17	0.18	0.20	0.20	0.20	0.20

i	41	42	43	44	45	46	47	48	49	50
e_i	$(2, 4)$	$(2, 23)$	$(4, 8)$	$(5, 8)$	$(6, 7)$	$(26, 31)$	$(28, 32)$	$(7, 18)$	$(25, 26)$	$(18, 25)$
$p(e_i)$	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.21	0.21	0.22

i	51	52	53	54	55	66	57	58	59	60
e_i	$(23, 30)$	$(9, 11)$	$(10, 12)$	$(10, 31)$	$(4, 11)$	$(13, 14)$	$(8, 13)$	$(6, 17)$	$(19, 20)$	$(21, 32)$
$p(e_i)$	0.22	0.23	0.23	0.24	0.24	0.25	0.25	0.25	0.25	0.26

i	61	62	63	64	65	66	67	68	169	70
e_i	$(29, 32)$	$(1, 22)$	$(9, 14)$	$(11, 12)$	$(13, 16)$	$(1, 2)$	$(3, 4)$	$(7, 8)$	$(18, 20)$	$(20, 27)$
$p(e_i)$	0.27	0.27	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30

i	71	72	73	74	75	76	77	78	79	80
e_i	$(25, 27)$	$(12, 16)$	$(30, 31)$	$(10, 15)$	$(24, 29)$	$(3, 5)$	$(17, 19)$	$(23, 24)$	$(26, 28)$	$(21, 22)$
$p(e_i)$	0.30	0.31	0.32	0.33	0.34	0.40	0.40	0.40	0.40	0.50

Initialisation

Soit $I = \emptyset$

$NI = \{e_i; i = \{1, ..., 80\}\}$

Commençant par le sommet 8.

Itération 1

On à : $e_1 = (8, 20)$ avec $C(8, 20) = 0.05$ (on choisit l'arrête de faible poids) ;

Soit $I = I \cup \{e_1\} = \{(8, 20)\}$;

Le graphe G n'est pas connexe, on pose alors :

• $I = \{(8, 20)\}$;

• $NI = NI \setminus I \Rightarrow NI = NI \setminus \{(8, 20)\}$;

$|I| = 1$ en continue ; $i = i + 1 = 2$.

Itération 2

On à : $e_{18} = (20, 24)$ avec $C(20, 24) = 0.1$;

Soit $I = I \cup \{e_{18}\} = \{(8, 20); (20, 24)\}$;

Le graphe G n'est pas, on pose connexe alors :

• $I = \{(8, 20); (20, 24)\}$;

• $NI = NI \setminus I \Rightarrow NI = NI \setminus \{(8, 20); (20, 24)\}$;

$|I| = 2$ en continue ; $i = i + 1 = 3$.

Itération 3

On à : $e_4 = (24, 4)$ avec $C(24, 4) = 0.06$;

Soit $I = I \cup \{e_4\} = \{(8, 20); (20, 24); (24, 4)\}$;

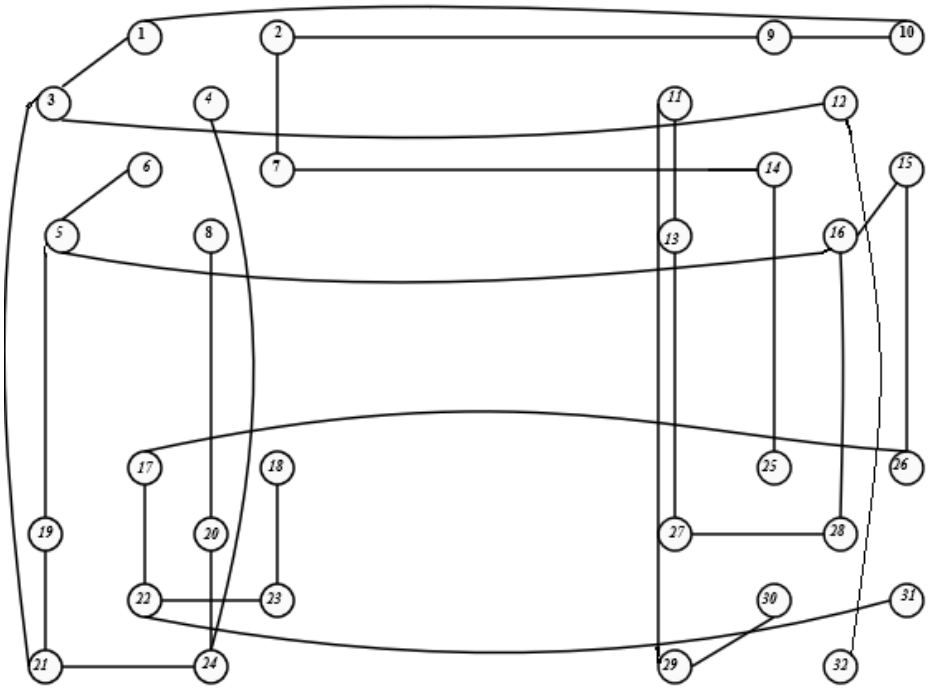
Le graphe G n'est pas, on pose connexe alors :

• $I = \{(8, 20); (20, 24); (24, 4)\}$;

• $NI = NI \setminus I \Rightarrow NI = NI \setminus \{(8, 20); (20, 24); (24, 4)\}$;

$|I| = 3$ en continue ; $i = i + 1 = 4$.

Après l'application de tout les itérations de l'algorithme, on obtient un arbre couvrant minimal est représenté par le graphe $A = (X, I)$, avec $|I| = n - 1 = 32 - 1 = 31$.



