Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abdelhamid Ben Badis de Mostaganem
Faculté des Sciences Exactes et de l'Informatique
Département de Mathématiques et Informatique
Filière : Mathématiques

MÉMOIRE DE FIN D'ÉTUDES

Pour l'Obtention du Diplôme de Master en Mathématiques délivré par

Université de Mostaganem

Spécialité M.C.O

présenté par :

BELAOUEDJ Mansouria

les problèmes de cheminement dans un réseau

soutenu publiquement le jour Mois Année devant le jury composé de :

Président :	Prénoms Nom	Grade	Établissement
Examinateur :	Prénoms Nom	Grade	Établissement
Encadreur :	HOCINE Ablaoui	prof	UMAB

Année Universitaire : 2020 /2021

Dédicaces

Je dédie ce mémoire...
À mon chère père
qui m'a toujours transmis l'amour du travail et le sens du perfectionnisme et qui m'a toujours encadré avec beaucoup d'amour et d'attention, que dieu lui réserve bonne santé.
À mes chères frères...
À mes chères sœurs ...
À toute ma famille BELAOUEDJ et AGBOUBI.
À mes amies...
À tous mes enseignants depuis mes premières années d'études.
À ma promotion de master M.C.O 2021
Dédicace spéciale pour l'âme de ma chère mère, que dieu lui accorde le paradis.

BELAOUEDJ Mansouria .

Résumé

Dans ce travail, nous avons étudié et programmé en langage Matlab quatre méthodes différentes de résolutions du problème de recherche d'un plus court chemin d'un sommet à un autre (voire de n'importe quel sommet vers n'importe quel autre). Une étude comparative a été faite entre ces différentes méthodes, en utilisant les exemples donnés par OR-library. La méthode de Bellman bien qu'elle soit efficace, présente l'inconvénient majeur de ne pas traiter les graphes qui contiennent des circuits, celle de Dijkstra est simple efficace mais « exige » qu'il y ait des couts positifs.

Notre objectif a été de concevoir un GPS pour la ville de mostaganem et toute l'algérie ,mais faute de disponibilité de données nous n'avons pas pu le réaliser et on s'attèlera à l'avenir d'y arriver.

Remerciements

Ce travail est le résultat d'efforts continus et de travail acharné tout long de cette année
Tout d'abord nous remercions au créateur de l'univers qui nous à donné de la force et nous à gardé en bonne santé pour terminer ce mémoire et cette année d'étude difficile.
Un remerciement particulier à monsieur **Hocine Ablaoui** pour tout le soutient et l'encadrement qu'il nous a donné.
Nous remercions les membres de jury•pour avoir accepté de présider et pour • avoir accepté d'examiner notre travail.
Nous remercions le corps enseignant et administratif du département de mathématiques et informatique.
Nous tenons à remercier également tous ceux qui nous ont aidé de prés et de loin pour l'élaboration de ce mémoire.
A tous ceux dont le soutient nous à été utile et nécessaire, nous disons, Qu'Allah récompense.

Table des matières

Introduction

Voilà un problème d'optimisation combinatoire par excellence. Pour se déplacer d'un point à un autre, la plus part des voitures sont dotées d'un GPS qui détermine le « meilleur » itinéraire dans un sens que l'usager désire. On résout un problème de recherche d'un plus court chemin d'un point à un autre.
Le problème de la recherche d'un plus court chemin dans un graphe est très varié. cela va du problème le plus simple où l'on veut minimiser la longueur du chemin reliant deux points à la minimisation d'une fonction de la longueur.
Très souvent nous avons des contraintes supplémentaires telles que la recherche d'un chemin Hamiltonien (problème du voyageur de commerce ou les problèmes de tournées de véhicules) ou un chemin Eulérien (problème du postier chinois avec ses variantes).
Les problèmes de cheminement sont très anciens et ils ont connus un grand développement avec l'apparition de la théorie des graphes.
Dans ce travail, on s'intéresse au problème le plus classique : Pour un réseau donné, on détermine un plus court chemin d'un sommet s (une racine) à un sommet p (puits).
Plusieurs algorithmes ont été développés selon que le graphe possède ou pas un circuit (algorithme de Bellman), que les couts soient positifs (algorithme de Dijkstra) ou dans le cas général (algorithme de Floyd et Ford). nous indiquerons l'algorithme de Floyd Warshall qui détermine un plus court chemin entre toute paire de sommets (x, y) du graphe.
Bien que la complexité théorique soit connu pour tous ces algorithmes, nous avons programmé toutes ces procédures en Matlab et fait une étude comparative en utilisant la base de données de OR-Library.

Chapitre 1

Définitions de base et notations

la théorie des graphes est une outil très puissant que l'on utilise pour résoudre des problèmes d'optimisation combinatoire et bien d'autre problèmes.
plusieurs problèmes réels peuvent être modélisés par des graphes : Nous citons les problèmes du voyageur de commerce les problèmes de localisation les problèmes du postier chinois , les problèmes de cheminement et plusieurs autres problèmes.
Dans ce travail,on s'intéresse aux problème de la recherche d'un plus court chemin dans un graphe .
Avant de pouvoir exposer notre problème, nous avons jugé utile de donner quelques rappels sur la théorie des graphes.

1 Définition et notation

1.1 Graphes orienté

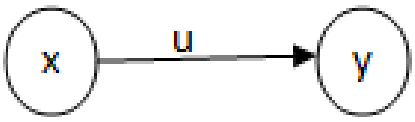
Définition 1.1.

un graphe G est défini par la donnée de :
* d'un ensemble X (fini) appelé ensemble de sommets(noeuds).
* d'un ensemble fini E appelé ensemble d'arcs.
* d'une application I :$E \to X$
A tout arc e, on lui associe son extrémité initiale $I(e)$.
* d'une application T :$E \to X$

A tout arc e, est associé un sommet ,$T(e)$, l'extrémité terminale de e.

On en déduit qu'un graphe est un quadruplé :$G = (X, E, I, T)$
Quand il n'y a pas d'ambiguïté possible et par abus de langage on écrit tout simplement $G(X, E)$.
dans tout ce qui suit, on considère $G(X, E)$, le cardinal de X ,noté $|X|$ égal à n qu'on appelle ordre de G et $|E| = m$.

Soit u un arc de E, si x et y sont, respectivement, l'extrémité initiale et terminale de u alors, x et y sont dits adjacents. si x et y sont confondus, u est appelé boucle.
Un sommet x est représenté par un rond et un arc $u = (x, y)$ par une flèche allant de x vers y.



Dans la figure 1.1, est représenté un graphe orienté d'ordre 5 sur 5 arcs.

1.2 Chemin

Définition 1.2.

Soient G un graphe et a et b deux sommets de X.
Un chemin de a à b dans G , est une séquence d'arcs de G telle que :
1- l'extrémité initiale de premier arc de la séquence est a.

