République Algérienne Démocratique et Populaire
Ministère de l'enseignement Supérieur et de la Recherche Scientifique
Université A/Mira de Béjaïa
Faculté des Sciences Exactes
Département recherche opérationnelle

Mémoire De Fin de cycle

En vue d'obtention du diplôme de master en recherche opérationnelle spécialité : MMTD

Thème

Algorithmes d'optimisation dans les graphes valués et Applications

Réalisé par :

- Melle TOUMI Kamelia
- Mr KABACHE Mounir

Soutenu devant le jury composé de :

Président	**Mr TAOUINET Smail**	**U. A/Mira Béjaia.**
Examinateur	**Mr TALEM Djamel**	**U. A/Mira Béjaia.**
Examinatrice	M^{eme} **YOUNSI Leila**	**U. A/Mira Béjaia.**
Encadreur	**Mr KABYL Kamal**	**U. A/Mira Béjaia.**

2020/2021

Remerciements

On remercie dieu le tout puissant de nous avoir donné la santé et la volonté d'entamer et de terminer ce mémoire.

Et on tient à remercier vivement **Mr KABYL Kamal** pour nous avoir honorés par son encadrement, pour sa gentillesse et patience, son orientation et précieux conseils.

Et on tient à exprimer notre reconnaissance à tous les enseignants du département de Recherche Opérationnelle qui nous ont tellement aidés durant notre cursus universitaire.

On remercie aussi les membres de jury d'avoir accepté d'évaluer notre travaille.

Enfin, dans le souci de n'oublier personne. On tient à remercier toute personne ayant partagé ou aidé de prés ou de loin à la réalisation de ce modeste travail.

Dédicaces

A ma chére maman et mon adorable papa.

A mes sœurs et mon frère.

A tous mes chers amis.

A ma familles sans exception.

A mes professeurs et tous ceux qui me sont chers.

Je dédie ce travail avec reconnaissance et amour.

Kamelia.

Dédicaces

Je dédie ce modeste travail

A ma maman qui ma soutenu et encouragé durant ces années d'études,

Qu'elle trouve ici le témoignage de ma profonde reconnaissance.

A mon pére pour son amour et ses sacrifices.

A mes adorables sœurs et frère, source d'espoir et de motivation.

A ma binôme, à qui je souhaite bonne chance pour son prochain projet.

A tous ceux que j'aime.

Mounir.

Table des matières

Table des figures

Introduction générale

La recherche opérationnelle peut être définie comme l'ensemble des méthodes et des techniques utilisables pour élaborer de meilleures décisions. On trouve également la théorie des graphes qui est un outil très puissant pour résoudre de nombreux problèmes concrets, on est amené à tracer des dessins qui représentent le problème à résoudre. Bien souvent, ces petits dessins se composent de points et de lignes continues reliant deux à deux certains de ces points. On appellera ces petits dessins des graphes, les points des sommets et les lignes des arcs ou arêtes.
Les graphes permettent de modéliser et manipuler plus facilement des problèmes de cas réel avec une représentation graphique, parmi les problèmes les plus anciens de la théorie des graphes et les plus importants par leurs application on a : problème de cheminement, problème de l'arbre couvrant de poids minimum problèmes de flots,problème d'ordonnancement.... etc.
Nous nous intéressons dans notre travail à la présentation ainsi que la programmation de quelques algorithmes d'optimisation dans les graphes values pour résoudre des problèmes concret. Notre travail est réparti en quatre chapitres :

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

• Le deuxième chapitre consiste à la présentation des problèmes d'optimisation et les algorithmes de résolution.

• Dans le troisième chapitre nous avons abordé quelques problèmes d'optimisation pour lesquels on a appliqué quelques algorithmes de résolution.

• Enfin, dans le dernier chapitre on a présenté le langage de programmation C++ sur lequel on a appliqué les algorithmes tout en illustrant les différentes étapes d'exécution.

On terminera par une conclusion qui clôturera notre travail.

Chapitre 1

Quelques Notions et Notations de Bases sur la théorie des graphes

1.1 Introduction

La théorie des graphes est un outil puissant de modélisation et de résolution de problèmes concrets. A l'origine, la théorie des graphes était présentée comme une curiosité mathématique ; Euler lors d'une de ses promenades nocturnes a voulu tracer un itinéraire circulaire dans la ville de Konigsberg. Partant d'un point donné, il voulait visiter les sept ponts de cette ville (disposés selon le schéma ci-dessous) une seule fois seulement, puis retourner à son point de départ. Les

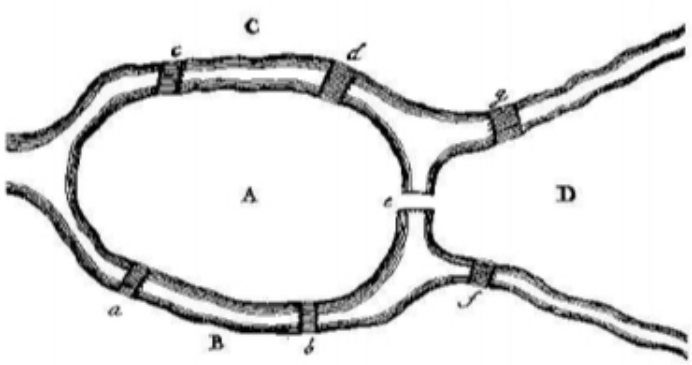


FIGURE 1.1 – les sept ponts de Konigsbeg

points A, B, C et D sont des rives. Ensuite la théorie des graphes a été utilisée pour modéliser des circuits électriques (Kirch-hoff), puis de nombreuses applications dans différents domaines tels : la chimie, la psychologie,etc [1].

1.2 Généralités sur les graphes

1.2.1 Graphe

Les graphes sont des concepts mathématiques utilisés pour modéliser des relations binaires entre des objets d'un même ensemble. Ils sont fréquemment utilisés pour modéliser des systèmes qui se présentent sous la forme d'un réseau. Il existe deux types de graphes : les graphes orientés et les graphes non orientés [1].

1.2.2 Graphe orienté

Un graphe orienté est un couple (X ;U), ou X est l'ensemble des sommets du graphe et U, l'ensemble de ses arcs. X et U sont finis. Un arc est une relation entre deux sommets , dotée d'une orientation :

Si u = (x ;y) est un arc de U alors avec $x, y \in X$, la relation est orientée de x vers y.
Le graphe G est noté G =(X ;U) [1].
La figure 1.3 montre un graphe orienté à 5 sommets et 9 arcs.

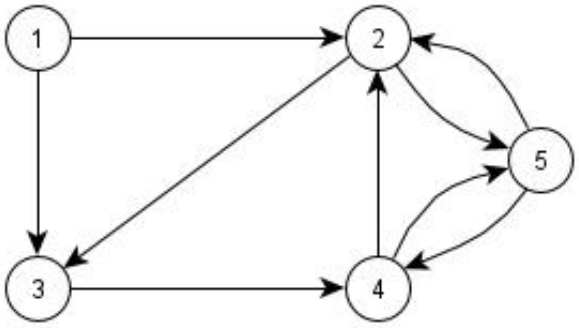


FIGURE 1.2 – Graphe orienté à 5 sommets et 9 arcs

1.2.3 Graphe non orienté

Un graphe non orienté G est un couple (X ;E), ou X est un ensemble dont les éléments sont appelés sommets et E un sous-ensemble de parties de X contenant des arêtes.
La figure 1.4 illustre un graphe non orienté à 5 sommets et 6 arêtes.

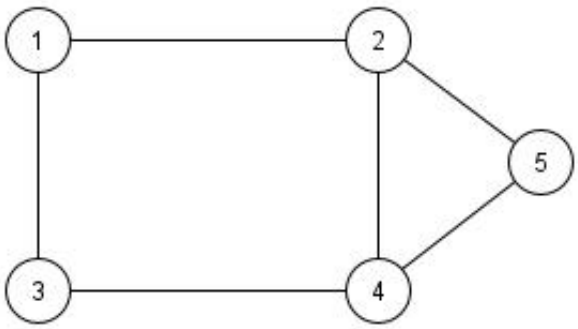


FIGURE 1.3 – Graphe non orienté à 5 sommets et 6 arêtes

1.2.4 Boucle

Une boucle est une arête dont son extrémité initiale et son extrémité terminal coïncident.

1.2.5 Sommets adjacents

On dit que deux sommets, x et y, sont adjacent si et seulement s'il existe une arête les reliant.

1.2.6 Les voisins d'un sommet

Les voisins d'un sommet x d'un graphe G sont les sommets qui admettent une arête ou un (arc) avec x.

Dans le graphe de La figure 1.2 les voisins de x_5 sont x_1, x_4 et x_6.

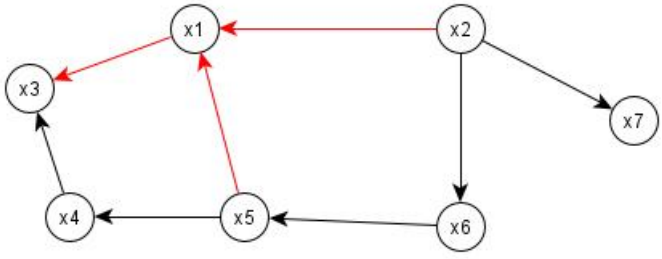


FIGURE 1.4 – Graphe orienté à 7 sommets et 8 arcs

1.2.7 Graphe simple

Un graphe G est dit simple si tous ses sommets sont sans boucles et entre chaque paire de sommets, il y a au plus une arête.

1.2.8 Graphe Multiple

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

1.2.9 Sous-graphe engendré

Un sous graphe est un graphe contenu dans un autre graphe, L'ensemble des sommets du sous-graphe H est un sous-ensemble A de l'ensemble des sommets de G et l'ensemble des arêtes de H est un sous-ensemble E'_A de l'ensemble des arêtes de G dont les extrémités sont dans A, la figure 1.5 montre un sous graphe de graphe de la figure 1.4.

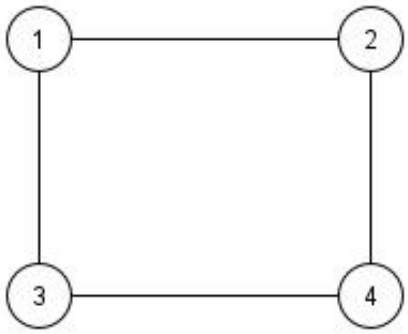


FIGURE 1.5 – Sous graphe

1.2.10 Graphe partiel

Soit G=(X,E) un graphe. Le graphe G'=(X,E') est un graphe partiel de G, si E'$\subset$ E. Autrement dit, on obtient G' en enlevant une ou plusieurs arêtes au graphe G.

1.2.11 Sous graphe partiel

Pour un sous-ensemble de sommets A inclus dans X, le sous-graphe de G induit par A est le graphe G=(A,E(A)) dont l'ensemble des sommets est A et l'ensemble des arêtes E(A) est formé de toutes les arêtes de G ayant leurs deux extrémités dans A. Autrement dit, On obtient G' en enlevant un ou plusieurs sommets au graphe G, ainsi que toutes les arêtes incidentes à ces sommets.

1.2.12 Graphe pondéré

un graphe pondéré est un graphe auquel on adjoint une fonction de valuation. Un graphe peut être pondéré/valué sur ses sommets comme sur ses arêtes. On note p(x) (resp,p(i)) le poids dun sommet x (resp, i) et p(x,x_0) (resp,p(i,j)) le poids d'une arête (x, x_0) (resp,(i,j)).

1.2.13 Extrémité initiale et terminale (successeur et prédécesseur)

Soit un arc (i ,j) ; i est dit extrémité initiale de (i ,j)et j extrémité terminale de (i ,j). On dit aussi que j est un successeur de i, et i un prédécesseur de j. L'arc (i ,j) est dit incident vers l'extérieur en i et vers l'intérieur en j.

Définition Un arc de graphe orienté G de la forme (i ,i) est appelé une boucle. Pour un arc u=(i ,j), le point i est son extrémité initiale, et le point j son extrémité terminale.

On dit que j est un successeur de i s'il existe un arc ayant son extrémité initiale en i et son extrémité terminale en j. l'ensemble des successeurs de i se note Γ_G^+(i)

De même, on dit j est un prédécesseur de i s'il existe un arc de la forme (j,i). L'ensemble des prédécesseurs de i se note Γ_G^-(i)

L'ensemble des sommets voisins de i se note Γ_G(i)= Γ_G^+(i) $\cup$ Γ_G^-(i)

On note Γ^+(i) l'ensemble des successeurs de i, Γ^-(i) l'ensemble des prédécesseurs de i.

1.2.14 Arcs adjacents, arêtes adjacentes

Deux arcs (resp.arêtes) sont adjacents (resp.adjacentes) en un sommet x, si x est une extrémité commune des deux arcs (resp.arêtes).

1.3 Chaine, Chemin, Cycle et Circuit

1.3.1 Chaine

- Une chaine est une suite finie de sommets reliés deux à deux par une arête. - Une chaine simple est une chaine qui n'utilise pas deux fois la même arête.
- Une chaine élémentaire est une chaîne n'utilise pas deux fois le même sommets.

1.3.2 Chemin

Dans le graphe G, un chemin menant x_0 à x_k est une suite de sommets reliés successivement par des arcs orientés dans le même sens, que l'on note $(x_0, x_1, ..., x_k)$, - Un chemin simple est un chemin qui passe une seule fois par ses arcs.
- Un chemin élémentaire est un chemin qui passe une et une seule fois par ses sommets.

1.3.3 Cycle

Un cycle est une chaîne dont les extrémités coïncident.

1.3.4 Circuit

Un circuit est un chemin dont les extrémités sont confondues.

1.4 Connexité et forte connexité

Dans cette section on s'intéresse ou graphe simple.

1.4.1 Graphe connexe

Soit le graphe G=(X,U). On dit que G est connexe si et seulement si $\forall$ x,y $\in$ X, il existe une chaine reliant x à y. La figure 1.10 montre un graphe connexe.

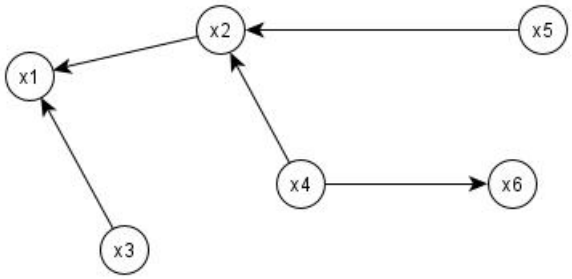


FIGURE 1.6 – Graphe connexe

1.4.2 Composante connexe

Une composante connexe notée CC est l'ensemble de sommets qui ont deux à deux une relation de connexité, et tout sommet en dehors de la composante n'a pas de relation de connexité avec les sommets de cette composante.

1.4.3 Graphe fortement connexe

Un graphe orienté G=(X,U) est dit fortement connexe si $\forall$ { x,y } $\in$ X, il existe un chemin de x à y et un autre chemin de y à x.