L'arbre de poids (probabilités) minimum est représenté par le graphe $A = (X, I)$. La probabilité minimale totale est :

$$P(W) = 1 - \prod_1^{32}(1 - P_{ij})$$

$P(W) = 1 - [(1 - P(8, 20)) \times (1 - P(19, 21)) \times (1 - P(22, 31)) \times (1 - P(4, 24)) \times (1 - P(3, 12)) \times (1 - P(22, 23)) \times (1 - P(1, 10)) \times (1 - P(12, 32)) \times (1 - P(3, 21)) \times (1 - P(17, 26)) \times (1 - P(29, 30)) \times (1 - P(16, 28)) \times (1 - P(15, 26)) \times (1 - P(11, 13)) \times (1 - P(14, 25)) \times (1 - P(13, 27)) \times (1 - P(17, 22)) \times (1 - P(20, 24)) \times (1 - P(21, 24)) \times (1 - P(9, 10)) \times (1 - P(18, 23)) \times (1 - P(5, 16)) \times (1 - P(11, 29)) \times (1 - P(5, 19)) \times (1 - P(27, 28)) \times (1 - P(1, 3)) \times (1 - P(2, 7)) \times (1 - P(7, 14)) \times (1 - P(5, 6)) \times (1 - P(2, 9)) \times (1 - P(17, 18)) \times (1 - P(15, 16))]$

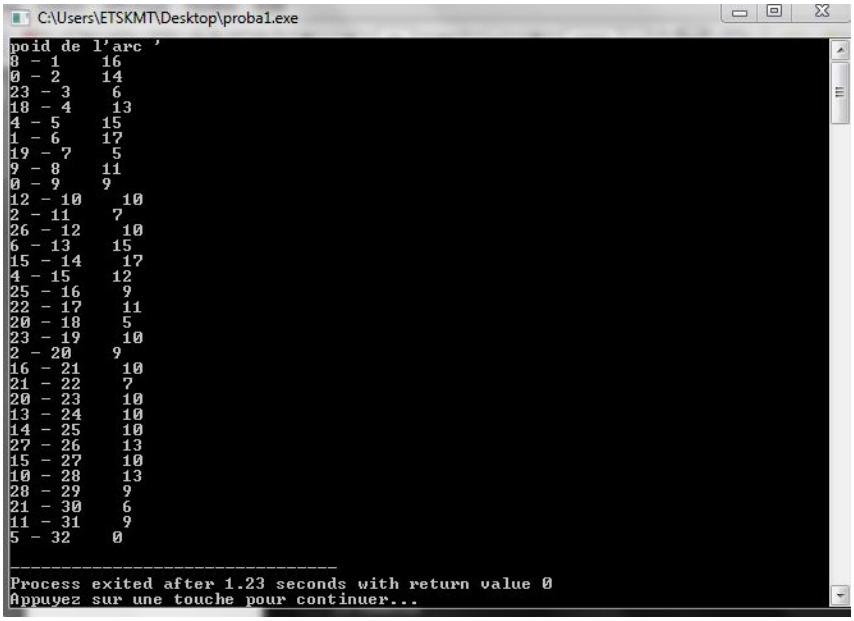
$P(W) = 1 - [(1 - 0.05)^2 \times (1 - 0.06)^2 \times (1 - 0.07)^2 \times (1 - 0.09)^5 \times (1 - 0.10)^8 \times (1 - 0.11)^2 \times (1 - 0.12) \times (1 - 0.13)^3 \times (1 - 0.14)^2 \times (1 - 0.15)^2 \times (1 - 0.16)^2 \times (1 - 0.17)]$

$P(W) = 0.97$

Implémentation de l'algorithme de Prim

L'implémentation de l'algorithme de Prim sous le logiciel $Dev-c^{++}$, pour but de chercher un arbre couvrant de poids minimum.

Exécution de l'algorithme de Prim



Le résultat obtenue est le même (ou bien en à obtenue le même arbre couvrant). Sauf que pour l'implémentation de l'algorithme on à multiplier la capacité de chaque arête par cent et l'initialisation des sommets $e_j, j = 0, ..., 31$.

3.2 Résolution du problème du plus cours chemin dans un réseau routier en Algérie

3.2.1 Plus cours chemin d'une ville à une autre

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

• Une ville représente un sommet ;

• Deux villes ayant des frontières en commun, on relie les sommets représentant les deux villes par une arête ;

• La capacité de chaque arête représente la distance en kilomètre entre deux villes.

A fin de simplifier la recherche du plus cours chemin entre les différentes wilayas d'Algérie on a opter à l'implémentation de l'algorithme de Djikstra sous le logiciel Dev-c++.

3.2.2 Recherche du plus cours chemin entre la wilaya d'Alger et les autres wilayas d'Algérie

D'abord nous allons élaborer un exemple illustratif où on utilise cette application pour déterminer les plus courts chemins menant de la wilaya d'Alger vers toutes les autres wilayas d'Algérie.

∗ Tableau représentatif des différents codes de chaque wilaya :

Code	Wilaya	Code	Wilaya	Code	Wilaya	Code	Wilaya
01	Adrar	13	Tlemcen	25	Constantine	37	Tindouf
02	Chlef	14	Tiaret	26	Medea	38	Tissemsilt
03	Laghouat	15	Tizi ouzou	27	Mostaghanem	39	El-oued
04	Oum elbouaghi	16	Alger	28	M'sila	40	Khenchela
05	batna	17	Djelfa	29	Mascara	41	Souk ahras
06	Bejaia	18	Jijel	30	Ouargla	42	Tipaza
07	Biskra	19	Setif	31	Oran	43	Mila
08	Bechar	20	Saida	32	El bayadh	44	Ain defla
09	Blida	21	Skikda	33	Ilizi	45	Naama
10	Bouira	22	Sidi bel abbes	34	Bordj bou arreridj	46	Ain temouchent
11	Tamanrasset	23	Annaba	35	boumerdes	47	Ghardaia
12	Tebessa	24	Guelma	36	El taref	48	Relizane