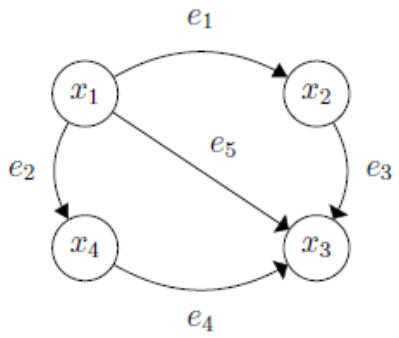


FIGURE 1.1 – Graphe orienté

2- l'extrémité terminale du dernier arc de la séquence est b.
3- pour les autre arcs :l'extrémité terminale de chaque arc coïncide avec l'extrémité initiale de l'arc qui le suit.

Un chemin est dit **élémentaire** si, en le parcourant, on rencontre un sommet en au plus une fois.
Si dans un chemin, en le parcourant, on rencontre un arc en au plus une fois alors il est dit **simple**
séquence : Une séquence d'arcs est une suite ordonnée d'arcs .

Remarque 1.1.

Tout chemin élémentaire est simple.

théorème 1.1.

Tout chemin contient (au sens de l'inclusion des arcs) un chemin élémentaire .

1.3 Graphe non orienté

Définition 1.3.

Un graphe non orienté est défini par :
Un ensemble fini X appelé ensemble de sommets.
Un ensemble fini E appelé ensemble des arêtes.
Une application e définie de :

$E \rightarrow P_2(X)$. $P_2(X)$ est l'ensemble des parties à deux éléments de X.
A toute arête u on associe ses extrémités $e(u)$.
Une arête u dont les sommets sont x et y est notée (xy).
Dans la figure 1.2 nous avons représenté un graphe non orienté d'ordre 4 sur 5 arêtes.

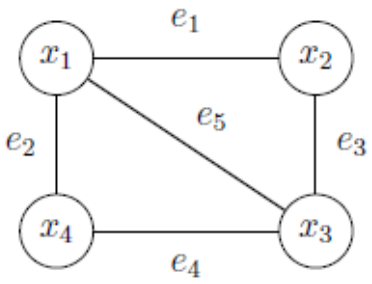


FIGURE 1.2 – Graphe non orienté

1.4 Sous-graphe

Définition 1.4.

Soit $G = (X, E)$ un graphe et X' un sous ensemble de X alors, le sous graphe de G construit X' est le graphe $G' = (X', E')$ tel que E' est constitué des arcs (ou arêtes) dont les extrémités sont dans E'.
Dans la figure1.3 est représenté le sous-graphe construit sur le sous ensemble des sommets $\{x_1, x_3, x_4\}$ du graphe donné par la figure 1.2

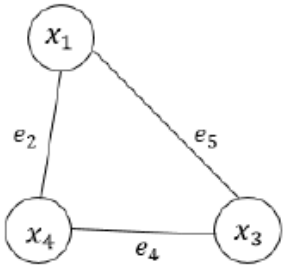


FIGURE 1.3 – sous-graphe

1.5 Graphe partiel

Définition 1.5.

Soit A un sous ensemble de E. le graphe partiel construit sur A est le graphe $H = (Y, A)$.
A la figure 1.4 est représenté un graphe partiel de G construit sur le sous ensemble d'arcs $\{e_1, e_2, e_4, e_5\}$ de E (donné à la figure 1.1)

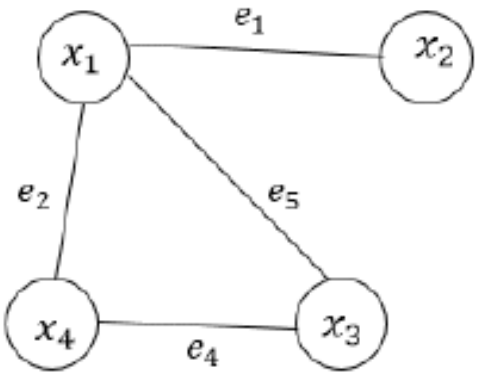


FIGURE 1.4 – Graphe partiel

1.6 Chaînes

Définition 1.6.

Une chaine reliant a et b est une séquence d'arêtes (ou d'arcs) telle que :
1- Une extrémité du le première arête (ou arc) de la séquence est a.
2- Une extrémité du la dernière arête (ou arc) de la séquence est b.
3- Deux arêtes (ou arc) successives de la séquence sont adjacents.

1.7 Cycle

Définition 1.7.

Un cycle de G est une séquence circulaire d'arête (ou arc) tout distincts de G.

1.8 Cycle

Définition 1.8.

Un circuit est une séquence circulaire d'arcs tous distincts.

1.9 Ordre de graphe

Définition 1.9.

Soit $G = (X, E)$un graphe telle que |X|=n, |E|=m.
On appelle ordre du graphe le cardinal de l'ensemble des sommets.

1.10 Connexité

Définition 1.10.

Un graphe connexe est un graphe tel que pour toute paire de sommets x et y, il existe une chaîne reliant x et y, un graphe non connexe est décomposé en plusieurs composantes connexes.
Dans la figure 1.5 est représenté un graphe connexe.

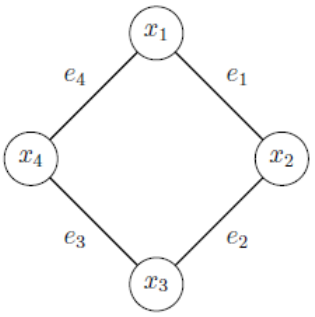


FIGURE 1.5 – Graphe connexe

1.11 f-connexe

Définition 1.11.

G est dit f-connexe s'il possède un seule composante f-connexe.

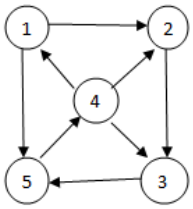


FIGURE 1.6 – f-connexe

1.12 Arborescence

Définition 1.12.

Soit $G(X,E)$ un graphe et soit $\alpha \in X$
"α" et dit Racine (anti racine) s'il existe un chemin de "α" à x pour tout $x \in X$ (respectivement de x à "α").

Définition 1.13.

Soit $G(X,E)$un graphe.
G et dit arborescence si :
1) $\exists\ \alpha \in X\ /\ \alpha$ racine de G.
2) (X,E) Arbre.

théorème 1.2.

$G(X,E)$, $n = |X| \geq 2$.
Le s conditions suivantes sont équivalentes et caractérisent une arborescence.
1) G arbre et possède "α" comme racine.
2) $\forall\ x \in X$, il existe un chemin unique de la racine "α" vers x.
3) G admet "α" comme racine étant minimale pour cette propriété.
4) G connexe et plus $(d_G^-(\alpha) = 0, d_G^-(x) = 1)$.
5) G sans cycles et plus (1) est vérifier.
6) G admet 'n' comme racine et est sans cycles.
le graphe de la figure1.7 est représenté une arborescence

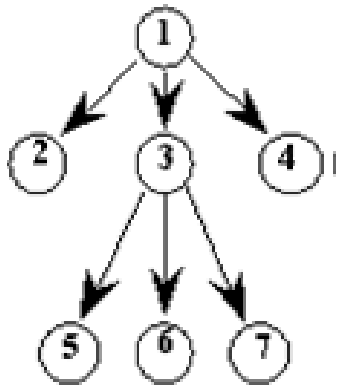


FIGURE 1.7 – Arborescence

2 Représentation Matricielle des graphes

2.1 Matrice d'adjacence

[2] Considérons un graphe orienté $G = (X, E)$ comportant n sommets (d'ordre n). On numérote les sommets de G de 1 à n.
On appelle matrice d'adjacence associée à G la matrice M dont chaque terme m_{ij} est égal au nombre d'arêtes orientées (d'arcs) allant du sommet i vers le sommet j.
La représentation par matrice d'adjacence de G consiste en une matrice booléenne M (ne contenant que des "0" et des "1") de taille n * n telle que :

$$m_{i,j} = \begin{cases} 1 \text{ si } (i,j) \in E \\ 0 \text{ sinon} \end{cases}$$

Propriété :

M est la matrice d'adjacence associée à un graphe orienté dont les sommets sont numérotés.
p désigne un nombre entier naturel Le terme m_{ij} (ligne i et colonne j) de la matrice M^p donne le nombre de chaînes de longueur p reliant i à j.