1.4.4 Composante fortement connexe

Une composante fortement connexe cfc est l'ensemble de sommets qui ont deux à deux la relation de forte connexité, de plus tout sommet en dehors de la composante n'a pas de relation de forte connexité avec aucun élément de cette composante.
la figure 1.11 montre un graphe à deux composantes fortement connexes.

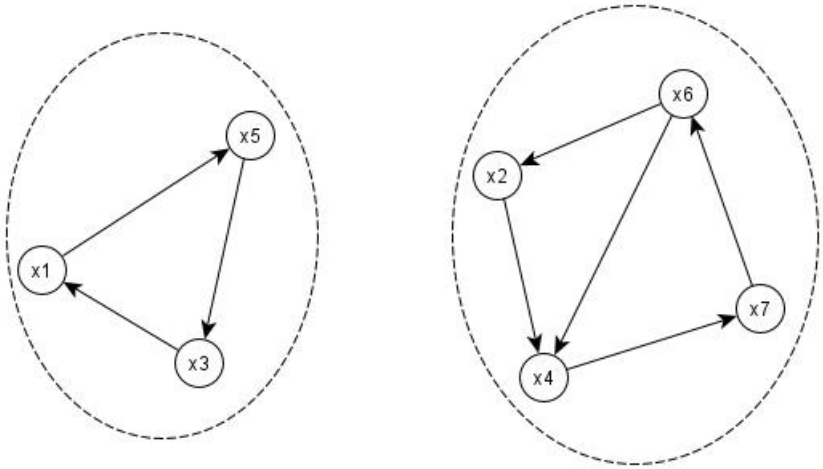


FIGURE 1.7 – Graphe à deux Composantes fortement connexes

Comme algorithme de recherche de nombre de cfc, nous avons l'algorithme de marquage suivant :

. Algorithme (Algorithme de marquage)

- Répéter

1) partir d'un sommet quelconque n'appartenant pas à une cfc, le marquer par ±.

2) Sur l'ensemble des sommets non marqués par ± et tant qu'on peut marquer un sommet faire :

(a) Marquer par + tout sommet suivant (successeur) d'un sommet marqué par +.

(b) Marquer par - tout sommet précédent (prédécesseur) d'un sommet marqué par -.

Tout sommet marqué par ± appartient à une cfc.

- Jusqu'à ce que tout sommet appartienne à une cfc.

1.4.5 Un réseau

Un réseau est un graphe orienté R = (X,U) avec une valuation positive de ses arcs. La valuation d'un arc (x, y), notée c(x, y), est appelée la capacité de l'arc.

On distingue sur un réseau R deux sommets particuliers : un sommet dit source s et un autre dit destination t.

Les arcs de capacité nulle ne sont pas représentés sur le réseau.[3]

La figure 1.13 montre un réseau à 8 sommets.

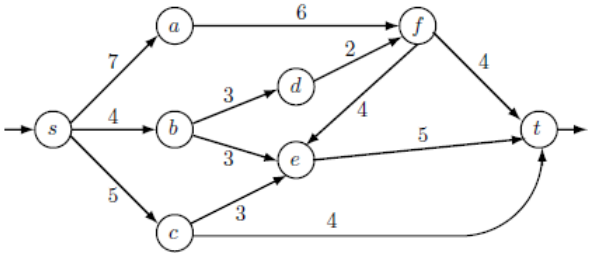


FIGURE 1.8 – Un réseau à 8 sommets

1.4.6 Un Flux

Un flux est la quantité ϕ_{ij} sur chaque arc (i, j) peut-être assimilée par exemple à la quantité d'électricité (ou de "véhicules") parcourant l'arc.

1.4.7 Flot

Un flot φ est déterminé par la donnée du flux pour tout arc du réseau. La valeur d'un flot V (φ') est par définition, la somme des flux partant de la source x_1(V (φ') est aussi égale à la somme des flux des arcs arrivant sur le puits x_p). La loi de Kirchoff (loi de conservation aux noeuds) est dite le flot entrant égale au flot sortant.

$\sum_{\varphi\in\omega^+} \nu(\varphi) = \sum_{\varphi\in\omega^-} \nu(\varphi) \forall i \in \omega$ /s,t arc

w^+(i) est l'ensemble des arcs sortant en i.

w^-(i) est l'ensemble des arcs entrant en i.

1.5 Quelques graphes particuliers

1.5.1 Graphe Complet

Un graphe G est dit complet si tous les sommets de G sont deux à deux adjacents. La figure 1.14 montre un graphe complet à 5 sommets et 10 arêtes[4].

Un graphe complet à n sommets et noté par k_n

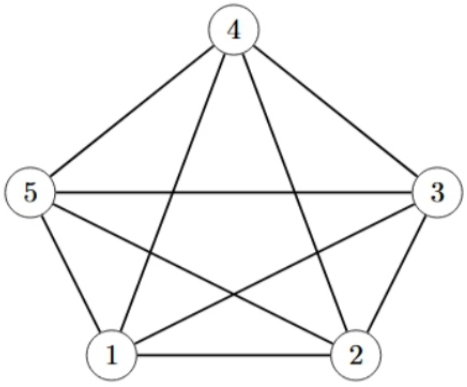


FIGURE 1.9 – Graphe complet

1.5.2 Graphe planaire

Un graphe G est dit planaire s'il est possible de le représenter sur un plan de sorte que les sommets soient des points distincts et les arêtes des courbes simples ne se rencontrant pas en dehors de leurs extrémités. Dans un graphe planaire, une face est par définition une partie du plan limitée par des arêtes de sorte que deux points de la face puissent toujours être reliés par un trait continu ne rencontrant ni sommet ni arête. Deux faces sont adjacentes si elles ont une arête en commun. Deux faces sont opposées si elles ont un sommet commun sans être adjacentes.[5]
On appelle dual d'un graphe planaire — appelé primal — le graphe obtenu de la façon suivante :
- dans toute face du primal on dessine un sommet du dual.
- pour toute arête séparant deux faces du primal, on dessine une arête joignant les deux sommets correspondants du dual (et qui traverse l'arête correspondante du primal).[6]

1.5.3 Graphe biparti

Un graphe G =(X ;E) est dit biparti si l'ensemble de ses sommets X peut être partitionner en deux sous-ensembles X_1 etX_2, de sorte que les sommets de même sous ensemble ne sont pas adjacents[4].
la figure 1.15 montre un exemple d'un graphe biparti.

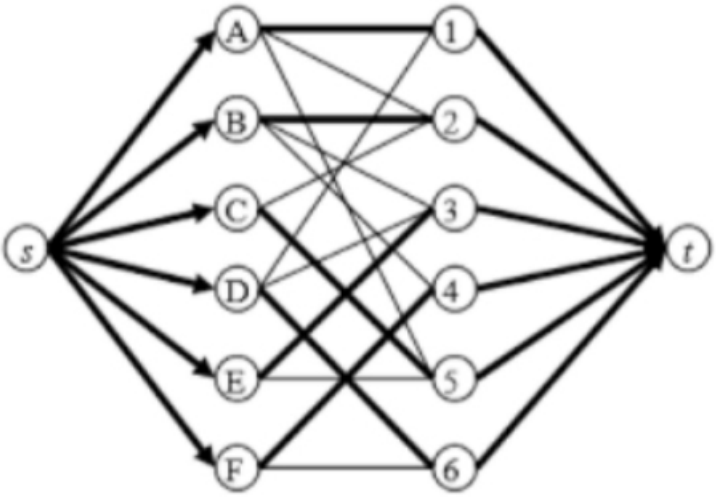


FIGURE 1.10 – Graphe biparti

1.5.4 Arbres et Arborescence

Un arbre est un graphe connexe et acyclique (c'est-à-dire un graphe sans cycle) , Ainsi un arbre est nécessairement simple, une chaîne élémentaire est en particulier un arbre [7].

Caractérisation des arbres

soit G = (X ;E) un arbre,alors :

1. G est acyclique et possède | X | -1 arêtes

2. G est un graphe connexe minimal (chaque arête est un isthme [une arrête dont la surpression rend le graphe non connexe])

3. G est un graphe maximal sont cycle (l'addition d'une arête quelconque crée un cycle),

4. Toute paire de sommets de G est connectée par une chaine unique [7].

Arbres couvrants d'un graphe

un arbre couvrant d'un graphe G = (X ;E) est un graphe A(X' ;E') partiel de G qui est un arbre.

Arborescence

Un arborescence est un graphe G = (X,E) orienté sans circuit admettant une racine $x_1 \in$ X telle que, pour tout autre sommet $x_i \in$ X, il existe un chemin unique allant de x_1 vers x_i.

Si l'arborescence comporte n sommets, alors elle comporte exactement (n 1) arcs.[8]

La figure 1.16 montre une arborescence.

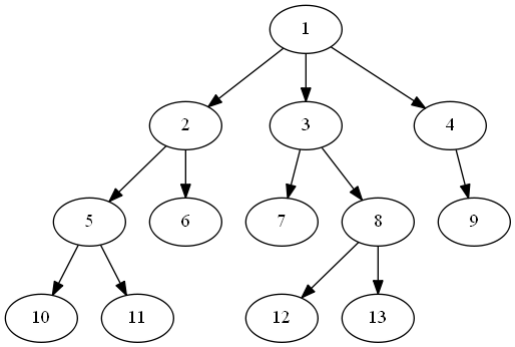


FIGURE 1.11 – Arborescence

1.5.5 Source, puits

une source S est un sommet ascendant de tous les autres sommets,un puits p, un sommet descendant de tous les autres sommets. Note : un synonyme de source (resp.puits) est racine (resp.antiracine).

1.5.6 Racine

Sommet particulier d'une arborescence à partir duquel il existe un chemin unique vers tous les autres sommets du graphe.

1.6 Mode de représentation graphique

1.6.1 Liste d'adjacence

Cette représentation consiste en un tableau de n (avec n le nombre de sommets du graphe) listes chainées. Pour chaque sommet x du graphe, on aura sa liste d'adjacence liste(x) qui sera composée des sommets adjacents a x.[9]
La figure 1.17 montre un graphe non orienté et sa liste d'adjacence.

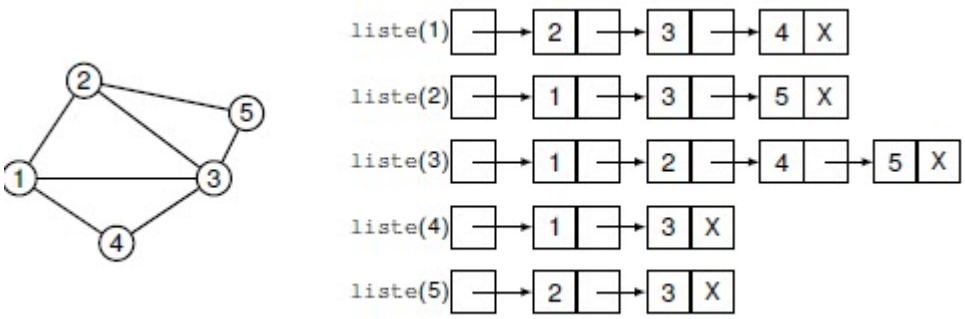


FIGURE 1.12 – Liste d'adjacence

1.6.2 Représentation matricielles

On présente deux types de représentation matricielle

a) Matrice d'adjacence

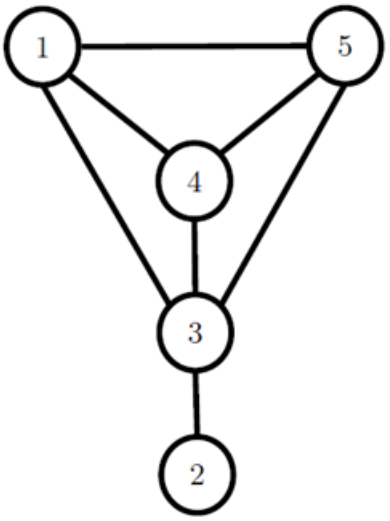
Soit G = (X,E) un graphe d'ordre n, avec X =$\{x_1, x_2, ..., x_n\}$.

On appelle matrice d'adjacence d'un graphe simple G d'ordre n, la matrice carrée M = (m_{ij}) 1≤ i ; j ≤ n telle que :[9]

$$\mathrm{m}_{ij} = \begin{cases} 1 \text{ , si } \mathrm{v}_i v_j \in E; \\ 0 \text{ , sinon.} \end{cases} \quad (1.1)$$

La matrice d'adjacence d'un graphe non orienté est symétrique, ayant pour éléments 0 et 1, avec des 0 sur la diagonale principale.[6]

la figure 1.18 montre un graphe non orienté et sa matrice d'adjacence.



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

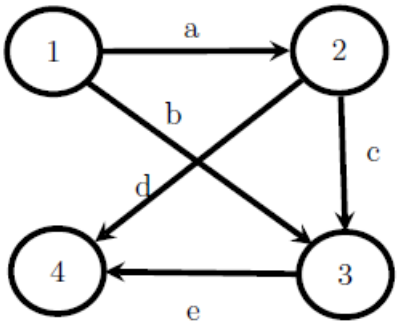
FIGURE 1.13 – Matrice d'adjacence

b) Matrice d'incidence

Soit G=(X,E) un graphe orienté, avec | X | = n et | E | = m, on définit la matrice d'incidence (sommets, arcs) notée m_{ij}, 1≤ i ≤ n, 1≤ j ≤ m par

$$\mathrm{m}_{ij} = \begin{cases} 1 \text{ si le sommet i est le sommet origine de l'arc j.} \\ -1 \text{ si le sommet i est le sommet destination de l'arc j.} \\ 0 \text{ sinon.} \end{cases} \quad (1.2)$$

la figure 1.19 montre un graphe orienté et sa matrice d'incidence.



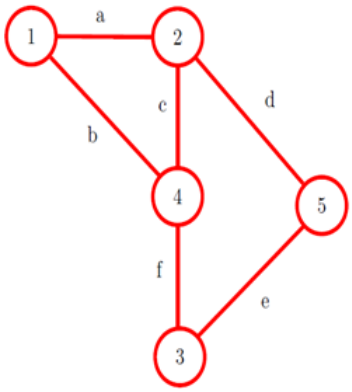
$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} \mathsf{a} & \mathsf{b} & \mathsf{c} & \mathsf{d} & \mathsf{e} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 \\ 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 \end{pmatrix} \end{array}$$

FIGURE 1.14 – Matrice d'incidence d'un graphe orienté

Pour un graphe non orienté : Un graphe non orienté à n sommets numérotés de 1 à n et m arcs numérotés de 1 à m peut être représenté par une matrice (n,m) d'entiers, l'élément m_{ij} vaut :

$$\mathrm{m}_{ij} = \begin{cases} 1 \text{ si le sommet x est une extriemité de l'arete e.} \\ 0 \text{ sinon.} \end{cases} \quad (1.3)$$

la figure 1.20 montre un graphe non orienté et sa matrice d'incidence.