∗ La carte géographique de l'Algérie représentant les frontières entre les wilayas :

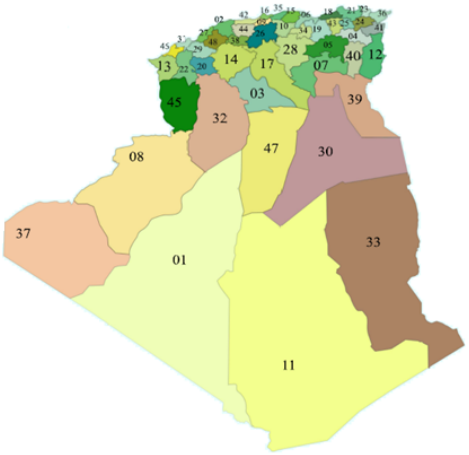


FIGURE 3.1 – Carte indiquant les frontières entre les wilayas

Le graphe induit de la carte routière :

Le graphe suivant représente le graphe induit de la carte routière en représentant les wilayas comme étant des sommets et les routes comme étant des arêtes.

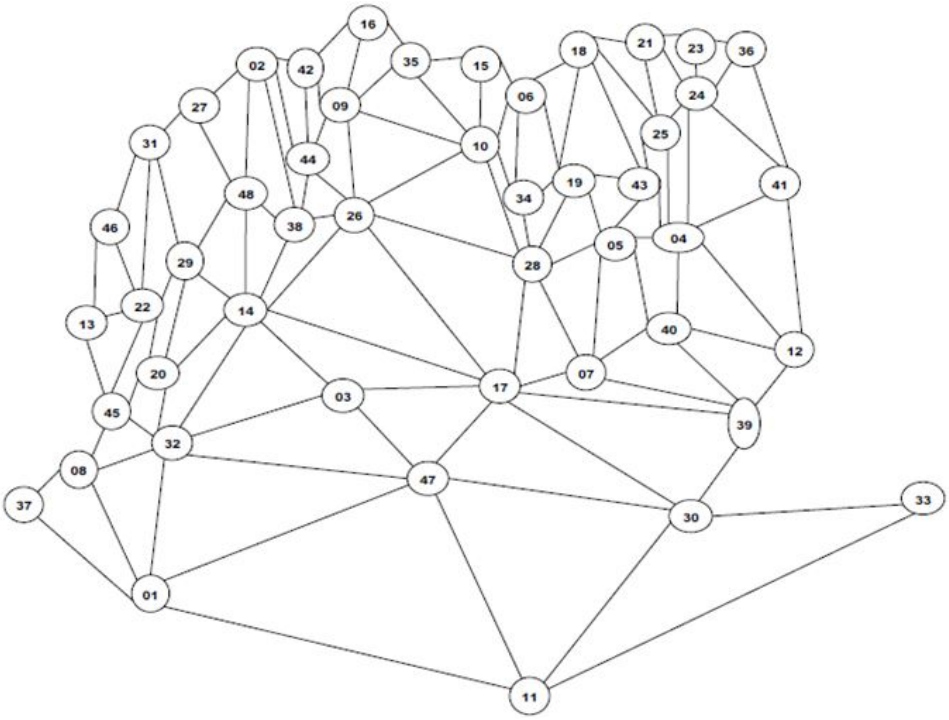


FIGURE 3.2 – Le graphe induit de la carte routière

Le tableau suivant représente la distance entre les wilayas qui ont des frontières en commun (en km) :

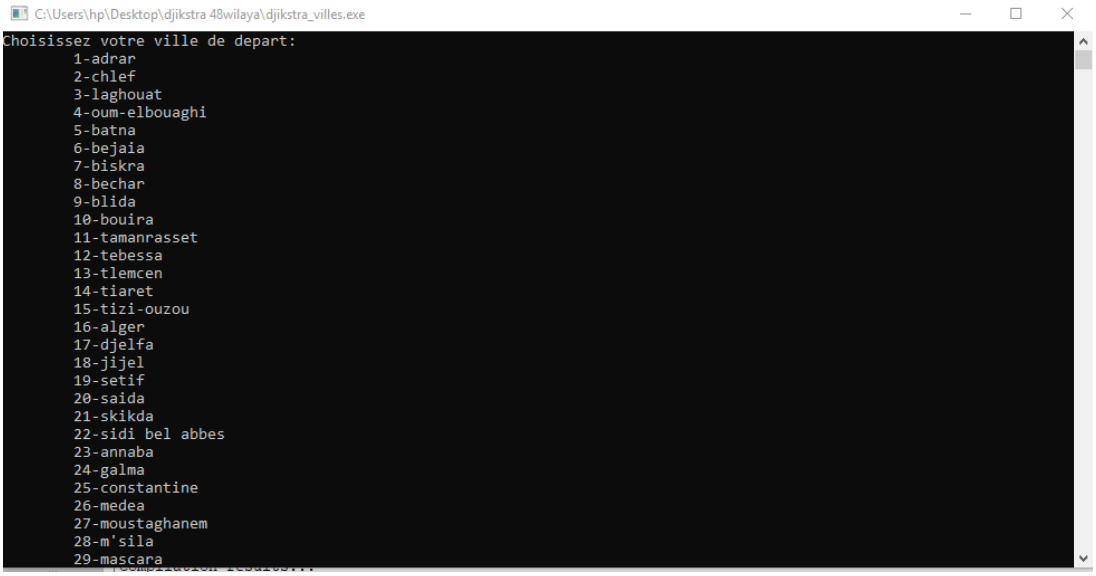
Wilaya	La distance entre les wilayas (voisines)						
(01)	(11) 1042	(47) 944	(37) 1194	(08) 574	(32) 1014	/	/
(02)	(38) 111	(42) 164	(27) 134	(44) 84	(48) 98	/	/
(03)	(32) 277	(17) 217	(14) 321	(47) 191	/	/	/
(04)	(41) 172	(40) 61	(25) 122	(12) 151	(24) 140	(41) 153	(05) 114
(05)	(40) 103	(07) 116	(28) 178	(19) 131	(43) 152	/	/
(06)	(34) 144	(18) 92	(19) 110	(10) 124	(15) 152	/	/
(07)	(40) 190	(39) 225	(28) 180	(17) 273	/	/	/
(08)	(32) 441	(37) 804	(45) 358	/	/	/	/
(09)	(35) 104	(16) 48	(42) 49	(44) 107	(10) 172	(26) 38	/
(10)	(34) 96	(15) 99	(35) 97	(28) 132	(26) 146	/	/
(11)	(47) 1454	(33) 838	(30) 1279	/	/	/	/
(12)	(40) 113	(45) 214	(22) 91	/	/	/	/
(13)	(46) 68	(45) 214	(22) 91	/	/	/	/
(14)	(32) 215	(29) 129	(48) 104	(38) 57	(26) 204	(20) 156	(17) 253