Exemple 2.1.

La matrice d'adjacence associée au graphe de la figure 1.9 est :

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

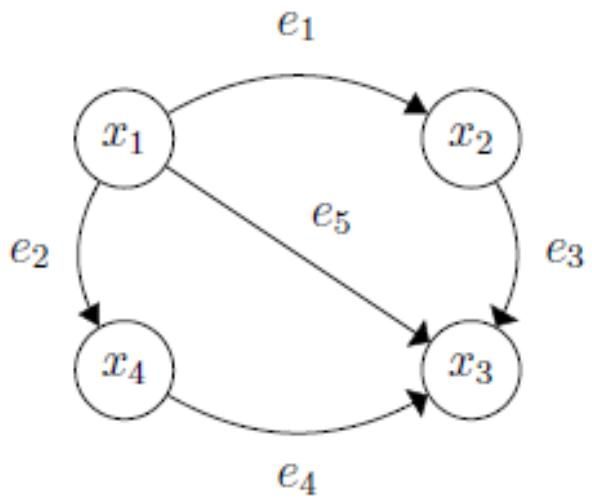


FIGURE 1.8 – Graphe orienté

Remarque 2.1.

1. l'identification des sommets peut aussi s'effectuer avec des lettres. On indique dans ce cas, pour éviter toute ambiguïté, l'ordre choisi sur les lettres pour écrire la matrice d'adjacence.
2. On admet que les propriétés de M^n vues pour les graphes non orientés restent valables pour les graphes orientés.

Pour un graphe non orienté la matrice d'adjacence est définie comme suit :

$$m_{i,j} = \begin{cases} 1 \quad si\ (i,j)\ ou\ (j,i) \in E \\ 0 \quad \text{sinon} \end{cases}$$

Dans le cas dune graphe G non orienté, la matrice d'adjacence M est symétrique.

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

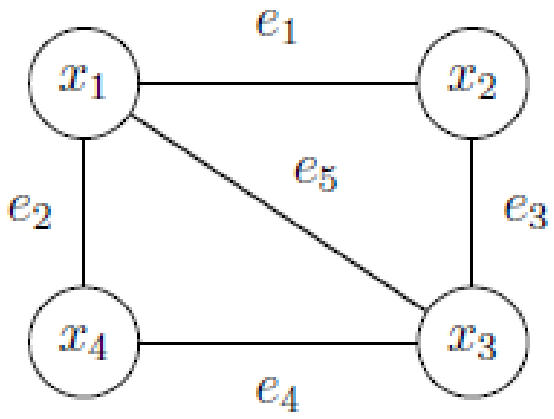


FIGURE 1.9 – Graphe non orienté

2.2 Matrice d'incidence

Une matrice d'incidence est une représentation matricielle d'un graphe montrant la relation d'incidence entre arêtes et sommets. Considérons G un graphe orienté sans boucle $G = (X, E)$ comportant n sommets $\{x_1, x_2, x_3..., x_n\}$ et m arêtes $\{e_1, e_2, e_3..., e_m\}$. On appelle matrice d'incidence de G la $M = (m_{ij})$ de dimension n * m telle que :

$$m_{i,j} = \begin{cases} 1 & si\ x_i \text{ est l'extrémité initiale de } e_j \\ -1 & si\ x_i \text{ est l'extrémité terminale de } e_j \\ 0 & x_i \text{ n'est pas une extrémité de } e_j \end{cases}$$

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

$$m_{i,j} = \begin{cases} 1 & si\ x_i \text{ est l'extrémité de } e_j \\ 0 & \text{sinon} \end{cases}$$

Pour un graphe orienté
La matrice d'incidence du graphe de la figure 1.9, est la suivante :

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

Pour un graphe non orienté La matrice d'incidence du graphe de la figure 1.10, est la suivante :

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

3 Conclusion

La théorie des graphes donc a occupé la majeure partie de ce chapitre, en particulier en ce qui concerne la fourniture de quelques concepts de base sur notre travail que nous aborderons dans les prochains chapitres.

Chapitre 2

Problème de cheminement dans un réseau

1 Introduction

Le problème de cheminement (CP), problème fondamental dans plusieurs disciplines, consiste à choisir un itinéraire de longueur minimum pour aller d'une ville A à une ville B. On suppose donnée une carte routière à laquelle on associe un graphe $G = (X, E)$ ou X, l'ensemble des sommets, est en bijection avec l'ensemble des carrefours de la carte. Deux sommets x et y étant joint par un arc si on peut se rendre du carrefour correspondant à xà celui correspondant à y par un tronçon de route direct. A chaque arc e de E, on associe sa longueur d(e) et on appelle réseau $R = (X, E, d)$ le graphe muni de l'application d.
Le problème de plus court chemin de A à B se formule de la manière suivante : étant donnés deux sommets A et;Bd'un réseau $R = (X, E, d)$, trouver un chemin joignant A à B de longueur minimum, la longueur d'un chemin étant égale à la somme des longueurs des arcs composant la séquence.

Définition 1.1.

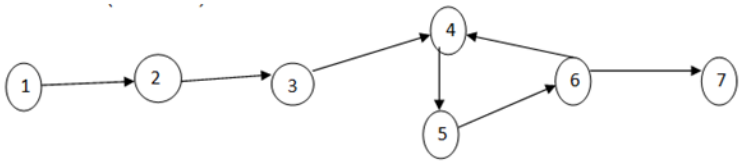
Un graphe orienté $G = (X, E)$.
Une fonction poids $d : E \rightarrow R$ qui à tout arc e on associe son poids $d(e)$. Ceci nous permet d'introduire la notion de réseau R qui est défini par $R = (X, E, d)$. pour tout sous ensemble V de E, on définit le poids, $l(V)$, de V par :

$l(V)=\sum_{e\in V} d(e)$ et par suite la longueur d'un chemin C par : $l(C)=\sum_{e\in C} d(e)$. On conviendra que la longueur d'un chemin qui ne contient aucun arc qu'il est de longueur nulle.