$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

FIGURE 1.15 – Matrice d'incidence d'un graphe non orienté

1.7 Conclusion

Dans ce chapitre nous avons défini les différentes notions relatives à la théorie des graphes qui seront utilisées par la suite.

Chapitre 2

Quelques algorithmes d'optimisation dans les graphes valués

2.1 Introduction

Dans ce chapitre nous allons présenter quelques problèmes connus dans les graphes pour lesquelles nous allons présenter les algorithmes d'optimisation en citant le problème de plus court chemin, le problème d'arbre couvrant minimum, le problème d'affectation, le Problème de flot, et le problème d'ordonnancement.

2.2 Problème de plus court chemin

On se place dans le cas des graphes orientés valués G = (S ; A ; V). Mais les résultats et algorithmes présentés se généralisent facilement aux cas des graphes non orientés valués. Une autre solution consiste à transformer le graphe non-orienté en un graphe orienté en remplaçant une arête entre deux sommets par deux arcs de sens opposé[10].
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 sommets 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 :

Il existe trois types de plus court chemin :

• Plus court chemin d'un sommet à un autre.

• Plus court chemin d'un sommet à tous les autres sommets.

• Plus court chemin entre tous les couples de sommets.[8]

Comme algorithmes de recherche de plus court chemin, nous présentons :

2.2.1 Algorithme de Dijkstra

Dijkstra est un algorithme permettant de résoudre le problème du calcul de l'arbre des plus courts chemins, dans un graphe dont les arcs sont tous de poids positif ou nul, permet de calculer toutes les distances minimales et les plus courts chemins associés dans un graphe connexe G = (V,E,w) depuis un sommet source s vers tout les autres sommets.

Données : R=(S,A,V), s :Racine

Résultats : π : Les plus courtes distances, B : Arborescence, C : Sous-ensemble de sommets accessibles a partir de s.

a) Initialisation : C=$\{s\}$,π(s)=0, B= $\emptyset$, B= π(j)={ v_{sj} si $(s,j)\in$ A,$+\infty$ si non.

b) Procédure de calcul :

(.) chercher un sommet $i \in$(X\C) vérifiant : π(i)=min{ π(j)} j$\in$ (X\C)

(..) Poser :$C = C \cup \{i\}$ et $\pi(j) = min\{\pi(j), \pi(i) + v_{ij}\}$ $j \in \tau_i \cap C^C$

C) Test d'arrêt : si |C|=|S|, Terminer, le Réseau R contient au moins un plus court chemin de s à tout sommet i.

Sinon : retourner en l'étape(b). [11].

Considérons le graphe de la figure 2.1 ou chaque sommet représente une ville on cherche avec l'algorithme de Dijkstra le plus court chemin entre la ville A et la ville F.

Les étapes de l'algorithme sont résumées dans le tableau suivant :

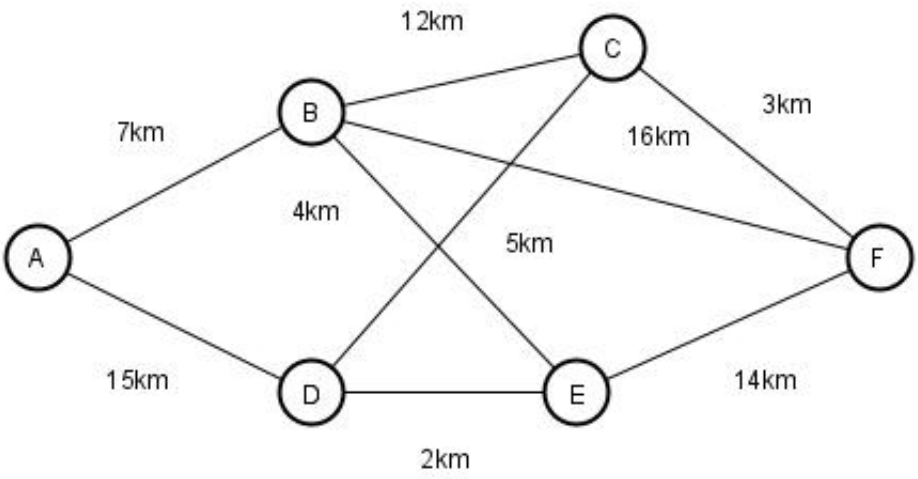


FIGURE 2.1 – Graphe représentant 6 villes.

A	B	C	D	E	F
OA					
OA	$7A$		$15A$		
	7A	$19B$		$11B$	$23B$
			$13E$	**11B**	$25E$
		$18D$	**13E**		
		18D			$21C$
					21C

Le plus court chemin de la ville A à la ville F est : F-C-D-E-B-A

2.2.2 Algorithme de Bellman

On applique cet algorithme pour chercher une arborescence des plus courts chemins dans un réseau R = (X, U, d) sans circuit.
Données : R=(S,A,V) s :racine

Résultats : π : Plus courtes distances de s à x, B : Arborescence , C : Sous-ensemble de sommets accessibles à partir de s. [11]

a) Initialisation : π (s)=0, B=$\emptyset$, C={ s }.

b) Procédure de calcul :

(.) Chercher un sommet j $\in C^C$ telle que Γ_j^{-1} est contenu dans C

(..) Calculer : π(j)=min{ π (i) + v_{ij}} i $\in \Gamma_j^{-1}$

(...) Poser : C=C $\cup$ { J }

c) Test d'arrêt : Si | C | = | S |, terminer

Sinon : retourner en l'étape (b)

Considérons le graphe de la figure 2.2

a) π(s)=0, C=1

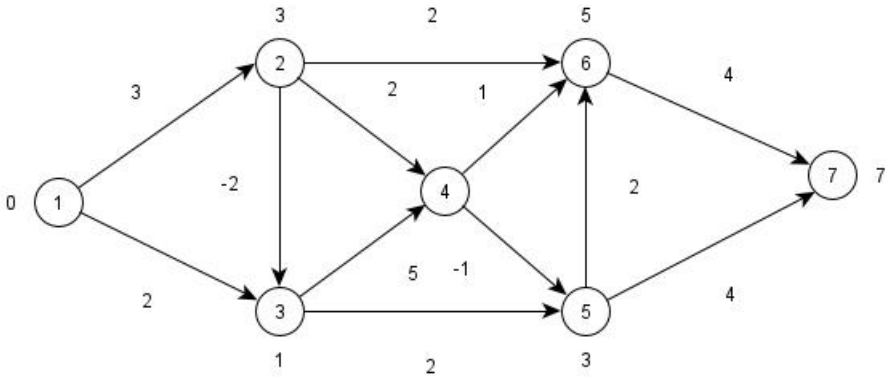


FIGURE 2.2 – Réseau à 7 sommets.

b) (.)2$\in C^c$ et Γ_2^{-1} est contenu dans C (..) π(2)=0+3 a=(1,2)$\in$B(...) C={1,2}

C) |C| $\neq$ |S|

b) (.)3$\in C^c$ et Γ_3^{-1} est contenu dans {1,2} (..) π(3)=min{3-2,0+2}=1, a=(2,3) (...) C=(1,2,3)

C) |C| $\neq$ |S|

b) (.)4$\in C^c$ et Γ_4^{-1} est contenu dans {1,2,3} (..) π(4)=min{3+2,1+5}=5, a=(2,4) (...) C=(1,2,3,4)

C) |C| $\neq$ |S|

b) (.)5$\in C^c$ et Γ_5^{-1} est contenu dans C(..) π(5)=min{1+2,5-1}=3, a=(3,5) (...) C=(1,2,3,4,5)
C) |C| $\neq$ |S|
b) (.)6$\in C^c$ et Γ_6^{-1} est contenu dans C(..) π(6)=min{3+2,5+1,3+2}=5, a=(2,6) ou bien a=(5,6)
(...)C=(1,2,3,4,5,6)
C) |C| $\neq$ |S|

b) (.)7$\in C^c$ et Γ_7^{-1} est contenu dans C (..) π(7)=min(3+4,5+3)=7 a=(5,7)(...) C=(1,2,3,4,5,6,7)
C) |C| = |S| , Terminer, les plus courtes distances π et l'arborescence des plus courts chemins sont déterminés.

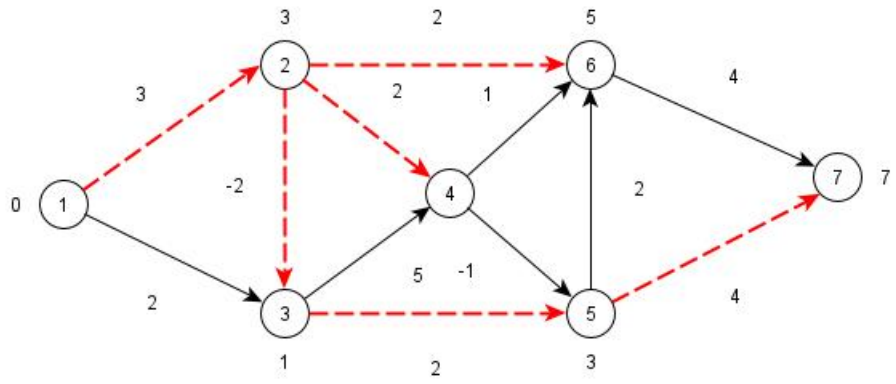


FIGURE 2.3 – L'arborescence de plus courts chemin.

2.2.3 Algorithme de Ford

Données :Un graphe value G=(S,A,V), sans circuit absorbant,s_1 une racine.
Résultats : Un potentiel π, Une arborescence de plus courts chemins B.[11]

a) Initialisation : poser π (s_1) =0 et π (s_j)= $+\infty$; $\forall$j ,j$\neq$1.
b) Tester s'il existe un arcs (s_i,s_j) $\in$ A , vérifiant : π (j)> π (i) +v_{ij} alors poser π (j) = π + v_{ij}
Sinon : Terminer le graphe G'=(S,B) ou B={ (s_i,s_j) $\in$ A tq : π (j)=π (i) + v_{ij} } Contient au moins un chemin C de longueur minimale.

2.3 Problème d'arbre couvrant

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

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)[8].

Pour la recherche d'arbre de poids minimum dans un réseau, nous allons utiliser les algorithmes suivantes :

2.3.1 Algorithme de Kruskal

L'algorithme de kruskal 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[8].

Algorithme

Début

Données :G = (X,E,C) un graphe,|X|=n.

Résultat :T = (X, 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 = (X, F).

Fin si ;

Fin pour ; Si ‖ E ‖ = n-1 Fin.

2.3.2 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'arêtes formant un arbre sur l'ensemble des sommets du graphe initial, et tel que la somme des poids de ces arêtes soit minimal. [8]

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 = (X,E,C) un graphe connexe d'ordre n.

Résultat :T(X,E) arbre couvrant de poids minimum.

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

a) Initialisation

I = $\emptyset$, NI = l'ensemble de tous les noeuds.

Début

1. Mettre le noeud de départ dans I ;

2. Si T est connexe alors aller a (5) ;

3. Trouver un noeud de NI dont la distance à un noeud quelconque a de I est minimal ;

On relie le noeud a à b et faire I = I $\cup$ b, NI = NI/{ b } ;

4. Aller à (3) ;

Si ‖ I ‖ = n-1

5. Fin.

Considérons le graphe de la figure 2.3

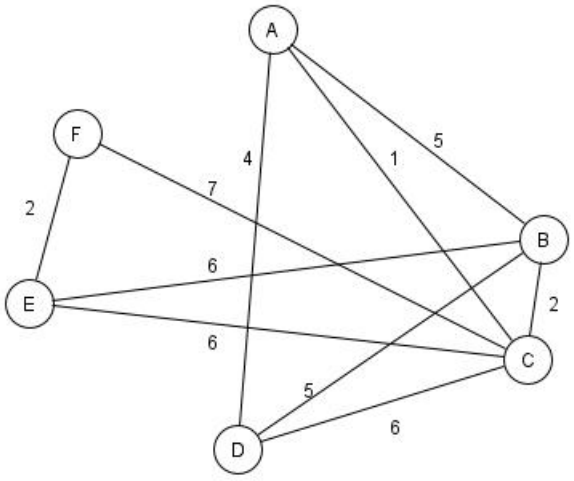


FIGURE 2.4 – Recherche d'arbre couvrant minimal.

Les étapes de l'algorithme sont résumées dans le tableau suivant :

Sommet	A Dist proche	B Dist proche	C Dist proche	D Dist proche	E Dist proche	F Dist proche
init	∞ /	∞ /	∞ /	∞ /	∞ /	∞ /
A		$5A$	**1 A**	$4A$		
C		**2 C**		$4A$	$6A$	$7C$
B				**4 A**	$6C$	$7C$
D					**6 C**	$7C$
E						**2 E**

Une arête de moins que le nombre de sommets :

{ {A,C},{B,C},{A,D},{C,E},{E,F} }

Poids de l'arbre couvrant égale à 1+2+4+6+2=15

2.4 Problème d'affectation

Le problème d'affectation est un problème classique de recherche opérationnelle consiste à attribuer au mieux des tâches à des agents, exemples affectation des ouvriers à la réalisation des tâches différentes, affectation des tâches à des machines[12].

Pour la résolution de problème d'affectation on propose la méthode hongroise.

On suppose, dans la suite que :

- Il y a autant de tâches que de machines, c'est-à-dire n=m si ce n'est pas le cas, il est toujours possible d'ajouter une machine (ou une tâche) fictive avec des coûts d'exploitation nuls.
- Tous les poids sont positifs ou nuls. Si ce n'est pas le cas, on peut augmenter tous les poids d'une même valeur, cela ne changera pas la structure de la solution optimale.

2.4.1 Algorithme hongrois

Soit V = $(v_{ij})1 \leq i,j \leq n$ la matrice des coûts associée à un problème d'affectation de coût minimal.

- Phase 1 : Réduction initiale
  1. Soustraire l'élément minimum de la ligne i de chaque élément de la ligne i, pour tout i = 1, ..., n.
  2. Soustraire l'élément minimum de la colonne j de chaque élément de la colonne j, pour tout j = 1, ..., n.
- Phase 2 : Marquage des zéros. Prendre la ligne contenant le moins de zéros ;
  encadrer le premier zéro de cette ligne et barrer les autres zéros de la ligne et de la colonne du zéro encadré.
  Refaire cette opération jusqu'à impossibilité d'encadrer un zéro.
- Phase 3 Recherche d'une solution optimale
  1. Procédure de marquage des lignes et des colonnes :
     – Marquer les lignes ne contenant aucun 0 encadré (s'il n'y en a pas : FIN) ;
     – Marquer toute colonne qui a un 0 barré sur une ligne marquée ;

- Marquer toute ligne qui a un 0 encadré dans une colonne marquée ;
- Retourner à b et c jusqu'à ce qu'il n'y ait plus de ligne ou de colonne à marquer.