(15)	(35) 38	/	/	/	/	/	/
(16)	(42) 71	(35) 75	/	/	/	/	/
(17)	(30) 545	(47) 341	(39) 469	(28) 179	(26) 220	/	/
(18)	(43) 98	(21) 147	(19) 135	(25) 131	/	/	/
(19)	(43) 98	(28) 126	(34) 71	/	/	/	/
(20)	(32) 199	(45) 246	(22) 97	(29) 96	/	/	/
(21)	(23) 99	(24) 99	(25) 82	(43) 111	/	/	/
(22)	(45) 273	(29) 90	(31) 81	(46) 61	/	/	/
(23)	(24) 68	(36) 70	/	/	/	/	/
(24)	(36) 99	(25) 110	(41) 75	/	/	/	/
(25)	(43) 50	/	/	/	/	/	/
(26)	(38) 149	(44) 94	(28) 231	/	/	/	/
(27)	(31) 83	(48) 60	(29) 79	/	/	/	/
(28)	(34) 58	/	/	/	/	/	/
(29)	(31) 104	(48) 67	/	/	/	/	/

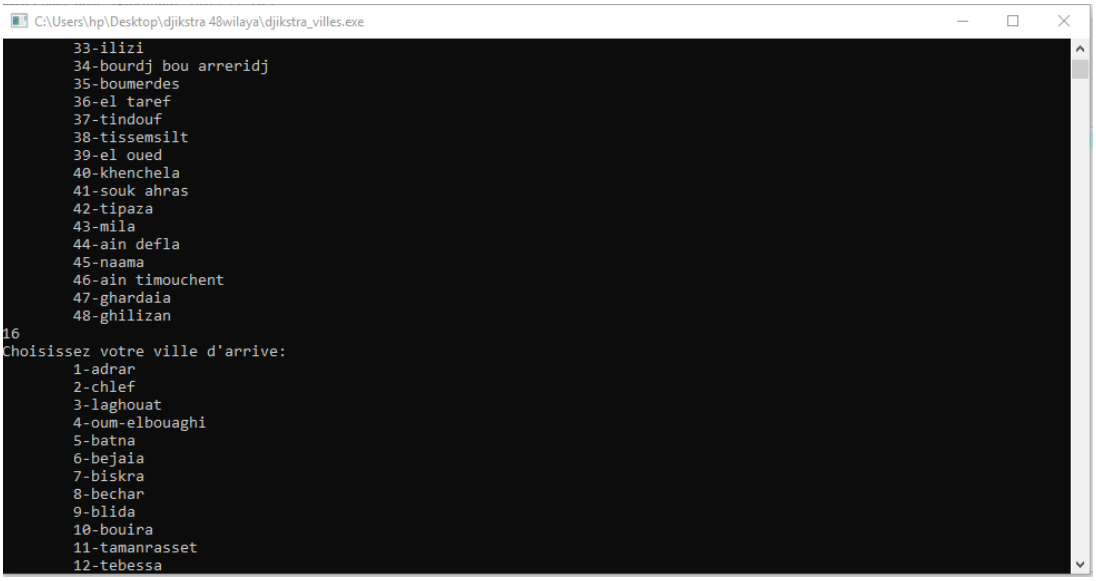
(30)	(47) 288	(39) 260	(33) 1057	/	/	/	/
(31)	(46) 79	/	/	/	/	/	/
(32)	(45) 249	(47) 430	/	/	/	/	/
(36)	(41) 91	/	/	/	/	/	/
(38)	(48) 154	(44) 96	/	/	/	/	/
(39)	(40) 312	/	/	/	/	/	/
(42)	(44) 121	/	/	/	/	/	/