2 Problèmes de cheminement :

Etant donnés deux sommets x et y d'un réseau $R = (X, E, d)$, nous avons trois cas possibles :
a. Il n'existe pas, dans R, de chemin de x à y.
b. Il existe un chemin de x à y, dans R, mais pas de longueur minimum (maximum).
c. Il existe un chemin, dans le réseau R, du sommet x au sommet y qui soit de longueur minimum (maximum).

Exemple 2.1.



$d(1,2) = 10; d(2,3) = 1; d(3,4) = 15; d(4,5) = -10; d(5,6) = 1; d(6,4) = 4; d(6,7) = 6.$
Il existe des chemins du sommet 1 au sommet 7 (et même une infinité de chemins), mais il n'existe pas de plus court chemin de 1à 7 (il suffit d'emprunter le cycle $((4,5),(5,6),(6,4))$ de poids négatif autant de fois que nécessaire). Il existe un plus court chemin de 1 à 3 (il est d'ailleurs de longueur 11) mais in n'existe pas de chemin de 3 à 1.

Définition 2.1.

La longueur d'un plus court (respectivement d'un plus long) chemin de x à y est appelée plus courte distance (respectivement plus longue distance) de x à y. Si on note par $d(x,y)$ la plus courte distance de x à y alors :

$$d(x,y) = \begin{cases} min_{\text{C : un chemin de x à y}}((l(C)) \\ +\infty \quad \text{s'il n'existe pas de chemin de x à y} \end{cases}$$

On introduit trois problèmes :

- **Pro1 :**déterminer un plus court chemin d'un sommet s à un autre sommet p.

- **Pro2 :**Pour tout sommet x de X, déterminer un plus court chemin d'un sommet s à tout sommet x de X.
- **Pro3 :**Pour tout couple de sommets (x,y) de l'ensemble X, déterminer un plus court chemin de x à y.

Ci-dessous, nous donnerons des conditions nécessaires et suffisantes d'existence de plus court chemins et des algorithmes efficaces pour résoudre les problèmes de type **Pro2**. Aussi paradoxal que cela puisse paraitre, l'obtention d'une solution d'un problème de type **Pro1** est obtenue en résolvant le problème de type **Pro2**.

Définition 2.2.

Un circuit de longueur négative est appelé circuit absorbant. le circuit $((4,5),(5,6),(6,4))$ de l'exemple 1 est absorbant puisque sa longueur est égale à -5.

Des conditions nécessaires et suffisantes d'existence d'un plus court chemin est donnée par les résultats suivants :

théorème 2.1.

Dans un réseau $R = (X, U, d)$ f-connexe , il existe un plus court chemin entre tout couple de sommets si et seulement s'il n'existe pas de circuit absorbant dans G.

preuve 2.1.

La condition est nécessaire, car si le réseau R admet un circuit absorbant Cab alors, il suffit de parcourir Cab indéfiniment pour obtenir un chemin de longueur égale à - ∞
La condition suffisante est évidemment vérifiée. En effet : Comme G est f-connexe alors il existe un plus court chemin élémentaire, dans R, entre un sommet x et autre sommet $y$de l'ensemble des sommets X et soit C un plus court chemin élémentaire de x à y. Supposons qu'il existe un chemin $C1$ dont la longueur est strictement inférieure à celle de C, alors $C1$ n'est pas élémentaire, $C1$ contient un chemin élémentaire de x à $y$et soit $C2$ ce chemin, alors $l(C2) \leq l(C1) < l(C)$ (car pour obtenir un chemin élémentaire on supprime tous les circuits que comporte $C1$ - la non existence de circuits absorbants entraine l'inégalité -)ce qui est en contradiction avec le fait que C est un plus court chemin élémentaire.

théorème 2.2.

Une condition nécessaire et suffisante d'existence d'une solution du problème 2 est que ;
a. G admet une racine s.
b. et R ne contient pas de circuit absorbant.

preuve 2.2.

La condition (a) garantit l'existence d'un chemin de s à n'importe quel sommet x de X et, on fait le même raisonnement que précédemment pour prouver le résultat.

théorème 2.3.

Dans un réseau $R = (X, E, d)$, tout portion de chemin C_x^y, située entre un sommet x et un autre sommet y, d'un plus court chemin C de s à p est un plus court chemin entre x et y.

demonstration 2.4. *(procédons par l'absurde)*

Supposons que C_x^y n'est pas un plus court chemin de x à y et soit $C1_x^y$ un plus court chemin de x à y alors $C'= (C \cup C1_x^y)$-C_x^y est un chemin de s à p qui est de longueur plus petite que celle de C. Cela contredit le fait que C est un plus court chemin de s à p, d'où le résultat.

théorème 2.5.

Une condition nécessaire et suffisante, dans le réseau R, pour que les potentiels $\pi(x)$ représentent les plus courtes distances du sommet $s$aux sommets x de X est que :
a) $\pi(s)=0$
b) On ait : $\pi(T(e)) - \pi(I(e) \leq d(e) \; \forall \; e \in E$ (1)
c) Le graphe partiel (X, v) défini par : $v = \{e \in E / \pi(T(e)) - \pi(I(e) = d(e)\}$ (2)
admette s comme racine.

demonstration 2.6.

Condition nécessaire :
Supposons que les potentiels $\pi(x)$ représentent les plus courtes distances du sommet s aux sommets x de X. Soient C un plus court chemin de s à x(x un sommet de X)et $e = (x, y)$un arc de E alors, $C' = C \cup \{e\}$ est un chemin de s y et sa longueur est égale à $\pi(x) + d(e)$ Cette longueur est nécessairement inférieure à à la plus courte distance de s à y (c'est à dire $\pi(y)$) d'où l'inégalité(1). Si l'arc u est situé sur un plus court chemin de s à y, d'après le théorème 3, $d(e)$ est un plus court chemin de x à y et donc : $l(C') = \pi(y)) = l(C) + d(e) = \pi(x) + d(e)$ d'où l'égalité (2) qui est vérifiée.

Condition suffisante :
Si les potentiels $\pi(x)$vérifient (a) et (b) alors pour tout chemin C de s à x on a :
l(C)=$\sum_{e\in c} d(e) \geq \sum_{e\in c}(\pi(T(e)) - \pi(I(e))) = \pi(x) - \pi(s)$ et pour un chemin de s à x dans (X, V) on a l'égalité.
Si l'on souhaite déterminer un plus court chemin allant d'un sommet s vers un autre sommet p (la destination est unique), on pourra utiliser la résolution du problème de type $Pro2$ (qui calcule tous les plus courts chemins partant de s). En effet, on ne connaît pas d'algorithme plus efficace pour résoudre ce problème.
Si l'on souhaite calculer tous les plus courts chemins entre tous les couples de sommets possibles, on pourrait aussi utiliser la résolution du problème précédent, mais dans ce cas, on n'obtiendrait pas un algorithme efficace.
Il faudra utiliser dans ce cas un algorithme spécifique, par exemple l'algorithme de Floyd-Warshall.

3 Arborescence des plus courts chemins

Au fait, on va non seulement déterminer les plus courtes (voire plus longues) distances entre une racine s et tout autre sommet de X, mais des plus courts chemins entres la racine s et les différents sommets de X. cela consiste à déterminer une arborescence de racine s et, pour déterminer un plus court chemin de s à x, on remonte l'arborescence de x vers la racine s.
Nous allons étudier trois algorithmes qui permettent de résoudre le problème pro2 :

- L'algorithme de**Bellman** dans le cas où le graphe ne possède pas de circuits.