2. Rayer chaque ligne non marquée et chaque colonne marquée.
3. Réduction : choisir le plus petit élément p du tableau non rayé, l'ajouter aux colonnes non rayées et le soustraire aux lignes rayées.
4. Retour à Phase 2.

On considère le tableau ci-dessous qui ce rapporte sur 5 ouvriers qu'on doit affecter à 5 postes les nombres du ce tableau sont les couts d'affectation

	$J1$	$J2$	$J3$	$J4$	$J5$
A	2	2	11	9	9
B	2	3	9	10	3
C	4	10	5	3	6
D	2	7	5	3	6
E	5	1	9	2	10

Le graphe associé au tableau ci-dessus est donné dans la figure 2.5

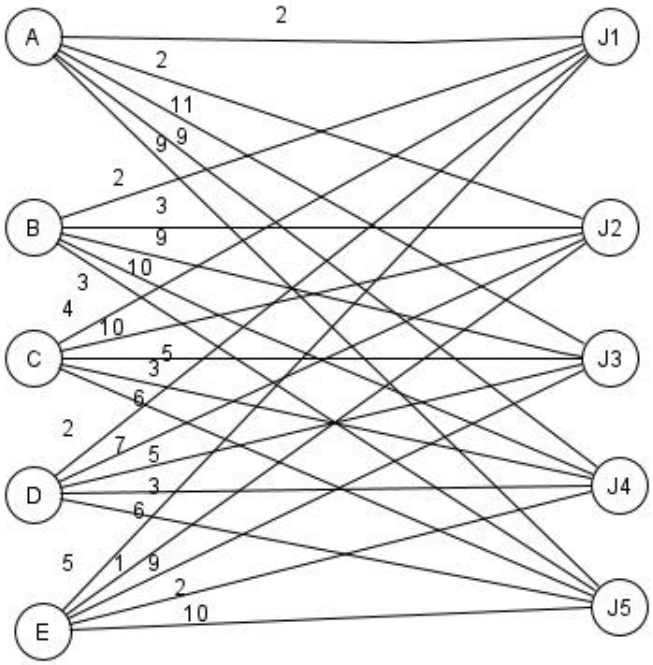


FIGURE 2.5 – Graphe d'affectation.

Les étapes de l'algorithme hongroise sont :

- Etape 0 : Réduction du tableau initial

- Soustraire à chaque ligne du tableau initial, le plus petit élément de la ligne.

- Soustraire à chaque colonne du tableau obtenu, le plus petit élément de la colonne.

	J1	J2	J3	J4	J5
A	2	2	11	9	9
B	2	3	9	10	3
C	4	10	5	3	6
D	2	7	5	3	6
E	5	1	9	2	10

min
2
2
3
2
1

⇨

	J1	J2	J3	J4	J5
A	0	0	9	7	7
B	0	1	7	8	1
C	1	7	2	0	3
D	0	5	3	2	4
E	4	0	8	1	9

	J1	J2	J3	J4	J5
A	0	0	9	7	7
B	0	1	7	8	1
C	1	7	2	0	3
D	0	5	3	2	4
E	4	0	8	1	9
Min			2		1

	J1	J2	J3	J4	J5
A	0	0	7	7	6
B	0	1	5	8	0
C	1	7	0	0	2
D	0	5	1	2	3
E	4	0	7	1	8

- Etape 1 : Encadrer et barrer les zéros

-Identifier la ligne comportant le moins de zéros non barrés (s'il y en a plusieurs, choisir la plus haute).

-Encadrer un des zéros de cette ligne (choisir le plus à gauche).

-Barrer tous les zéros se trouvant sur la même ligne et sur la même colonne que le zéro encadré.

-Recommencer l'étape 1 jusqu'à ne plus pouvoir encadrer, ni barrer des zéros

	J1	J2	J3	J4	J5
A	~~0~~	[0]	7	7	6
B	~~0~~	1	5	8	[0]
C	1	7	[0]	~~0~~	2
D	[0]	5	1	2	3
E	4	~~0~~	7	1	8

La solution obtenue n'est pas optimale car le nombre de zéros encadre est inférieur à 5.

Etape 2 : Marquer et barrer les lignes et les colonnes.

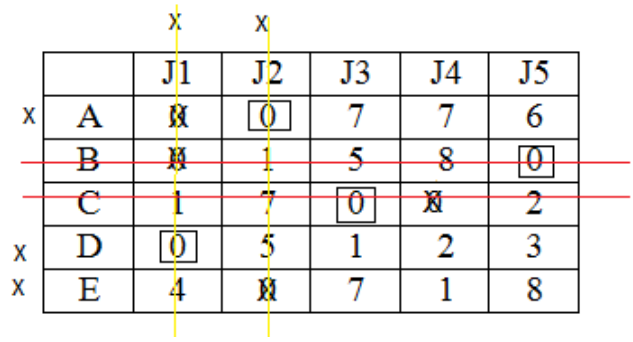
a. Marquer d'une croix toutes les lignes ne contenant aucun zéro encadré.

b. Marquer d'une croix toutes les colonnes ayant un zéro barré sur une ligne marquée.

c. Marquer d'une croix toutes les lignes ayant un zéro encadré sur une colonne marquée.

d. Répéter b et c jusqu'à ne plus pouvoir marquer deux lignes et de colonnes.

e. Tracer un trait sur toutes lignes non marquées et sur toutes les colonnes marquées.



	J1	J2	J3	J4	J5
A	~~0~~	0	7	7	6
B	~~1~~	~~1~~	~~5~~	~~8~~	~~0~~
C	~~1~~	~~7~~	0	~~0~~	2
D	0	5	1	2	3
E	4	~~1~~	7	1	8

Tableau partiel

7	7	6
1	2	3
7	1	8

Min = 1

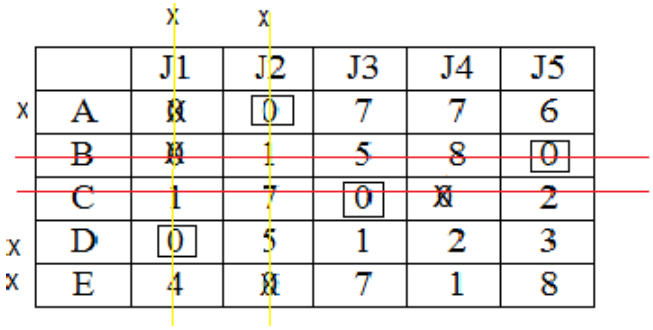
Etape 3 : Modification du tableau

- Les cases non traversés par un trait constituent un tableau partiel.

- Soustraire à toutes les cases du tableau partiel le plus petit élément de celui-ci.

- Ajouter le plus petit élément du tableau partiel à toutes les cases barrés deux fois (horizontalement et verticalement).

- Aller à l'étape 1.



	J1	J2	J3	J4	J5
A	~~0~~	0	7	7	6
B	~~1~~	~~1~~	~~5~~	~~8~~	~~0~~
C	~~1~~	~~7~~	0	~~0~~	2
D	0	5	1	2	3
E	4	~~1~~	7	1	8

	J1	J2	J3	J4	J5
A	0	0	6	6	5
B	1	2	5	8	0
C	2	8	0	0	2
D	0	5	0	1	2
E	4	0	6	0	7

	J1	J2	J3	J4	J5
A	0	~~0~~	6	6	5
B	1	2	5	8	0
C	2	8	~~0~~	0	2
D	~~0~~	5	0	1	2
E	4	0	6	~~0~~	7

Chaque lignes et chaque colonne contient un zéro encadrer, donc la solution est optimale.

Le cout de l'affectation = 2+3+3+5+1=14

2.5 Problème de flot maximum

Le problème de flot maximum consiste à transporter la quantité maximale possible d'une origine (source) à une destination (puits) données, sans dépasser les capacités des arcs.
Dans le problème de flot maximum, on cherche parmi les vecteurs de flot admissibles, ayant une divergence nulle en tout sommet différent de s ou t, celui qui maximise la divergence de s.
Nous présentons maintenant les algorithmes de flots, qui vont nous permettre de traduire de manière efficace le résultat théorique d'intégrité des flots . Nous commençons par le problème du flot de valeur maximale , car l'algorithme est plus simple à présenter dans ce cas. Nous cherchons donc un flot de valeur maximale v d'une source s à un puits t, qui soit admissible, c'est-à-dire qui vérifie la contrainte de capacité , puis nous présentons l'algorithme du flot maximum à coût minimum.[12]
L'algorithme le plus connu pour résoudre ce problème est celui de Ford Fulkerson.

2.5.1 Algorithme de Ford Fulkerson

l'algorithme de Ford Fulkerson permet de calculer un flot maximum entre un sommet source S et un sommet puit T . Le principe et de chercher à chaque itération une chaîne améliorant ou joignant S et T , Si on trouve une telle chaîne alors on calcule l'augmentation du flot d'une valeur ε tel que

$$\varepsilon = \min\{\ \varepsilon_1, \varepsilon_2\ \}$$

avec

$$\varepsilon_1 = min_{(i,j)\in\mu^+}\ \{\ \mu_{ij} - x_{ij}\ \}$$
$$\varepsilon_1 = min_{(i,j)\in\mu^-}\ \{\ x_{ij}\ \}$$

en effet ,on peut augmenter la valeur du flot de " unités sur les arcs avant de $(^+)$ et diminuer de μ unités sur les arc arrières de (μ^-) . l'algorithme se termine lorsque on arrive pas à trouver une chaine augmentante.

procédure du marquage

pour la recherche d'une chaîne améliorante, on utilise la procédure de marquage suivante :

- Initialisation S = $\emptyset$,$\mu^+ = \mu^- = \emptyset$
- On marque le sommet s d'un " +" ,S =$\{s\}$.
- On marque d'une " + " une sommet j $\in$ S tel que

$$x_{ij} < \mu_{ij} \text{ , i} \in \text{S, } \mu^+ = \mu^+ \cup \text{(i,j), S=S} \cup \{j\}$$

- On marque d'un " " un sommet j S, tell que

$$x_{ij} > 0 \text{ , i} \in \text{S, } \mu^- = \mu^- \cup \text{(i,j), S=S} \cup \{j\}$$

- On arrête cette procédure lorsqu'on marque le sommet puits (T) .

Algorithme

1. Initialisation : k = 0, partir d'un flot initial réalisable ,par exemple x = 0.
2. à l'itération k, soit x^k un flot réalisable
   – trouver une chaîne augmentée μ^k reliant s et t en utilisant la procédure de marquage .
   – S'il n'en n'existe pas , alors x^k est un flot maximum ,arrêter l'algorithme
   sinon aller à (3).
3. Mise à jour du flot x^{k+1}
   soit ε^k la capacité réalisable de la chaine μ^k, alors on pose

$$\mathrm{x}_{ij}^{k+1} = \begin{cases} \mathrm{x}_{ij}^k + \varepsilon^k \text{ si (i,j)} \in \mu^{k+} \\ \mathrm{x}_{ij}^k - \varepsilon^k \text{ si (i,j)} \in \mu^{k-} \\ \mathrm{x}_{ij}^k \text{ si (i,j)} \in \mu \end{cases} \quad (2.1)$$

   k = k + 1 , aller à (2).
   (La coupe minimale s est composée de tous les sommets marqués durant la dernière itération de l'algorithme .)

2.6 Problème d'ordonnancement

Pour la formulation et la résolution du problème d'ordonnancement de projet, on utilise les notions de la théorie des graphes. En effet, la majorité des méthodes développées en gestion de projet repose sur la notion de réseau, qui est un graphe simple valué.[13]

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.

Il existe plusieurs méthodes de résolution de ce problème à savoir la méthode PERT.[14]

2.6.1 Méthode PERT

Le terme PERT est l'acronyme de "Program Evaluation and Review Technique", en français : Technique d'évaluation et d'examen de programmes.

Le planning PERT est surtout un outil d'analyse et d'organisation, il servira à établir les autres types de planning parce qu'il est peu pratique pour visualiser l'évolution journalière des opérations, ne comportant ni échelle de temps, ni calendrier. La méthode PERT consiste à analyser de façon systématique et critique les diverses opérations d'un projet et leur enchaînement.

Réalisation du réseau PERT

Les tâches à réaliser sont représentées par un graphe mettant en évidence toutes les relations qui existent entre elles tel que :

– Chaque tâche est associée à un arc (i ; j) du graphe.

– La valeur de l'arc (i ; j) corresponde à la durée de la tâche en question.

– Les sommets sont utilisés pour traduire les relations de succession temporelle ou les étapes (figure 2.7) entre les tâches.

Ainsi, si la tâche x doit suivre la tâche y, l'extrémité terminale de l'arc (i ; j) représentant la tâche x coïncidera avec l'extrémité initiale de l'arc (j ; k) représentant la tâche y.[13]

Deux sommets (étapes) d et f fictifs sont necessaire pour représenter le début et la fin de projet.

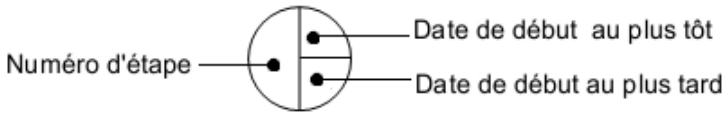


FIGURE 2.6 – Représentation d'un sommet dans un réseau PERT.

Calcul des dates

Après avoir tracé le réseau PERT, on peut calculer les dates de début et de fin au plus tôt et au plus tard.

Date de début au plus tôt (t_i) : C'est le maximum des date de fin des tâches qui précèdent directement la tâche en question. Autrement dit, il est impossible de commencer cette tâche avant cette date.

$$t_i = \max\{\ t_k + d_k\ \},\ k \in \Gamma^-(i)$$

$\Gamma^-(i)$: Ensemble des prédécesseur de la tâche (i ; j).

Date de début au plus tard (T_i) : C'est la date limite de commencement d'une tâche i sans retarder celles des tâches qui la succèdent.

$$T_i = \min\ \{\ T_k + d_i\ \},\ k \in \Gamma^+(i)$$

$\Gamma^+(i)$: Ensemble des successeurs de la tâche (i ; j).

Date de fin au plus tôt ($t_i + d_i$) : Est la date de fin correspondant au commencement d'une tâche au plus tôt. Ce qui signifie que si une tâche a commencé au plus tôt, elle finira aussi au plus tôt.

Date de fin au plus tard ($d_i + T_i$) : Est la date de fin correspondant au commencement d'une tâche au plus tard.

Autrement dit, si une tâche a commencé au dernier moment, pour ne pas retarder la tâche suivante, elle finira aussi au plus tard.

- Marge

Considérons une tâche (i ; j) et sa durée d_i dans le réseau R = (X ;U ;D), le calcul des dates de début au plus tôt et au plus tard, permet de déterminer les différentes marges.

- Marge totale (MT_i) : La marge totale de la tâche (i ; j) est le retard maximale période de temps que l'on peut allouer à une tâche sans influencer la durée du projet.