Exécution de l'algorithme de Djikstra

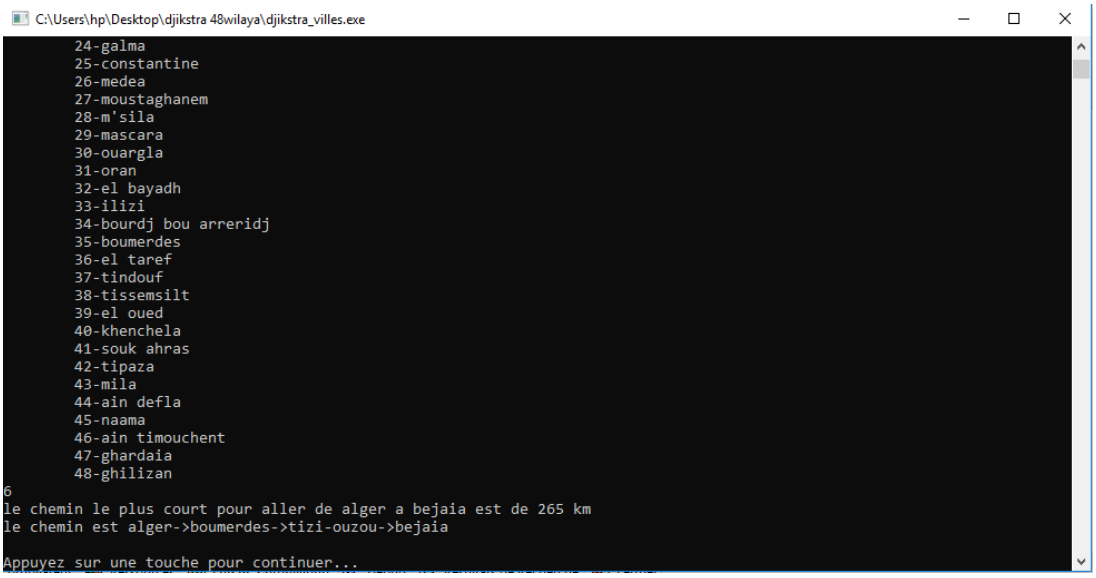
• L'application demande la wilaya de départ choisit : dans ce cas on a choisit la wilaya d'Alger.



• Puis le choix de la wilaya d'arrivé dans ce cas on a choisit la wilaya de bejaia.



Le résultat obtenu est :



Le tableau récapitulatif des plus cours chemins entre la wilaya d'Alger et toutes les autres wilayas :

La destination	Le plus court chemin	La distance (km)
Alger → Adrar	16 → 09 → 26 → 17 → 03 → 47 → 01	1543
Alger → Chlef	16 → 42 → 02	235
Alger → Laghouat	16 → 09 → 26 → 17 → 03	523
Alger → Oum el-bouaghi	16 → 35 → 10 → 34 → 19 → 05 → 04	584
Alger → Batna	16 → 35 → 10 → 34 → 19 → 05	470
Alger → bejaia	16 → 35 → 15 → 06	265
Alger → Biskra	16 → 35 → 10 → 28 → 07	484
Alger → Bechar	16 → 09 → 26 → 14 → 32 → 08	946
Alger → Blida	16 → 09	48
Alger → Bouira	16 → 35 → 10	172
Alger → Tamanrasset	16 → 09 → 26 → 17 → 47 → 11	2001
Alger → Tebessa	16 → 35 → 10 → 34 → 19 → 05 → 40 → 12	686
Alger → Tlemcen	16 → 42 → 02 → 48 → 29 → 22 → 13	581
Alger → Tiaret	16 → 09 → 26 → 14	290
Alger → Tizi-ouzou	16 → 35 → 15	113
Alger → Djelfa	16 → 09 → 26 → 17	306
Alger → Jijel	16 → 35 → 15 → 06 → 18	357
Alger → Setif	16 → 35 → 10 → 34 → 19	339
Alger → Saida	16 → 42 → 02 → 48 → 29 → 20	476
Alger → Skikda	16 → 35 → 15 → 06 → 18 → 21	504

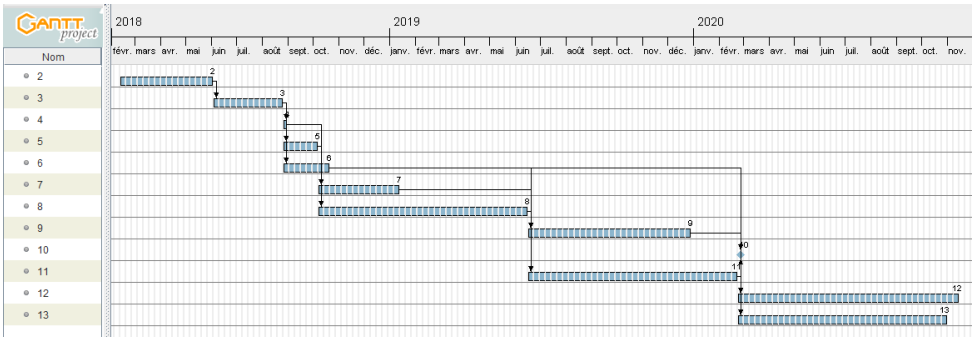
Alger → Sidi bel-abbes	16 → 42 → 48 → 29 → 22	490
Alger → Annaba	16 → 35 → 15 → 06 → 18 → 21 → 23	603
Alger → Guelma	16 → 35 → 15 → 06 → 18 → 25 → 24	598
Alger → Constantine	16 → 35 → 15 → 06 → 18 → 25	488
Alger → Medea	16 → 09 → 26	86
Alger → Mostaghanem	16 → 42 → 02 → 27	369
Alger → M'sila	16 → 35 → 10 → 28	304
Alger → Mascara	16 → 42 → 02 → 48 → 29	400
Alger → Ouargla	16 → 09 → 26 → 17 → 47 → 30	935
Alger → Oran	16 → 42 → 02 → 27 → 31	452
Alger → El-bayadh	16 → 09 → 26 → 14 → 32	505
Alger → Ilizi	16 → 09 → 26 → 17 → 47 → 30 → 33	1992
Alger → B.B.arreridj	16 → 35 → 10 → 34	268
Alger → Boumerdes	16 → 35	75
Alger → El-taref	16 → 35 → 15 → 06 → 18 → 21 → 23 → 36	673
Alger → Tindouf	16 → 09 → 26 → 14 → 32 → 08 → 37	1750
Alger → Tissemsilt	16 → 09 → 26 → 38	235
Alger → El-oued	16 → 35 → 10 → 28 → 07 → 39	709
Alger → Khenchela	16 → 35 → 10 → 34 → 19 → 05 → 40	573
Alger → Souk ahras	16 → 35 → 15 → 06 → 18 → 25 → 24 → 41	673
Alger → Tipaza	16 → 42	71
Alger → Mila	16 → 35 → 10 → 34 → 19 → 43	452
Alger → Ain defla	16 → 09 → 44	155
Alger → Naama	16 → 42 → 02 → 48 → 29 → 20 → 45	722
Alger → Ain defla	16 → 42 → 02 → 27 → 31 → 46	531
Alger → Ghardaia	16 → 09 → 26 → 17 → 47	647
Alger → Ghelizane	16 → 42 → 02 → 48	333