- L'algorithme de **Dijkstra** est utilisé dans le cas où tous les poids

sont positifs.

- Enfin un algorithme **"bellman ford"** pour un réseau quelconque.

4 Optimisation dans les réseaux

De nos jours, les réseaux de transport au sein d'une ville sont de plus en plus complexes. Un voyageur commutateur se voit confronté à déterminer un trajet optimal combinant plusieurs modes de transport publics et /ou privés de disponibilité différentes. C'est l'une des plus intéressantes applications du problème du plus court chemin en raison de l'intérêt croisant pour la gestion dynamique des systèmes de transport complexes. Ainsi, nombreuses applications algorithmiques calculant le chemin optimal d'une source à une destination dans ce type de réseau, ont été proposées pour accompagner les voyageurs dans leur planification d'itinéraire Nous étudions dans cet recherche, le problème du plus court chemin dans un réseau dont les données de disponibilité des moyens de transport changent dans le temps à cause des congestions, des retards et des tableaux d'horaire pour le transport public.

2.1 Réseau

Définition 4.1.

Un réseau est un graphe connexe value $G = (X, E, d)$dans lequel il y a un sommet s (appelé la racine) tel que $d^-(s) = 0$ et un sommet p (appelé le puits), tel que $d^+ = 0$. pour $e \in E$, le nombre $d(e)$est appelé le coût de la flèche u.

Dans la figure(2.1) est représenté un réseau .

Remarque :

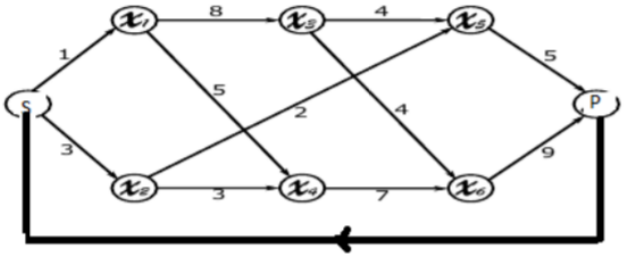


FIGURE 2.1 – Réseau

L'arc en gras est un arc de retour dans le cas des réseaux de transport.

4.1 Problème d'optimisation

Il existe plusieurs problèmes d'optimisation, considérons les problèmes suivants :

4.2 Problème de plus court chemin

Un problème de plus court chemin est un problème algorithmique de la théorie des graphes.
L'objectif est de calculer un chemin entre des sommets d'un graphe qui minimise ou maximise une certaine fonction.
Il existe de nombreuses variantes de ce problème. Étant donné un graphe connexe et fini. 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, selon la nature des valeurs et d'éventuelles contraintes supplémentaires. Dans de nombreux cas, il existe des algorithmes de complexité en temps polynomiale, comme l'algorithme de Dijkstra,... , dans des graphes avec valeur positifs sur les arcs.

Définition 4.2.

4.3 Définition de problème du plus court chemin

[4] on appelle Le problème du plus court chemin le problème suivant : étant donné un graphe G associons à chaque arc u un nombre $d(e) \geq 0$ que appelons la "longueur" de l'arc e ,Trouver un chemin élémentaire C allant d'un sommet a à un sommet b et telle que la longueur totale :
l(C)=$\sum_{e \in c} d(e)$
soit aussi petit que possible.
un chemin e joignant un sommet i à un sommet k est dit de longueur minimale ou maximale s'il minimise ou maximise cette longueur $l(C)$ dans l'ensemble de tous les chemins joignant i à k.
Il sera intéressant d'introduire une matrice M dite matrice de longueurs dont élément M_{ij} se définit comme suit :

$$M_{i,j} = \begin{cases} 1 & \text{si } (i,j) \in e \\ \infty & \text{si } (i,j) \notin e \\ 0 & \text{si } i = j \end{cases}$$

4.4 Cheminement entre deux points x et y

Plusieurs problématiques peuvent être considérées lorsque l'on s'intéresse au cheminement entre deux points d'un graphe. La plus simple consiste à déterminer si un chemin entre x et y existe. On veut savoir s'il est possible de se rendre d'un point x à un point y en utilisant un $(ou\ plusieurs)$ chemin(s). Ce sont des problèmes d'existence de chemins.

Chapitre 3

Résolution de problème

Il est courant, lorsque l'on cherche à se rendre d'un point à un autre dans un réseau (routier, par exemple), de chercher le plus court chemin, c'est-à-dire celui dont la distance est la plus petite. Si le nombre de trajets possibles entre le point de départ et le point d'arrivée est faible, il suffira de calculer les longueurs de chacun des trajets en additionnant la longueur des liens qui le composent et de comparer directement les longueurs obtenues. Mais une telle solution exhaustive devient rapidement impraticable si le nombre de trajets possibles est grand.
Heureusement, il existe des algorithmes qui calculer tous les trajets possibles. Pour cela, ils mettent en œuvre diverses stratégies. C'est ce que nous allons présenter dans le chapitre suivante.[3]

1 Algorithme de résolution

Il existe plusieurs algorithmes pour résoudre les problèmes de cheminement (plus court chemin) dans un réseau, nous représenterons ici les méthodes les plus connues.[1]

1.1 Algorithme de bellman

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

Le principe :

L'idée de l'algorithme de Bellman, est de calculer de proche en proche, l'arborescence des plus courtes distances, issue du sommet s à un sommet donné p.
On calcule la plus courtes distances 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.[1]

Enoncé :

Données : Un réseau $R = (X, E, d)$ sans circuit absorbant avec $d(e) \in R$.
Résultat : Arborescence des plus courtes distances A.

Algorithme de bellman

Début :
Entrée : (réseau R = (X,U,d) , sommet s)
Sortie :(potentiel π , arborescence A , sommets s)
Le graphe est supposé ne contenant pas de circuits

Initialisation: S ← { s } , π(s)=0 , A(s) ← nil

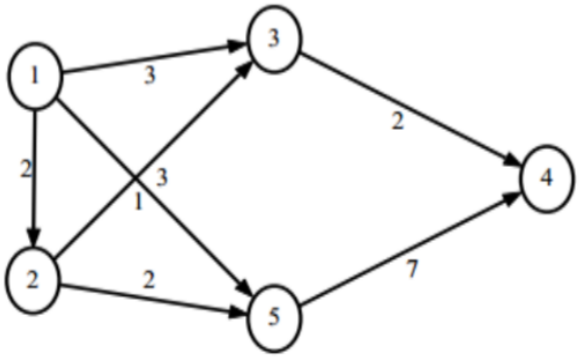
Tant que (il existe x n'appartenant pas à S dont tous ses prédécesseurs sont dans S) Faire
$\pi(x) \leftarrow min_{e\,:\Gamma(e)=x}[\pi(I(\epsilon)) + d(\epsilon)]$.