Lors de l'exécution du projet, si la marge total de la tâche (i ; j) est consommée, elle deviendra critique.

$$MT_i=(T_j\text{-}(t_i+d_{ij})),\ \forall j\in \Gamma^+(i)$$

- Marge libre (ML_i) : Est la réserve de temps dont on dispose sur une tâche (i,j) qui permet, si elle est consommée de ne pas retarder les dates au plus tôt des tâches ultérieures.

$$ML_i=t_j\text{-}(t_i+d_{ij}),\ \forall j\in \Gamma^+(i)$$

- Tâches critiques et chemins critiques

– La branche du graphe comportant uniquement des tâches sans aucune marge total est dite chemin critique.

Autrement dit, un chemin critique est le plus long chemin dans le réseau PERT.

– La durée totale du projet est égale à la somme de la durée des tâches se trouvant sur le chemin critique.

– Tout retard intervenant dans la réalisation de l'une de ces tâches se répercutera sur la date de fin du projet.

– Une tâche située sur le chemin critique est appelée critique elle ne dispose d'aucune marge de temps, elle ne peut supporter aucun retard.

– Les tâches critiques devront être particulièrement surveillées pour respecter le délai total du projet.

– Lorsque l'on cherche à réduire la durée totale d'un projet, il faut donc réduire la durée d'une ou de plusieurs tâches critique.

2.7 Conclusion

Les graphes constituent ainsi des outils de modélisation importants et très utilisés. On peut ainsi ramener un problème concret et complexe à un modèle mathématique plus clair qui consiste en l'étude de sommets et arêtes, dans ce chapitre on a présenté les méthodes de résolution de différents problèmes d'optimisation dans les graphes ainsi quelques algorithmes avec leur exemples pour mieux comprendre leurs résolution.

Chapitre 3

Résolution de quelques problèmes concerts

3.1 Introduction

Dans ce chapitre, nous allons aborder dans un premier temps le problème de plus court chemin résolu avec l'algorithme de Dijkstra en suite le problème d'affectation avec l'algorithme hongrois, et à la fin nous allons présenter le problème d'ordonnancement résolu avec la méthode perte.

3.2 Application de l'algorithme de Dijkstra à un problème concret

Après une catastrophe naturelle ou un conflit ou (situation d'urgence dans une région : séisme, inondation, tsunami, ...), lorsque on parle de solidarité, les autres régions interviennent pour acheminer rapidement de l'aide. Ces derniers cherchent le plus court chemin afin d'arriver le plus vite possible et d'éviter les pertes de vies humaines après la catastrophe.
Ce problème peut être modéliser sous forme d'un graphe planaire G tel que les région illustré sur un plan.
Considérons le graphe planaire de la figure 3.1

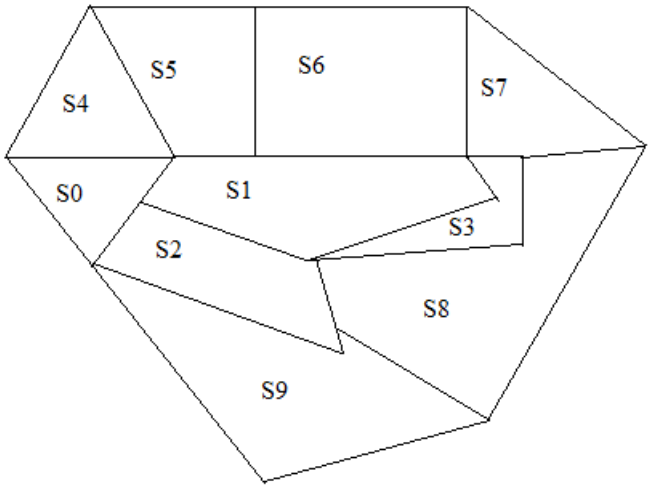


FIGURE 3.1 – Projection 2D du découpage en régions d'une carte géographique.

Le dual de la figure 3.1 est donné par la figure 3.2

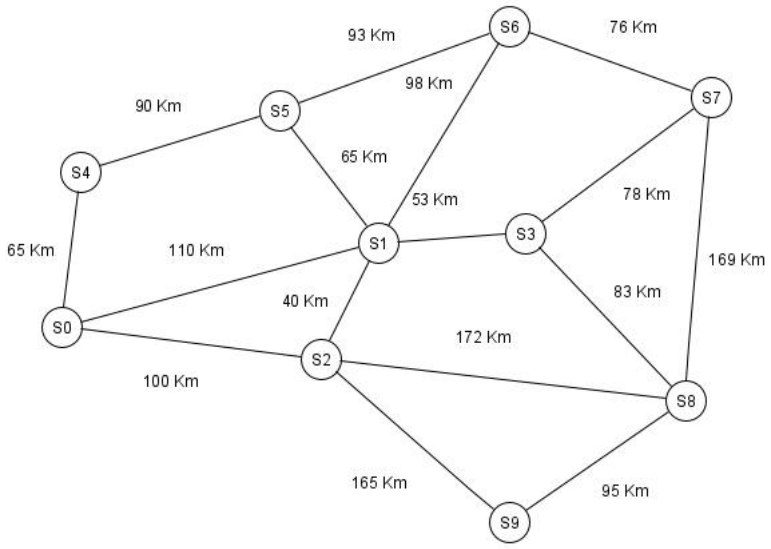


FIGURE 3.2 – Le graphe induit de la carte géographique .

Supposons que la ville S_0 est celle touchée par la catastrophe

Les étapes 1,2,3 de l'algorithme de Dijkstra sont montrées respectivement dans les figures 3.4 , 3.5 , 3.6 .

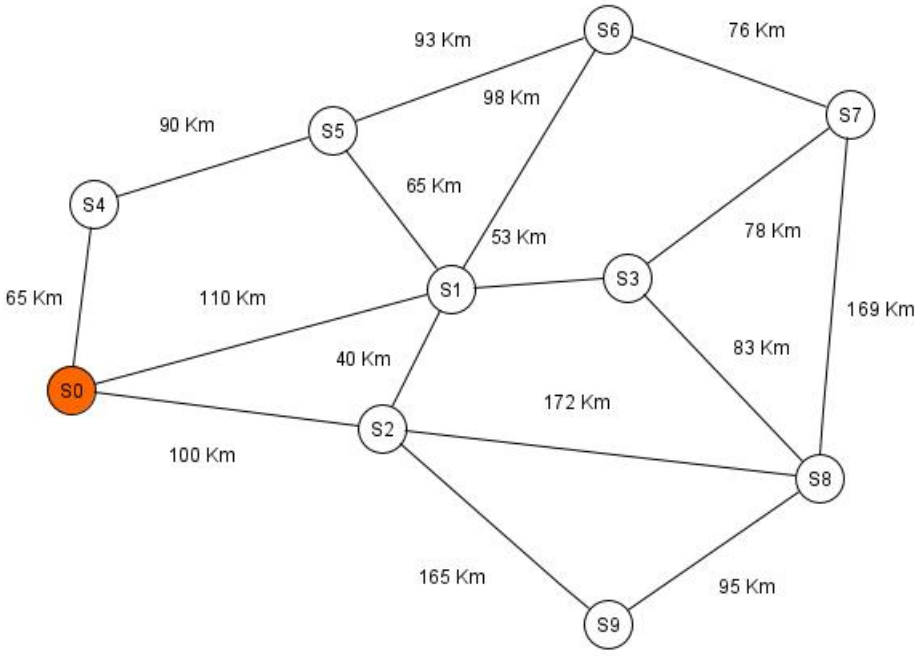


FIGURE 3.3 – Réseau associé après la modélisation.

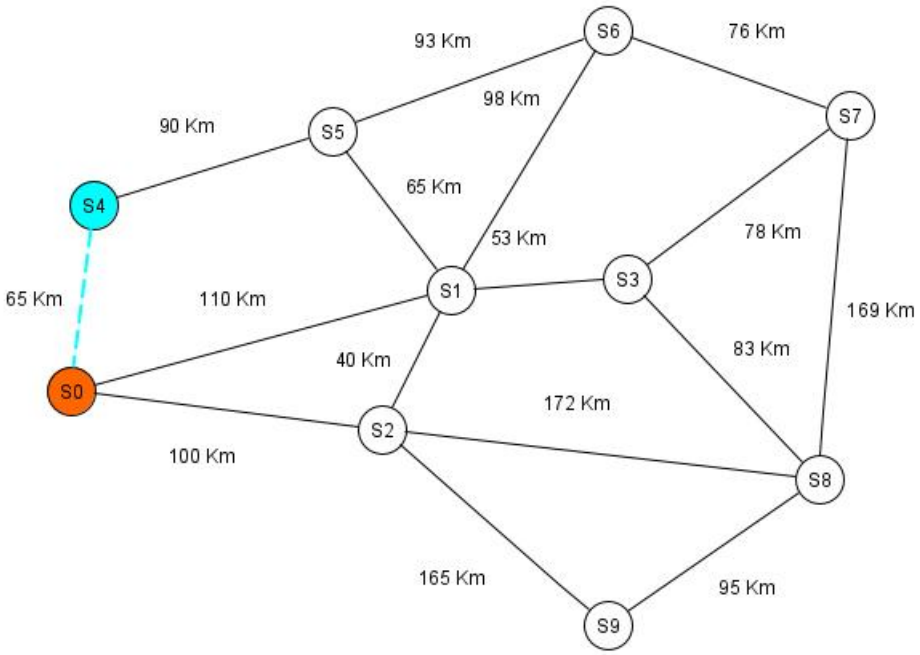


FIGURE 3.4 – Réseau obtenu à la première itération.

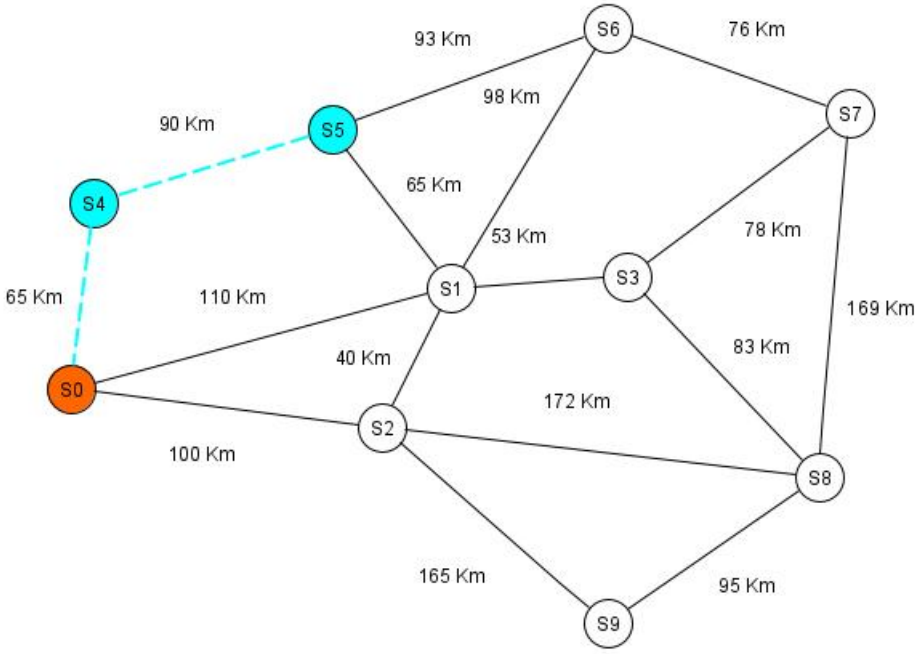


FIGURE 3.5 – Réseau obtenu à la deuxième itération.

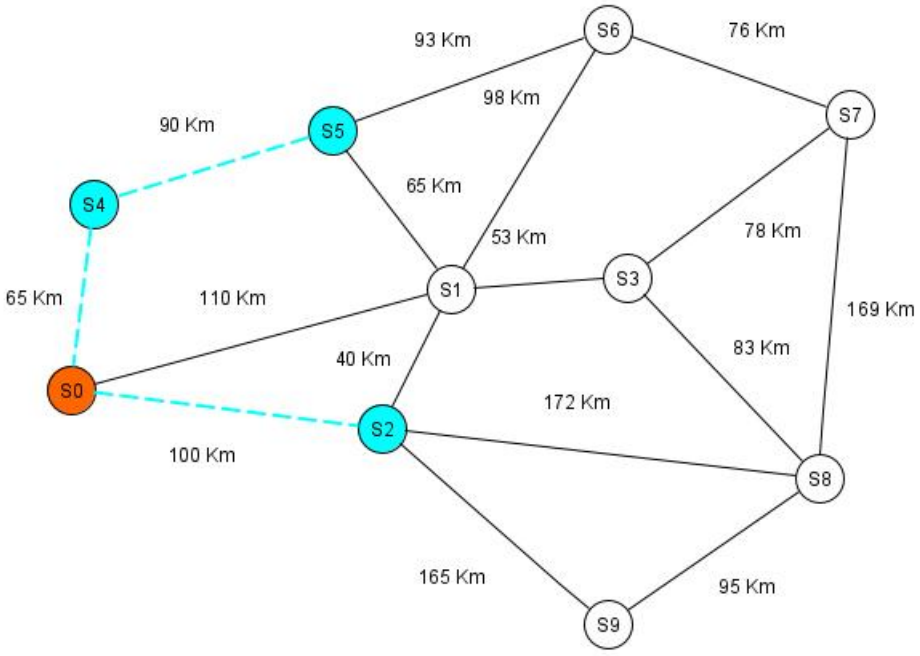


FIGURE 3.6 – Réseau obtenu à la troisième itération.

La dernière étape de l'algorithme de Dijkstra est montrée dans la figure 3.7.

. Dernière Itération

On continue jusqu'à tout les sommet sont marquer, alors on s'arrête, les plus courts chemin sont tels qu'il illustre ce dernier graphe

L'arborescence des plus courts distances, issue du sommet 0 dans le réseau est la suivante.

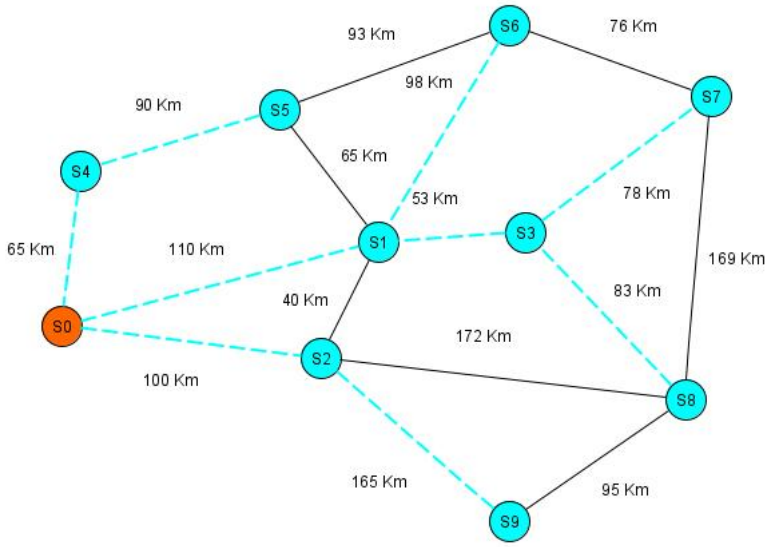


FIGURE 3.7 – Réseau obtenu à la dernière itération.