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

La mise en exploitation d'un nouveau gisement minier demande la réalisation d'un certain nombre de tâches. Le tableau suivant représente ces différentes tâches avec leurs relations d'antériorité.

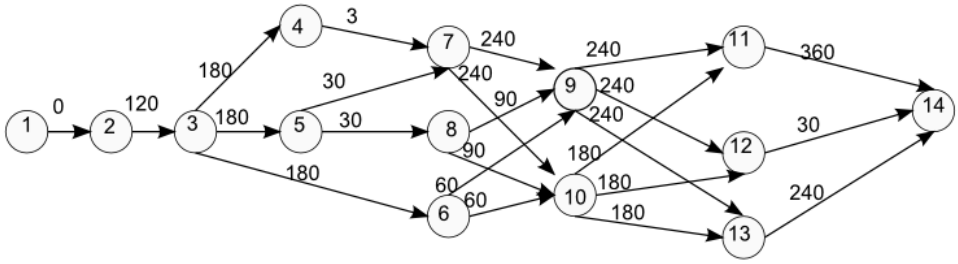
Tâche	Description	Durées (jours)	Tâches anterieurs
2	obtention d'un permis d'exploitation	120	–
3	établissement d'une piste de 6 km	180	2
4	transport et installation à pied d'œuvre de 2 sondeuses	3	3
5	création de bâtiments provisoires pour le bureau des plans, le logement des ouvriers sondeurs	30	3
6	goudronnage de la piste	60	3
7	adduction d'eau	90	5
8	campagne de sondage	240	$4, 5$
9	forage et équipement de trois puits	180	$6, 7, 8$
10	transport et installation au fond du matériel d'exploitation	30	$11, 9$
11	construction de bureaux et logements, ouvriers et ingénieurs	240	$6, 7, 8$
12	traçage et aménagement du fond	360	$11, 9$
13	construction d'une laverie	240	$11, 9$

Le diagramme de gant associé au tableau est donné par la figure suivante :



On peut modéliser le tableau précédent sous forme d'un réseau Pert potentiels tâches, tel que les tâches sont représentés par des sommets et les contraintes de potentiels sont représentées par des arcs.

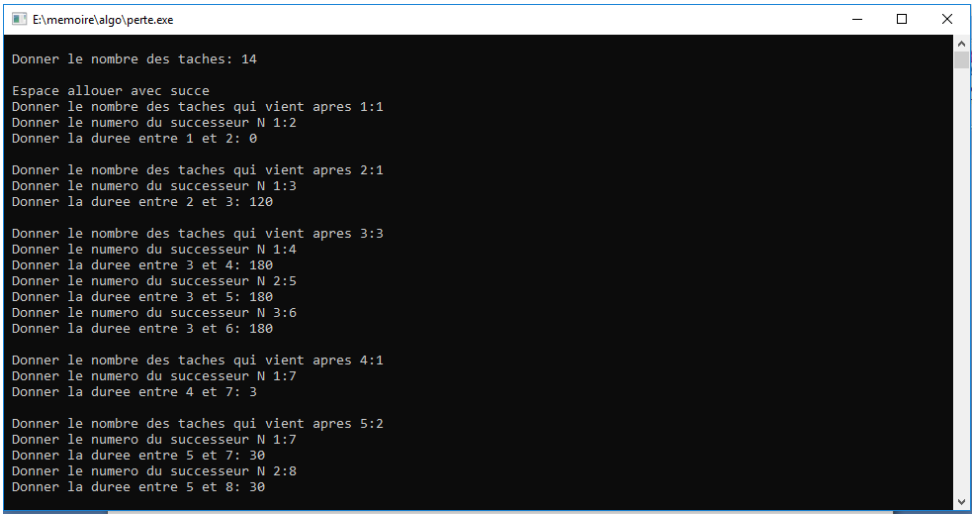
Le graphe potentiels-tâches correspondant est donné par la figure suivante :



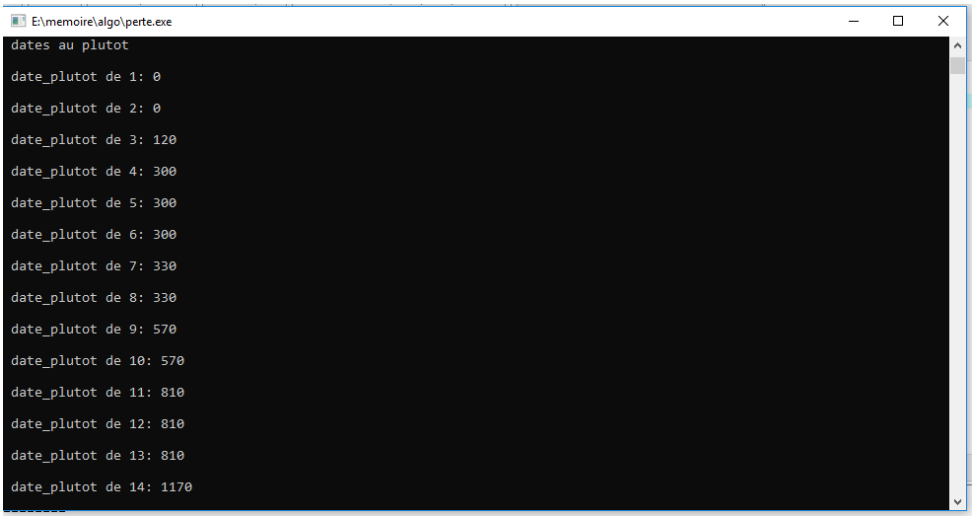
Les tâches 1 et 14 sont respectivement début et fin du projet.