Soit u un arc pour lequel $\pi(x) = \pi(I(u))\,\pi + \pi\ d(u))$.
$A(x) \leftarrow$ u ; S ← S ∪ {x} ;
Fin tant que

Fin algorithme

Exemple 1.1.

Soit le réseau $R = (X, E, d)$ de la figure suivante :



Initialisation :

Soit 1 un sommet de X. On pose : $S=\{1\}$; $\Pi(1)=0$; $A=\emptyset$.

Itération 1 :

Le sommet 2 est hors de S et tous leurs prédécesseurs sont dans S, alors

$\Pi(2)= \Pi(1)+d(1,2)=0+2=2 \Rightarrow$ u=(1,2)

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

Itération 2 :

Les sommets 3 et 5 sont hors de S et tout leurs prédécesseurs sont dans S, alors

$\Pi(3)(x) = \text{Min}\{\Pi(1) + d(1,3); \Pi(2) + d(2,3)\}$.

$\Pi(3)(x)= \text{Min}\{0+3; 2+3\} = \text{Min}\{3;5\}=3 \Rightarrow$ u=(1,3)

$\Pi(5)(x) = \text{Min}\{\Pi(1) + d(1,5); \Pi(2) + d(2,5)\}$.

$\Pi(5)(x)= \text{Min}\{0+1; 2+2\} = \text{Min}\{1;4\}=1 \Rightarrow$ u=(1,5)

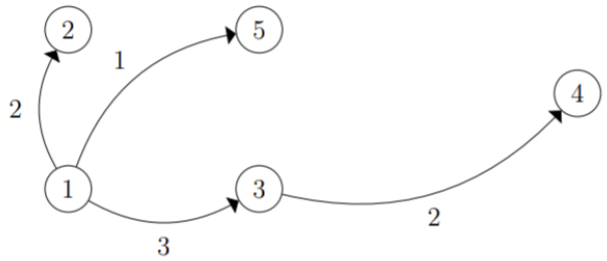
$S\cup\{3,5\} = \{1,2,3,5\}$; A= $A\cup\{(1,3),(1,5)\}= \{(1,2),(1,3),(1,5)\}$

Itération 3 :

Le sommet 4 est hors de S et tout leurs prédécesseurs sont dans S, alors

$\Pi(4)(x) = \text{Min}\{\Pi(3) + d(3,4); \Pi(5) + d(5,4)\}$.

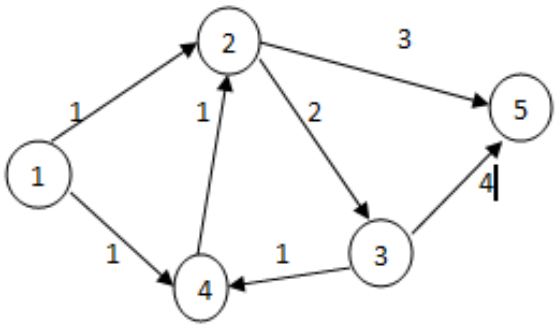
$\Pi(3)(x)$= Min $\{3+2; 1+7\}$ = Min $\{5;8\}$=5$\Rightarrow$ u=(3,5)
$\Pi(5)(x)$ = Min $\{\Pi(1) + d(1,5); \Pi(2) + d(2,5)\}$.
$S \cup \{4\} = \{1,2,3,4,5\}$; A= $A \cup \{(3,4)\}$= $\{(1,2),(1,3),(1,5),(3,4)\}$ On a : $|S| = |X|$; Terminer.
On obtient donc, l'arborescence de la figure suivante :



Exemple 1.2.

Soit le réseau $R = (X, E, d)$ de la figure suivante :

bellman : le cas d'un circuit :



Soit 1 un sommet de X. On pose : S=$\{1\}$; $\Pi(1)$=0; $A = \emptyset$.

Itération 1 :

Le sommet 2 est hors de S et tous les prédécesseurs ne sont pas dans S.
Le sommet 4 est hors de S et tous les prédécesseurs ne sont pas dans S.
Alors, l'algorithme elle est sorte.
Donc, Bellman ne fonctionne pas. car il existe un circuit.

1.2 La Complexité de bellman

La complexité de l'algorithme de bellmman est : $O(m)$.

1.3 Algorithme de dijkstra

Cet algorithme détermine les plus courts chemin d'un point s à tous les autres points d'un réseau $R = (X, E, d)$ où les longueurs des arcs sont positives ou nulles.[1] $(d(e) \geq 0 \forall e \in E)$.

1.4 Historique

Cet algorithme est dû à Edsger W. Dijkstra (1930-2002), qui est l'un des trois ou quatre informaticiens les plus importants du XXe siècle : il reçut, en 1972, le prestigieux Turing Award, équivalent du Prix Nobel ou de la médaille Fields. Il publia l'algorithme décrit ici en 1959 (« A Note on Two Problèmes in Connexion with Graphs », Numerische Mathematik, vol.1, PP. 269-271, 1959), l'année où il soutint sa thèse à l'université d'Amsterdam. Un excellent article (en anglais), téléchargeable en PDF, résume ses nombreuses contributions.

Le principe :

L'idée de l'algorithme de Dijkstra est de calculer de proche en proche, l'arborescence des plus courtes distances, issue du sommet s à un sommet donné p.
Une particularité de cet algorithme est que les distances s'introduisant dans l'ordre croissant.

Énoncé :

Données : Un réseau $R = (X, E, d)$ avec $(d(e) \geq 0 \quad \forall e \in E)$
Résultat : Arborescence des plus courtes distances A.

Algorithme de Dijkstra:

Entrée: (réseau R=(X,U,d), sommet s)

Sortie: (potentiel π, arborescence A et l'ensemble des sommets S)

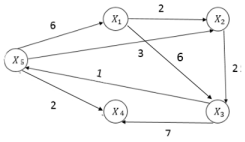
ON suppose d(u) ≥ 0 ∀ u ∈ U

Initialisation: S ← { s } , π(s)=0 , A(s) ← nil .K= 1 ; x_1=S

Pour tout $x \in X$, **Faire** $\pi(x) \leftarrow +\infty$ **fin pour tout**

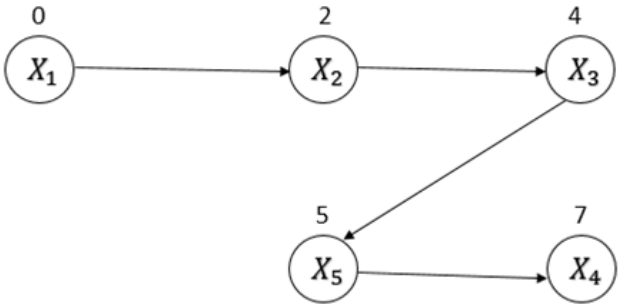
Tant que (K < n et $\pi(x_k) < \infty$) **Faire**
  Pour tout u ∈ U tel que I(u) = x_k et T(u) ∉ S faire
    $x \leftarrow$ T(u) ;
    Si $\pi(x) > \pi(x_k)$ + d(u) **alors :**
      $\pi(x) \leftarrow \pi(x_k)$ + d(u) ; $A(x) \leftarrow$ u ;
    Fin si
  Fin pour
  Choisir $x \notin$ S tel que $\pi(x) = min_{y \notin S}\, \pi(y)$
  K ← k+1 ; $x_k \leftarrow x$; S ← S ∪ {x_k} ;
Fin tant que
Fin algorithme