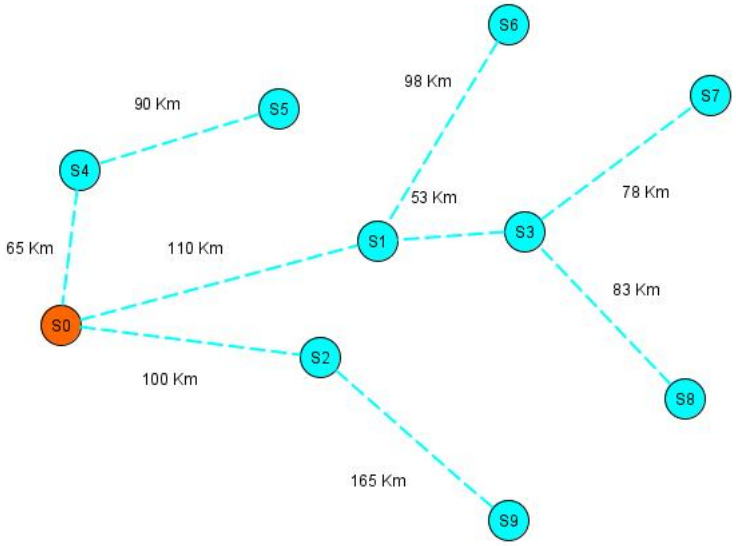


FIGURE 3.8 – L'arborescence des plus courts chemins.

Si le nœud s_9 est la région touchée par la catastrophe alors les autres régions cherchant le plus court chemin pour donné de l'aide comme le montre la figure 3.9.

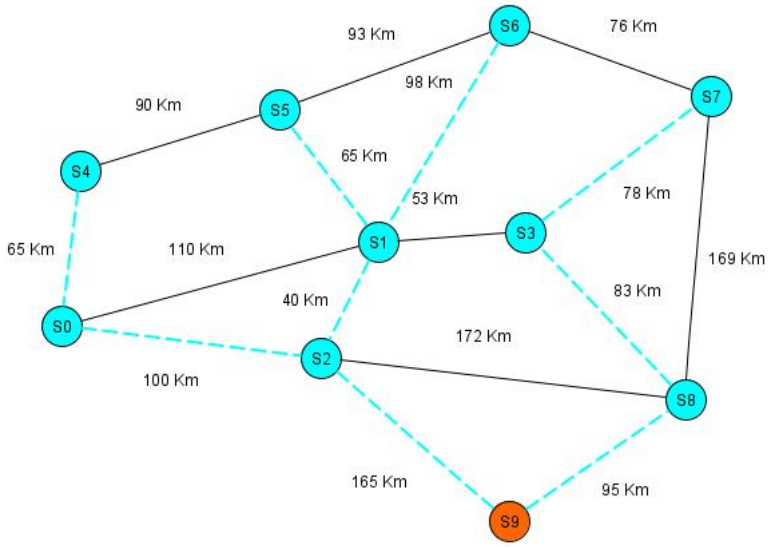


FIGURE 3.9 – Résultat de plus court chemin.

Dans l'arborescence de plus court chemin est donnée par la figure 3.10.

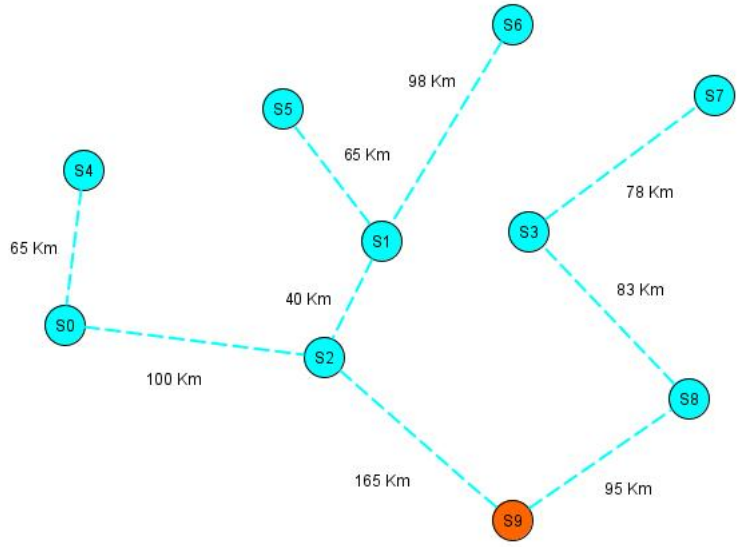


FIGURE 3.10 – L'arborescence des plus courts chemins.

3.3 Application de la méthode pert à un problème concret

Une entreprise algérienne souhaite créer une nouvelle unité de production pour renforcer sa position concurrentielle du marché agroalimentaire cette unité nécessite la réalisation de 9 taches pour la phase logistique, les antériorités et la durée estimative de chaque tache se présentent dans le tableau suivant.

tâche	prédécesseurs	Durée en jours
A	–	7
B	–	3
C	A	4
D	C	5
E	A	2
F	$C.E$	8
G	$A.B$	3
H	G	4
I	$D.F$	5

Collecte des données Le tableau ci-dessus représente les 9 activités, les prédécesseurs et leurs durées. :

Notre problématique consiste à la réalisation de l'ensemble des activités du projet avec seulement les ressources disponibles (en nombre limité) durant une durée limité.

- Le délai de projet doit être calculé en utilisant la méthode P.E.R.T.

Minimisation de la duré totale de projet

Le tableau suivant indique les dates au plus tôt, les dates au plus tard, les marges libres et les marges totale de chaque activité.

N de l'activité	Debut au plus tôt	Debut au plus tard	Marge total
A	0	0	0
B	0	14	14
C	7	7	0
D	11	14	3
E	7	9	2
F	11	11	0
G	7	17	10
H	10	20	10
I	19	19	0
J	24	24	0

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, comme le montre la figure 3.11.

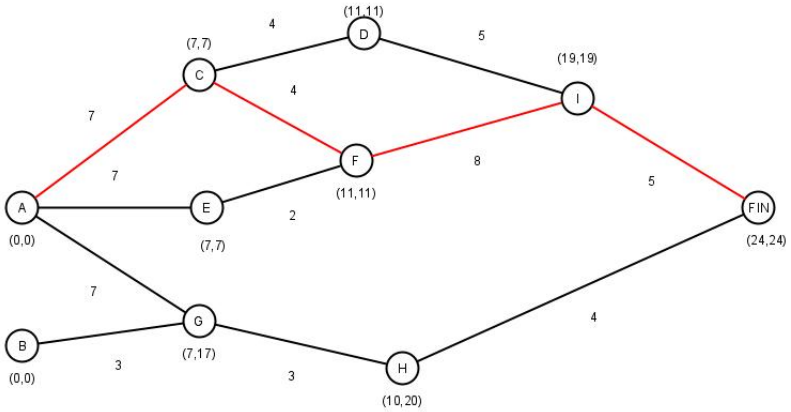


FIGURE 3.11 – Réseau P.E.R.T.

3.4 Application de l'algorithme hongrois à un problème concret

Algorithme de Hongrois pour la recherche d'une affectation minimale

Considérons le problème d'affectation suivant : quatre ouvriers doivent être affectés à 4 tâches, le temps mis par chaque personne pour la réalisation de chaque tâche est donné par la matrice suivante.

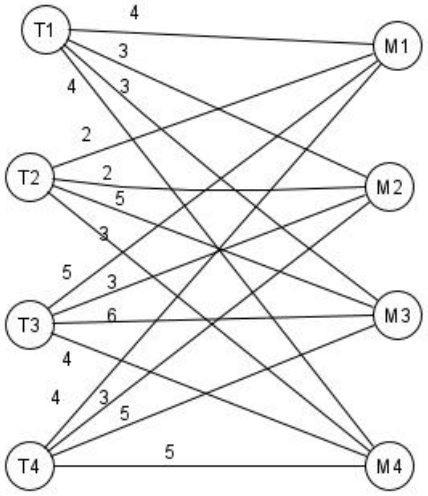


FIGURE 3.12 – Graphe d'affectation.

Phase1 obtention initiale de zéros sur chaque ligne et chaque colonne

	$M1$	$M2$	$M3$	$M4$	Min
$T1$	4	3	3	4	3
$T2$	2	2	5	3	2
$T3$	5	3	6	4	3
$T4$	4	3	5	5	3

Le graphe associé à la matrice ci-dessus est donné par la figure 3.12

	$M1$	$M2$	$M3$	$M4$
$T1$	1	0	0	1
$T2$	0	0	3	1
$T3$	2	0	3	1
$T4$	1	0	2	2
Min				1

Phase 2 :

recherche d'une solution de cout nul

1. prendre la ligne contenant le moins de zéros

2. encadrer le premier zéro de cette ligne et barrer les autres zéros de la ligne et de la colonne du zéro encadré

3. retour à 1 jusqu'à impossibilité d'encadrer un zéro

On obtient 4 zéros indépendants, alors l'affectation est optimale

	M1	M2	M3	M4
T1	1	0	[0]	0
T2	[0]	0	3	0
T3	2	0	3	[0]
T4	1	[0]	2	1

On revient à la matrice de départ, la valeur d'affectation minimale est égale à (3+2+4+3=12) de la matrice de départ

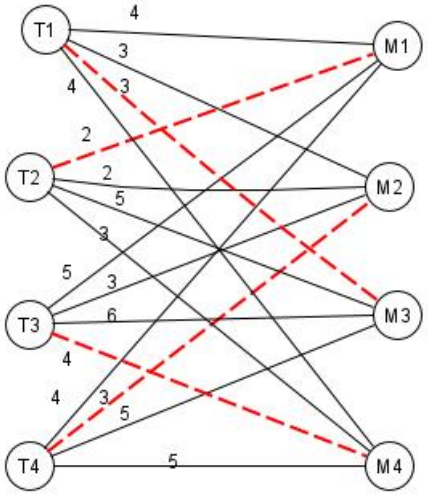


FIGURE 3.13 – Résolution du problème d'affectation.

3.5 Conclusion

Dans ce chapitre nous intéressons a partie réalisation ou nous avons donnée les résultats obtenus relatif certains problème concrets.

Chapitre 4

Implémentation de quelques méthode d'optimisation sur le C++

4.1 Implémentation

Ce chapitre traite le langage de programmation utilisé pour coder les algorithmes étudié dans le cadre de notre travail.

4.1.1 Présentation de langage C++

Apparu au début des années 90, le langage C++ est actuellement l'un des plus utilisés dans le monde, aussi bien pour les applications scientifiques que pour le développement des logiciels. En tant qu'héritier du langage C, le C++ est d'une grande efficacité. Mais il a en plus des fonctionnalités puissantes, comme par exemple la notion de classe, qui permet d'appliquer les techniques de la programmation-objet.

Programmer pour un ordinateur, c'est lui fournir une série d'instructions qu'il doit exécuter. Ces instructions sont généralement écrites dans un langage dit évolué, puis, avant d'être exécutées, sont traduites en langage machine (qui est le langage du microprocesseur). Cette traduction s'appelle compilation et elle est effectuée automatiquement par un programme appelé compilateur.

Un programme écrit en C++ se compose généralement de plusieurs fichiers-sources. Il y a deux sortes de fichiers-sources :

– ceux qui contiennent effectivement des instructions ; leur nom possède l'extension .cpp,

– ceux qui ne contiennent que des déclarations ; leur nom possède l'extension .h (signifiant "header" ou en-tête).

Un fichier.h sert à regrouper des déclarations qui sont communes à plusieurs fichiers.cpp, et permet une compilation correcte de ceux-ci. Pour ce faire, dans un fichier.cpp on prévoit l'inclusion automatique des fichiers.h qui lui sont nécessaires, grâce aux directives de compilation include. En supposant que le fichier à inclure s'appelle untel.h, on écrira include <untel.h> s'il s'agit d'un fichier de la bibliothèque standard du C++, ou include "untel.h" s'il s'agit d'un fichier écrit par nous-mêmes [15].

Aspect pratique

Utiliser Code : :Blocks sous Windows

Lorsque on lance Code : :Blocks, nous voyons apparaitre l'écran ci-contre.

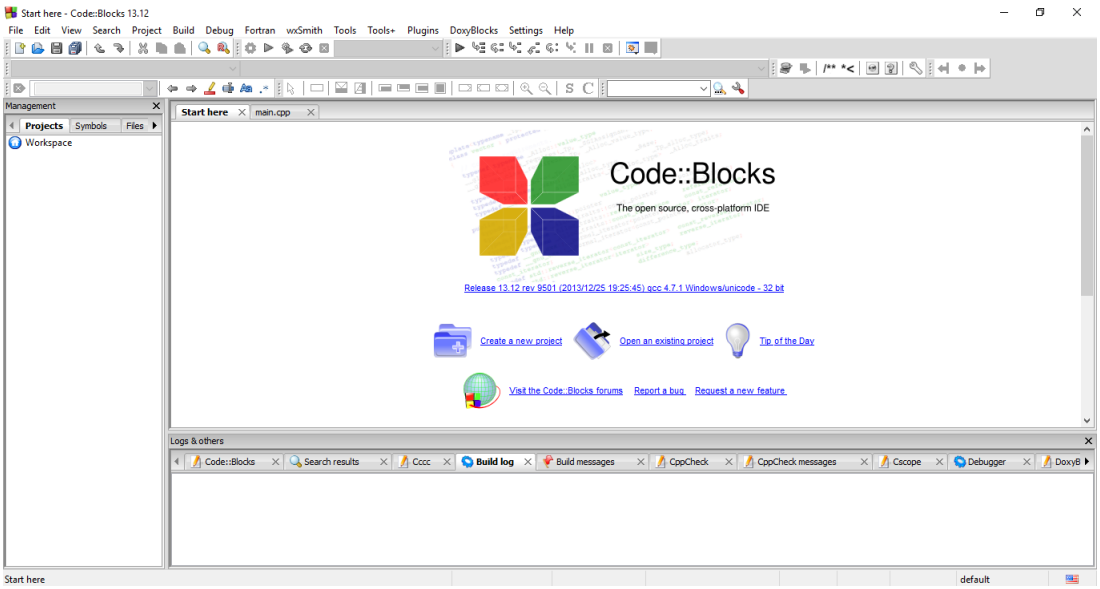


FIGURE 4.1 – l'interface de code blocks.

Pour créer un nouveau projet, il faut choisir dans le menu File puis New puis Project.

4.1.2 Implémentation de l'algorithme de Dijkstra

Nous finissons par restituer les différents résultats de recherche du plus court chemin, obtenus à travers l'exécution de l'algorithme de dijkstra implémenté sous le langage C++.