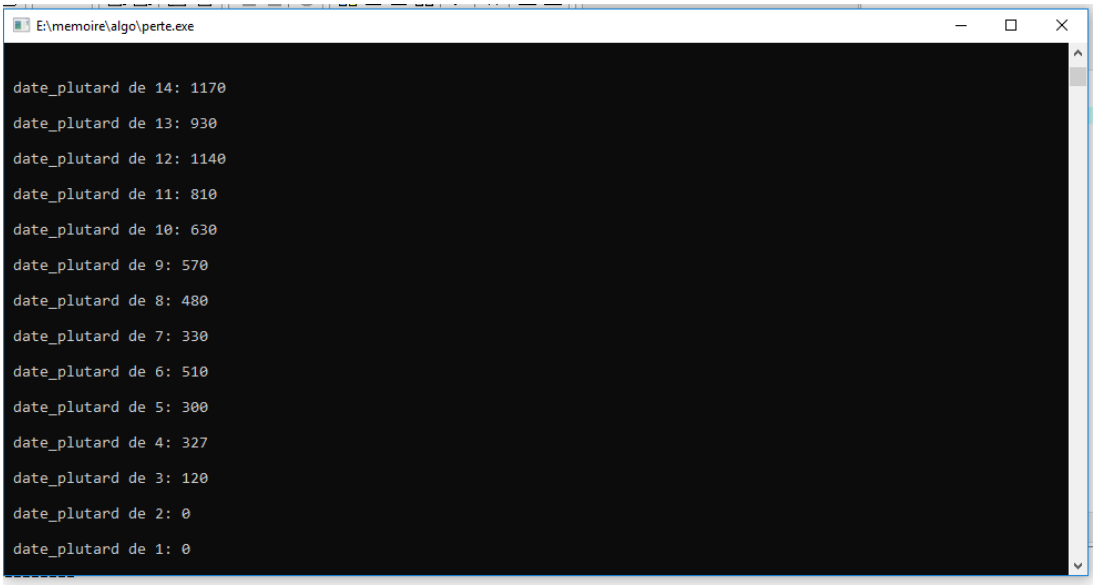
A fin de simplifier les calculs des dates au plutôt et au plutard des tâches ainsi que les marges et le chemin critique on a opté à l'implémentation de la méthode Pert sous le logiciel $Dev - c^{++}$.

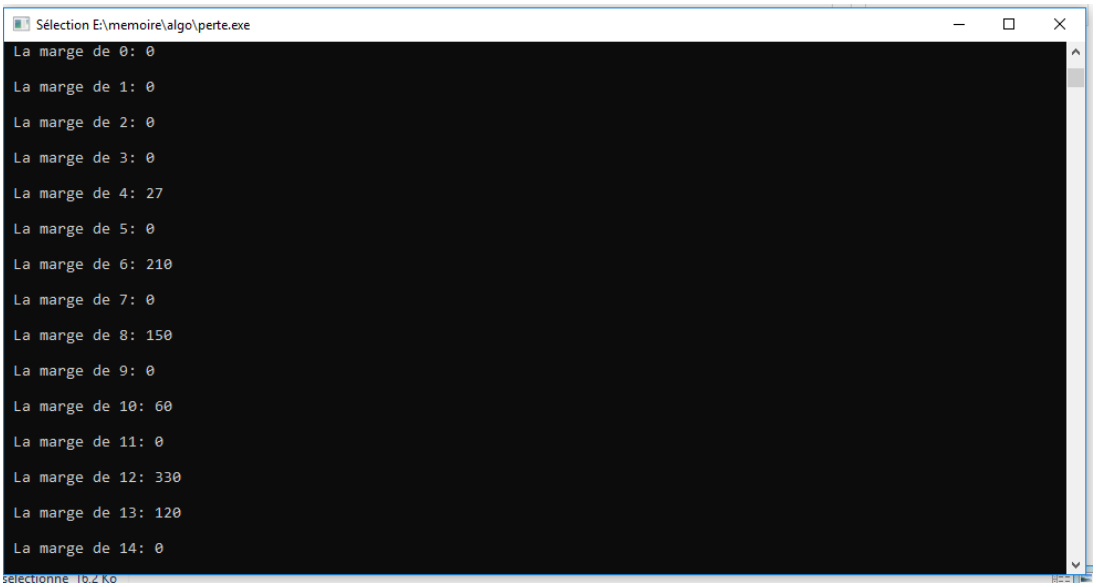
La figure suivante représente l'insertion des données.