Exemple 1.3. *Soit le réseau* $R = (X, E, d)$ *de la figure suivante :*



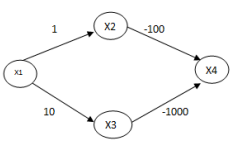
•	x_i	S	$\Pi(1)$	$\Pi(2)$	$\Pi(3)$	$\Pi(4)$	$\Pi(5)$
S	$\{x_1\}$	$\{x_1\}$	0	2	6	∞	∞
S	$\{x_2\}$	$\{x_1, x_2\}$	0	2	4	∞	∞
S	$\{x_3\}$	$\{x_1, x_2, x_3\}$	0	2	4	11	5
S	$\{x_5\}$	$\{x_1, x_2, x_3, x_5\}$	0	2	4	7	5
S	$\{x_4\}$	$\{x_1, x_2, x_3, x_5, x_4\}$	0	2	4	7	5

Les $\Pi(X)$ sont les plus courtes distances. L'arborescence des plus courts chemins, issue du sommet X_1 dans le réseau est la suivante :



Exemple 1.4.

Soit le réseau $R = (X, E, d)$ de la figure suivante : cas des coûts négatifs.



Regardons ce qui se passe si on utilise notre algorithme pour trouver le plus court chemin entre le nœud 1 et 4 :

itération	x_i	$\pi(x_i)$	choix
1	1	0	0 + 1 (1 vers 2) ou 0 + 10 (1 vers 3)
2	2	1	1 - 100 (2 vers 4) ou 0 + 10 (1 vers 3)
3	4	-99	

alors l'algorithme de dijkstra ne donne pas ici le bon résultat, donc, l'algorithme ne marche pas quand le graphe contient des arcs dont les coûts sont négatifs.

On voit bien que le chemin trouvé a une distance totale de -99, cependant on peut faire mieux en empruntant le chemin suivant : 1, 3,4 pour une distance de -990. L'algorithme ne voit pas le nœud 3 comme intéressant car l'arc permettant d'y accéder a un poids de 10 alors que sur le chemin actuel notre poids est bien inférieur à ce dernier, du coup on n'atteindra jamais l'arc de -1000 séparant le nœud 3 et 4 qui offre un chemin optimal jusqu'à l'arrivée.
L'algorithme de Dijkstra ne fonctionne donc pas sur ce genre de graphe, et ceci pour une simple raison : c'est un algorithme dit glouton. Cela signifie qu'il va chercher à faire des choix locaux optimaux (c'est-à-dire à un instant t bien précis), pour espérer trouver un choix global optimal aussi. Dans notre cas , on cherche à emprunter à un instant t le chemin avec le poids le plus faible possible (tout en considérant la distance déjà parcourue), pour espérer tomber sur le nœud d'arrivée avec une distance minimale (ce qui est démontrable).
L'algorithme se base donc sur le fait que rajouter un arc ne peut jamais améliorer le chemin (puisque les poids sont forcément positifs ou nuls), et il ne peut donc pas fonctionner avec des poids négatifs (qui eux peuvent dans certains cas améliorer le coût total).

1.5 La Complexité de dijkstra

La complexité de l'algorithme de dijkstra est : $O(n^2)$.

1.6 Algorithme de bellman ford

Énoncé :

Données : Un réseau $R = (X, E, d)$ avec $d(e) \in R$.
Résultat : Arborescence des plus courtes distances A.

Algorithme de Bellman-Ford(G ,C, s)

```
pour tout (sommet v ∈ V) faire
    Initialisation : d(v) ← ∞, A(v)← NIL , d(s)← 0
    circuit ← Faux
Fin pour
trouve ← 0 , i = 1
while ( (i < |V| - 1) et trouve= 0)
    Trouve=1
    pour tout e =(u, v) ∈ E faire
        si d(v)> d(u) + c(e)alors
            d(v)= d(u) + c(e)
            A(v)← u
            trouve ← 0
        Fin si
    Fin pour
    i = i+1
Fin while
Pour e =(u, v) ∈ E faire
    si d(v)> d(u) + c(e)
        circuit ← vrai
    Fin si
Fin pour
```

Algorithme de bellman Ford :

Cet algorithme appliqué pour la recherche d'un plus court chemin sur un réseau quelconque avec $d(e) \in R$.
Algorithme de Ford permet soit :

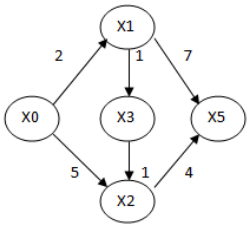
– De mettre en évidence un circuit absorbant si celui-ci existe.
– De déterminer une arborescence des plus courts chemins de racine s dans un réseau s'il ne contient pas de circuit absorbant.

Le principe :

Le principe de cet algorithme est de changer en mieux une arborescence réalisable (initiale) de racine s jusqu'à l'obtention d'une arborescence optimale des plus courts chemins, issue de s si celle-ci existe.[1]

Exemple 1.5.

Soit le réseau $R = (X, E, d)$ de la figure suivante :



$\pi(x_i)$	$\pi(x_0)$	$\pi(x_1)$	$\pi(x_2)$	$\pi(x_3)$	$\pi(x_4)$	S
Initialisation	0	∞	∞	∞	∞	/
Itération 1	0	2	5	∞	∞	x_0
	0	2	5	3	9	x_1
	0	2	5	3	9	x_2
	0	2	4	3	9	x_3
Itération 2	0	2	4	3	9	x_0
	0	2	4	3	9	x_1
	0	2	4	3	8	x_2
	0	2	4	3	8	x_3
Itération 3	0	2	4	3	8	x_0
	0	2	4	3	8	x_1
	0	2	4	3	8	x_2
	0	2	4	3	8	x_3

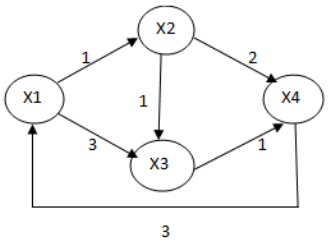
1.7 La Complexité de bellman-ford

Si le graphe comporte n sommets et m arcs, chaque arc sera relâché $n-1$ fois, et on effectuera donc au total $(n-1)m$ relâchements successifs. Si le graphe est représenté par des matrices d'adjacence, on aura une complexité en $O(n^3)$, alors que s'il est représenté par des listes d'adjacence, on aura une complexité en $O(nm)$.

1.8 Algorithme de Floyd- Warshall

C'est un algorithme en $O(n^3)$, très efficace pour déterminer la matrice des plus courtes distances dans un réseau $R=(X,U,C)$ il a été proposé par Broy en 1962, puis publié dans les articles de Floyd et warshall.
l'algorithme consiste à : pour chaque paire de sommets $(i,j)\in X^2$ de calculer de proche en proche la plus courte distance de i à j en considérant que le chemin passe par un sommet intermédiaire k ($k\in\{1,2,,n\}$) ou pas.