On introduit la capacité de chaque arc jusqu'à atteindre le dernier.

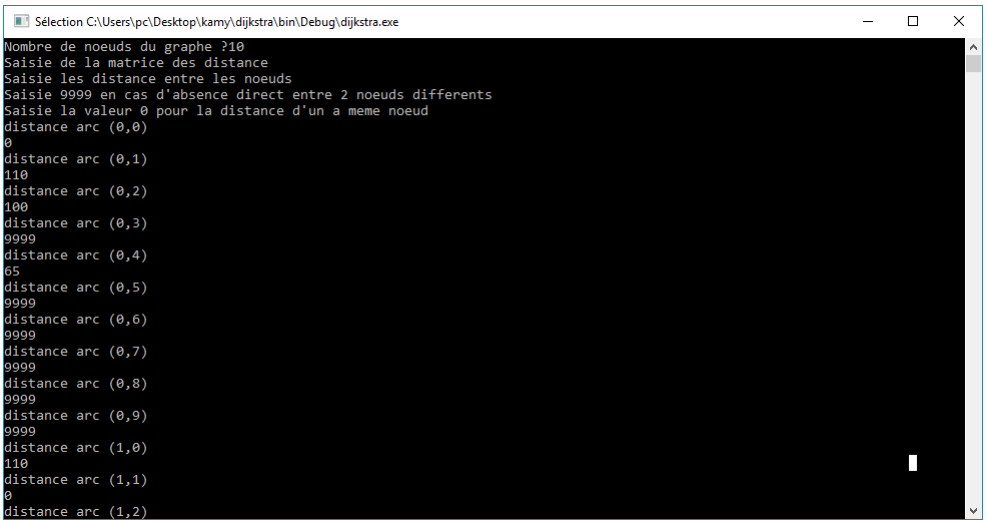


FIGURE 4.2 – Introduire les données.

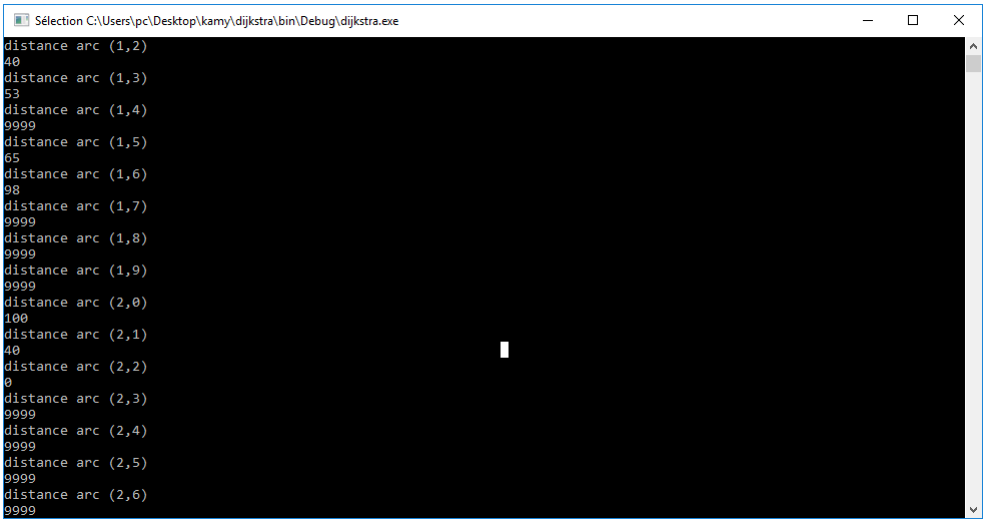


FIGURE 4.3 – Introduire les données.

Résultat

On donne le noeud source est : s_0, le programme nous donne la distance minimale de s_0 a tous les noeud de graphe.

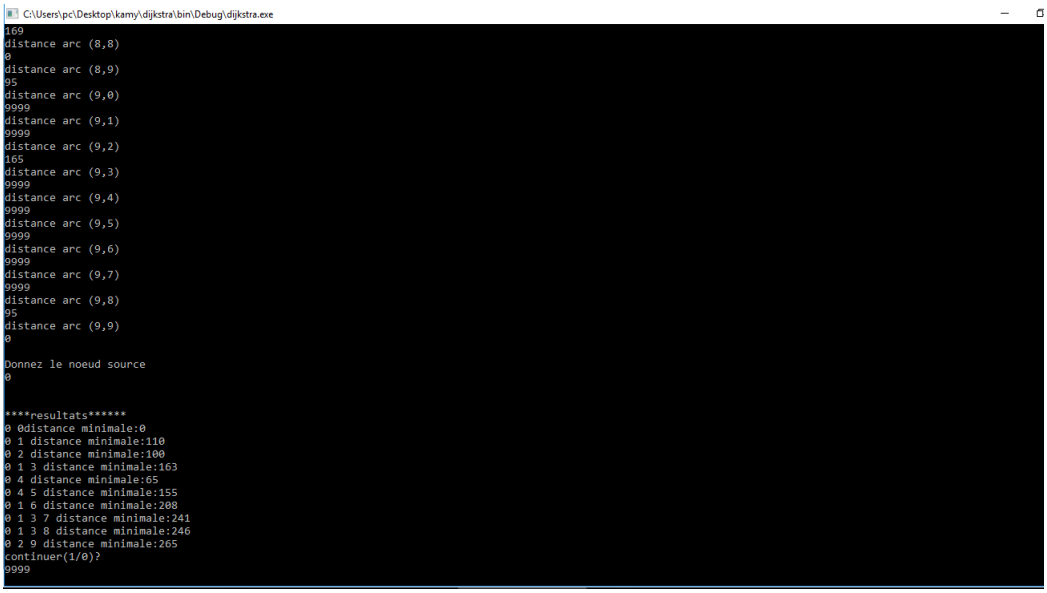


FIGURE 4.4 – Résultat.

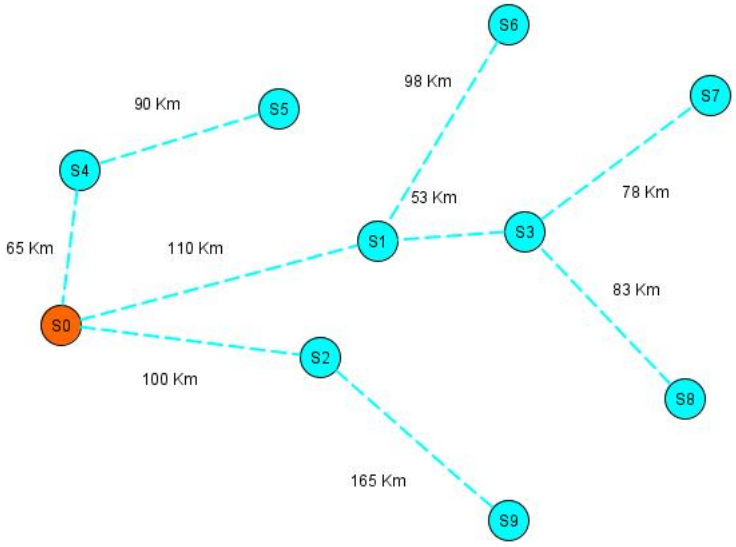


FIGURE 4.5 – L'arborescence des plus courts chemins.

on tape 1 pour continue, on donne un autre noeud source s_9 et s_8

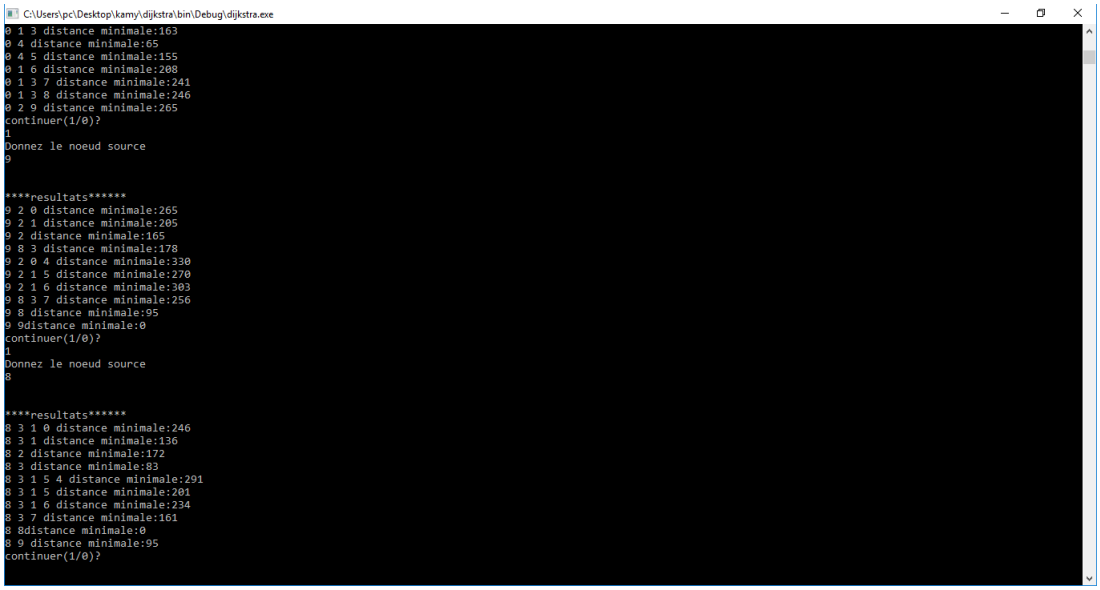


FIGURE 4.6 – Résultat.

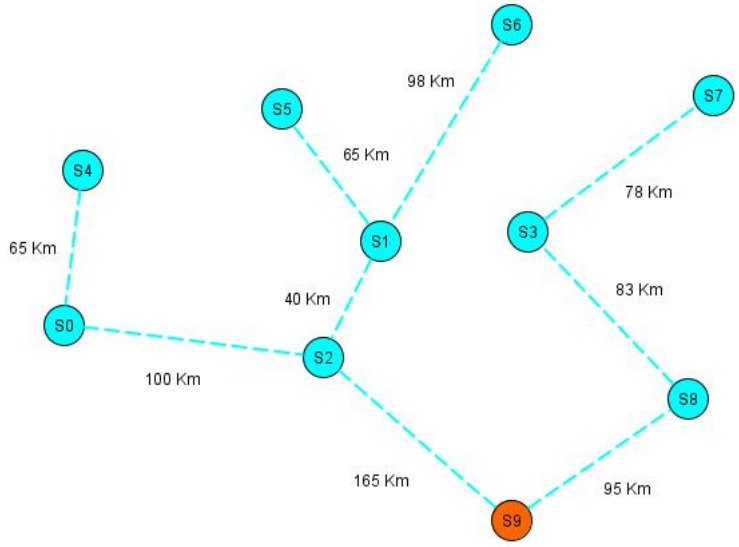


FIGURE 4.7 – Résultat de plus court chemin.

4.1.3 Implémentation de l'algorithme pert

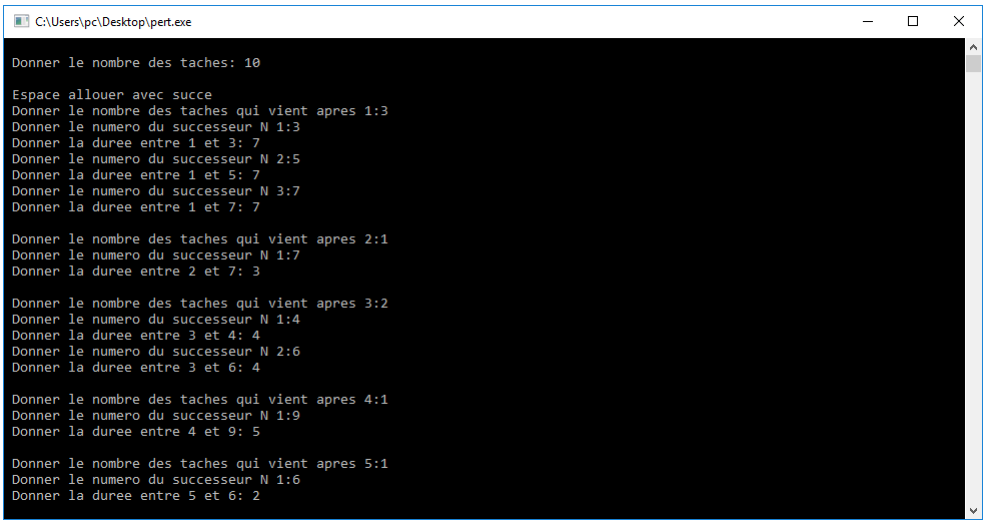


FIGURE 4.8 – Introduire les données.

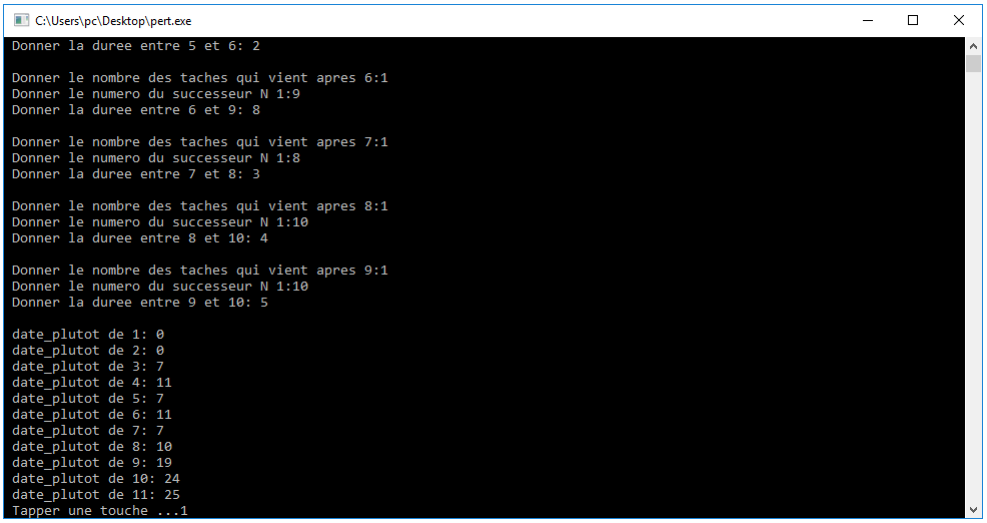


FIGURE 4.9 – Introduire les données.