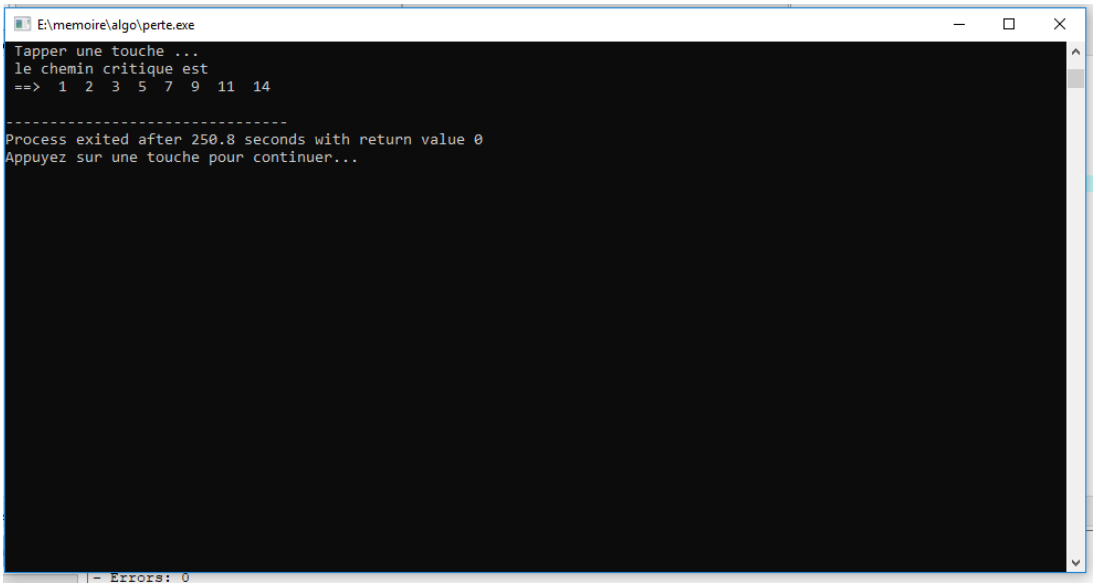


Les résultats obtenus sont les suivants :




E:\memoire\algo\perte.exe
date_plutard de 14: 1170
date_plutard de 13: 930
date_plutard de 12: 1140
date_plutard de 11: 810
date_plutard de 10: 630
date_plutard de 9: 570
date_plutard de 8: 480
date_plutard de 7: 330
date_plutard de 6: 510
date_plutard de 5: 300
date_plutard de 4: 327
date_plutard de 3: 120
date_plutard de 2: 0
date_plutard de 1: 0


Sélection E:\memoire\algo\perte.exe
La marge de 0: 0
La marge de 1: 0
La marge de 2: 0
La marge de 3: 0
La marge de 4: 27
La marge de 5: 0
La marge de 6: 210
La marge de 7: 0
La marge de 8: 150
La marge de 9: 0
La marge de 10: 60
La marge de 11: 0
La marge de 12: 330
La marge de 13: 120
La marge de 14: 0



Conclusion

Ce chapitre était consacré sur l'application et l'implémentation de quelques problèmes d'optimisation dans les réseaux. Nous avons présenté l'un des problèmes du réseaux téléphonique, réseaux routier et problème d'ordonnancement.

CONCLUSION GÉNÉRALE

Dans ce travail, nous nous sommes intéressés au problème d'optimisation dans les réseaux en utilisant les graphes valués.
On a pu constater que les graphes constituent une méthode de pensée qui permet de modéliser une grande variété de problèmes concrets en se ramenant à l'étude de sommets et d'arcs.
La théorie des graphes permet de générer des circuits optimisés et de gérer des réseaux (routiers, de communication, de transport, d'eau ...), d'ordonnancer des tâches et de gérer des planning. Elle est la clé de l'intelligence artificielle avec la notion du "plus court chemin".
Ces nombreuses applications font de la théorie des graphes un outils appréciable d'aide à la décision (en recherche opérationnelle), en apparence, sa mise en œuvre est simple et ludique, voire enfantine.

L'exactitude de notre objectif consiste à résoudre des problèmes en utilisant l'optimisation en théorie des graphes en particulier les réseaux, pour cela on a traité les différents problèmes suivants :
Le premier consiste à résoudre un problème de télécommunication en prenant compte la fiabilité de communication en utilisant l'algorithme de PRIM (recherche d'un arbre couvrant qui minimise la probabilité de l'interception) Le second consiste à la mise en exploitation d'un nouveau gisement minier en minimisant le délai des travaux et cela est fait en utilisant la méthode PERT.
Le dernier problème consiste à résoudre un problème de recherche du plus court chemin

entre les wilayas d'Algérie de tel sorte qu'elle soit aussi conviviale que possible, et cela en utilisant l'algorithme de Djikstra.
Pour trouver des meilleurs résultats on a implémenté ses algorithmes et méthodes sous le langage $Dev - c^{++}$.

BIBLIOGRAPHIE

[1] C. Berge, Livre "graphe et hypergraphe". deuxiéme ed, 1973.

[2] H. Mélot, Thèse "Journée des mathématiques et des sciences", in Les graphes au quotidien, (6, Av. Du Champ de Mars, 7000 Mons), Université de Mons-Hainaut, mars 2004.

[3] Livre " Les graphes Stochastique-Dynamique". La cote 003/25. Université de Béjaia.

[4] Louis Esperet, Thèse " Coloration De graphes", ENS Lyon-2010.

[5] L. X. Jean-Manuel Mény, Gilles Aldon, Livre "Introduction à la théorie des graphes". 2005.

[6] M. SAKAROVITCH. Livre " Optimisation Combinatoire". Tome 1 : Graphe et programmation linéaire Coll Enseignement des science. Hermann. Paris. 1984

[7] $\mathrm{M}^{r}.Taouinat.CoursThéoriedesgraphesavancée.UniversitéA.MiraBejaia$, 2014/2015.

[8] N. Akkouchee M. Bentoubache. Thèse " Application de gestion de projet a ressource limite au demain de la construction bâtiment". A. Mira Béjaia. 2002.

[9] N. Bellharat, Livre " La théoré des graphes", 2010

[10] O. C. Et Claudine Robert, Livre " Theorie des graphes". 2003.

[11] R. Karp, Livre " eruducibility among combinatorial problems", Complexity of computer computation 85, 1972.