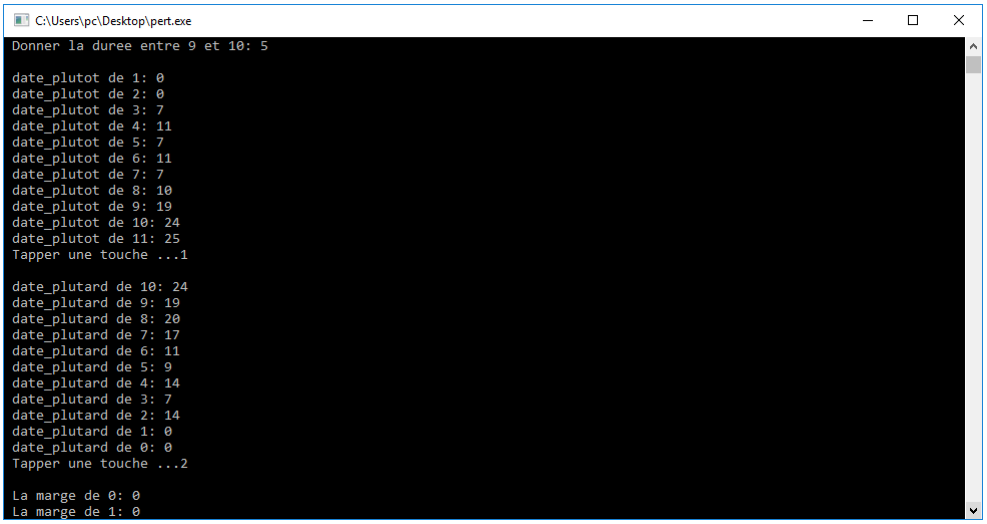


FIGURE 4.10 – Résultat.

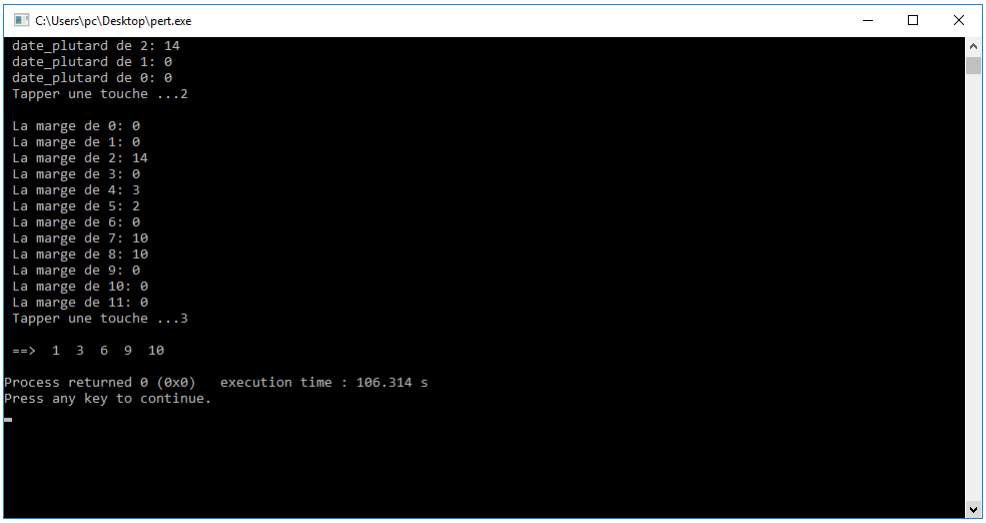


FIGURE 4.11 – Résultat.

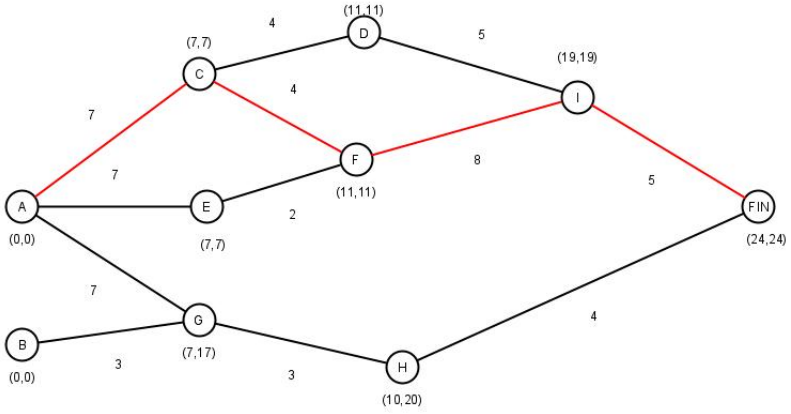


FIGURE 4.12 – Réseau P.E.R.T.

4.1.4 Implémentation de l'algorithme hongrois

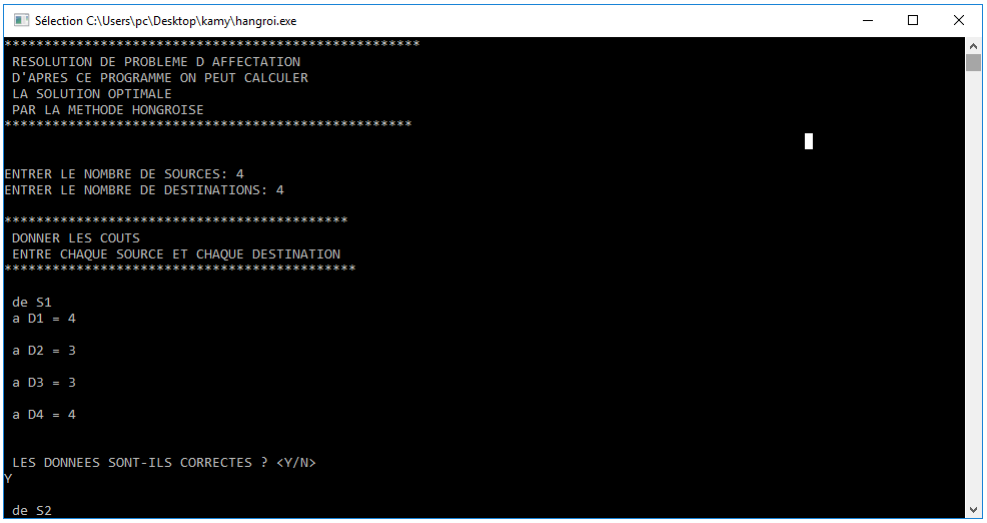


FIGURE 4.13 – Introduire les données.

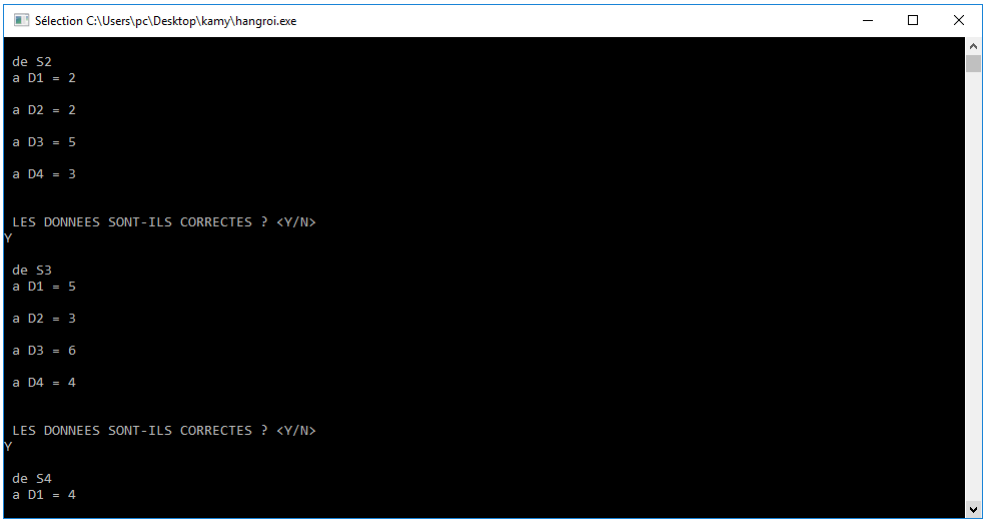


FIGURE 4.14 – Introduire les données.

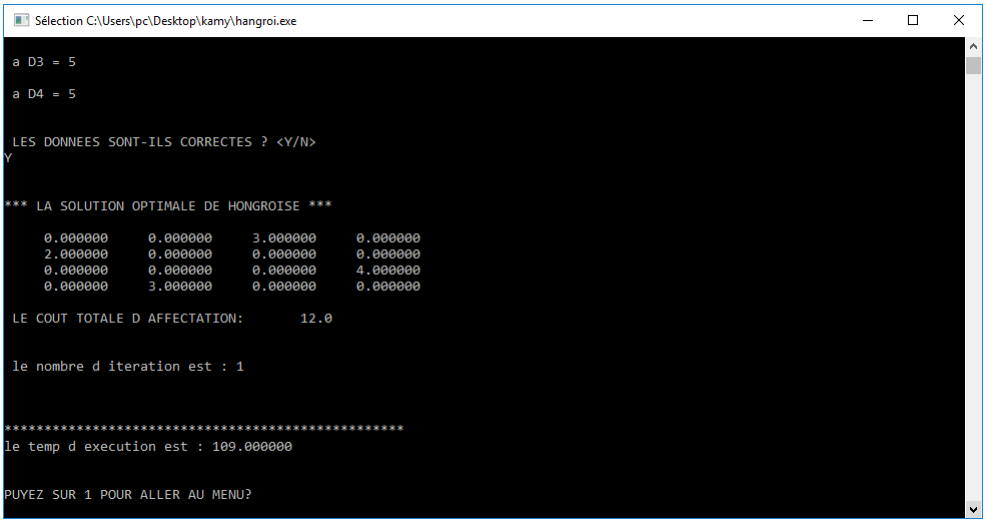


FIGURE 4.15 – Résultat.

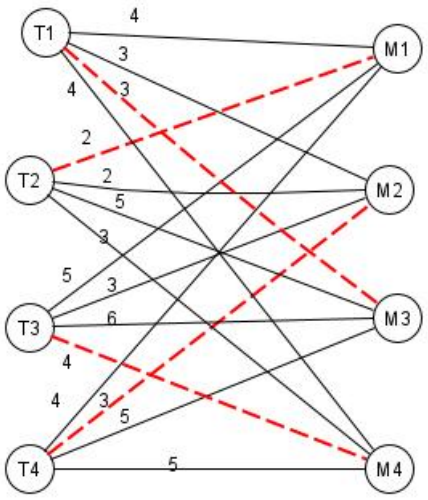


FIGURE 4.16 – Résolution du problème d'affectation.

Conclusion générale

Plusieurs situations de la vie quotidienne font appel aux graphes, c'est-a-dire l'utilisation des graphes pour modéliser les problèmes. Ce mémoire se focalise sur l'étude de quelques méthodes d'optimisation dans les graphes.

Notre travail consiste à donner un certain nombre de méthode d'algorithme pour résoudre des problèmes d'optimisation. Pour cela on a résolu trois problèmes concrets, Le premier est l'élaboration d'une application pour la résolution d'un problème de recherche du plus court chemin entre les régions solidaires et la région touchée par la catastrophe de telle sorte que le trajet soit aussi agréable que possible en utilisant algorithme de dijkstra.

Le deuxième problème est le problème d'affectation peut être résolue par l'algorithme hongrois l'affectation de travailleurs à des machines est moyen reconnu de maximiser la production, mais dépend de la personne maniant la machine a fin d'obtenir une affectation qui soit la plus satisfaisante.

Le troisième problème consiste a minimiser les délais et moyens à réduire le temps et coûts des travaux de remise à neuf, pour cela on a modélisé le problème sous forme d'un réseau PERT.

Notre objectif, l'exactitude, consiste à déverrouiller des problèmes en utilisant, l'optimisation mise à portée par la Théorie des graphes, différents cas ont été traités.

Pour trouver des meilleurs résultats on a implémenté ses algorithmes et méthodes sous le langage c++.

Ce travail nous a également permis d'appliquer nos connaissances acquises durant nos années de formation. Nous espérons que l'étude effectuée apporte un éclairage aux étudiants qui auront à préparer leurs projets de fin d'études ultérieurement.

Bibliographie

[1] A. SADI : Algorithmes linéaires du nombre de broadcast domination de quelques classes de graphes, mémoire de master 2 en mathématique option : Méthodes et modèles de décision Université Mouloud Mammeri, Tizi-Ouzou 2014/2015.

[2] S. YAHIAOUI,N. MAOUNI : Quelques Techniques d'optimisation dans les Réseaux Et Applications, Mémoire de fin de cycle Université A/MIRA de Bejaïa,2017-2018.

[3] C.PHILIPPE, C. VASSARD : Cours TG, IREM de Rouen 2 1736.

[4] I. GUEMARSSA : Reconnaissance des motifs dans des graphes EMF, Mémoire de Fin d'études master filière : informatique 2019 Université de 8 Mai 1945 – Guelma -

[5] M. BENMAAMAR, S. IRID : Quelques techniques d'optimisation dans les réseaux, mémoire de master 2 en Recherche Opérationnele, Université de Bejaia, 2017/2018.

[6] C. SOLNON : Théorie des graphes et optimisation dans les graphes, Institut National des Sciences Appliquées, Lyon.

[7] C. BAHLOUL : Appariement de graphes pondérés approche spectrale, mémoire de master Université Dr. Tahar Moulay Saida, 2019.

[8] S. PELLE : La Théorie des Graphes (cours IT1 2002), Cours conçu au sein de l'Équipe Formats d'Échange du Service de la Documentation Géographique, École Nationale des Sciences Géographiques 2002.

[9] S.KEBBI, N.MAOUCHE : Affectation des vols au niveau de l'Aéroport de Bégaïa, Spécialité Recherche Opérationnelle et Aide à la Décision, Université de Bejaia 2017-2018.

[10] IUT Lyon Informatique « Théorie des Graphes » 2011-2012.

[11] S. TAOUINET : Cours du module, Théorie des Graphes destiné aux étudiantes de licence 2 Recherche Opérationnelle et aide à la décision, Université de Béjaia, (2019/2020).

[12] S.KEBBI, S. MESSAOUD : Modélisation et Résolution de Probléme de Transport, Spécialité Recherche Opérationnelle et Aide à la Décision, Université de Bejaia 2019-2020.

[13] Dr. LARBI ASLI : Support de Cours Management et gestion de projets, Mathématiques Appliquées, Université A/MIRA de Bejaïa 2019-2020.

[14] S. HAYOUNE, N. MAOUCHE : Quelques méthodes d'optimisation et application dans les graphes, Mémoire de fin de cycle Université A/MIRA de Bejaïa,2015-2016.

[15] O. MARGUIN : Cours d'informatique : « C++ : LES BASES » 2003/2004.

Résumé

L'objectif de ce travail est de montrer l'utilité de la théorie des graphes en optimisation, où on a mit en œuvre la résolution de certains problèmes de la RO, et cela en cherchant quelques problèmes d'optimisation pour les quelles nous donnons quelque algorithmes de résolution pour chaque problème suivie d'une résolution, en faisant appel à un programme réalisé sous Dev-C++.
Mot clés : : Graphe, Optimisation, Algorithme.

Abstract

The objective of this work is to show the utility of the Graph theory in optimization, Where we have implemented in the resolution of some problems of the RO, and it by looking for some problems of optimization for the which we give about algorithms of resolution for every problem followed by a resolution, by appealing to a program realized under Dev-C ++.
Key words : : Graphs, Optimization, Algorithm.