Exemple 1.6.



Initialisation	$d_0 = \begin{pmatrix} 0 & 1 & 3 & \infty \\ \infty & 0 & 1 & 2 \\ \infty & \infty & 0 & 1 \\ 3 & \infty & \infty & 0 \end{pmatrix}$
k=1	$d_1 = \begin{pmatrix} 0 & 1 & 3 & \infty \\ \infty & 0 & 1 & 2 \\ \infty & \infty & 0 & 1 \\ 3 & 4 & 6 & 0 \end{pmatrix}$
k=2	$d_2 = \begin{pmatrix} 0 & 1 & 2 & 3 \\ \infty & 0 & 1 & 2 \\ \infty & \infty & 0 & 1 \\ 3 & 4 & 5 & 0 \end{pmatrix}$
k=3	$d_3 = \begin{pmatrix} 0 & 1 & 2 & 3 \\ \infty & 0 & 1 & 2 \\ \infty & \infty & 0 & 1 \\ 3 & 4 & 5 & 0 \end{pmatrix}$
k=4	$d_4 = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 5 & 0 & 1 & 2 \\ 4 & 5 & 0 & 1 \\ 3 & 4 & 5 & 0 \end{pmatrix}$
k=5	$d_5 = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 5 & 0 & 1 & 2 \\ 4 & 5 & 0 & 1 \\ 3 & 4 & 5 & 0 \end{pmatrix}$

La dernière matrice d_5 représente la distance de chemin la plus courte entre chaque paire de sommets.

2 Résultats numériques

voir le tableau suivant :

problème	n(nombre des sommets)	m(nombre des arcs)	Temp de bellman	Temp de dijkstra	Temp de Bellman ford	Temp de Floyd-walshall
Pmed1	100	200	/	0.0096	0.0043	4.6085
Pmed2	100	200	/	0.0098	0.0041	4.8614
Pmed3	100	200	/	0.0104	0.0045	4.8744
Pmed4	100	200	/	0.0087	0.0043	3.3408
Pmed5	100	200	/	0.0085	0.0040	4.6169
Pmed6	200	800	/	0.0168	0.0652	18.4948
Pmed7	200	800	/	0.0177	0.0037	18.5852
Pmed8	200	800	/	0.0160	0.0617	21.1345
Pmed9	200	800	/	0.0143	0.0640	17.5792
Pmed10	200	800	/	0.0154	0.0662	18.6757
Pmed11	300	1800	/	0.0285	0.4215	59.7180
Pmed12	300	1800	/	0.0332	0.3710	56.6736
Pmed13	300	1800	/	0.0362	0.4262	64.9163
Pmed14	300	1800	/	0.0334	0.3292	69.7173
Pmed15	300	1800	/	0.0291	0.4033	96.7740
Pmed16	400	3200	/	0.0482	0.9881	170.1322
Pmed17	400	3200	/	0.0481	1.0338	182.1604
Pmed18	400	3200	/	0.0519	0.6328	174.3518
Pmed19	400	3200	/	0.0595	0.9177	180.0729
Pmed20	400	3200	/	0.0538	1.0192	162.5307
Pmed21	500	5000	/	0.0713	2.0786	284.2884
Pmed22	500	5000	/	0.0754	2.1048	294.8196
Pmed23	500	5000	/	0.0760	2.1861	326.8718
Pmed24	500	5000	/	0.0755	2.1470	297.7418
Pmed25	500	5000	/	0.0825	2.1317	297.6286
Pmed26	600	7200	/	0.1062	3.7886	509.7163
Pmed27	600	7200	/	0.1191	3.8094	469.4410
Pmed28	600	7200	/	0.0977	3.9768	470.9297
Pmed29	600	7200	/	0.1040	3.6737	426.8859
Pmed30	600	7200	/	0.1043	3.8451	444.2237
Pmed31	700	9800	/	0.1291	9.9110	665.0813
Pmed32	700	9800	/	0.1275	6.1927	668.3568
Pmed33	700	9800	/	0.1323	6.4443	891.9482
Pmed34	700	9800	/	0.1350	6.8600	929.8674
Pmed35	800	12800	/	0.1862	9.1186	1.3658e+03
Pmed36	800	12800	/	0.1807	9.0308	1.3802e+03
Pmed37	800	12800	/	0.1854	9.7745	1.2032e+03
Pmed38	900	16200	/	0.2116	15.3607	1.7174e+03
Pmed39	900	16200	/	0.1946	18.1145	1.4801e+03
Pmed40	900	16200	/	0.1932	14.5941	1.6659e+03

- Interprétation des résultats

Comme le montre le tableau des tests sur les exemples de OR-library, on remarque que tous les exemples contiennent des circuits et par conséquent on ne peut pas appliquer l'algorithme de Bellman même s'il est théoriquement efficace. cela explique pourquoi on a le plus recours à l'algorithme de Dijkstra et surtout à celui de Bellman-Ford.
Enfin le dernier algorithme présenté par Floyd et Warshall peut s'avérer très efficace quand on souhaite obtenir toutes la matrice

des plus courtes distances.

3 Introduction

Il arrive que, dans un réseau routière par exemple, l'on veuille déterminer un plus court chemin de s à p qui ne passe pas par un certain nombre d'arc (pour le réseau routière) éviter de passer par exemple par les autoroutes peut être souhaitable car elle sont payants ou bien éviter des route dangereuse ou déterminer un plus court chemin qui existe un certain nombre de sommets (pour le réseau routière cela consiste à éviter des villes réputées être dangereuses ou faisant l'objet de trafic intense).

4 Conception du GPS

Pour concevoir un GPS, outil oh combien nécessaire dans la vie moderne, le principe est simple :
Il s'agit d'adopter n'importe quel algorithme parmi ceux qu'on a cité et, selon les demandes, si par exemple on veut éviter d'emprunter les autoroutes car elles sont payantes cela se traduit par affecter un cout infini à tout arc représentant une autoroute ou, au contraire si on veut aller vite (sans regarder la dépense) et n'utiliser que les autoroutes on devrait travailler uniquement sur le graphe partiel construit surs les arcs représentants les autoroutes.
Il arrive assez souvent que l'on veuille éviter de passer par certaines villes (sources de trafic intense, ou de dangerosité comme les manifestations). Là aussi il suffit de travailler sur le sous-graphe construit sur l'ensemble des sommets réputés sures.

4.1 Conclusion

Nous avons implémenté toutes les procédures en langage Matlab et on peut affirmer l'efficacité de l'algorithme de Dijkstra et surtout la procédure de Bellman-Ford.
L'algorithme du à Floyd et Warshall, en plus de sa simplicité s'est avéré très efficace pour la détermination de la matrice des plus courtes distances.
Nous pensons pouvoir l'optimiser en évitant certains tests inutiles.
Faute de temps et vu les circonstances de pandémie, nous n'avons pas pu terminer notre mémoire par la confection d'un logiciel de GPS. nous nous attelons fortement à cette tache dans un proche avenir.

Bibliographie

[1] Amichi,Djamila, and Rebiha Rahmani. Application de Quelques méthodes de recherche opérationnelle sur les circuits de production cas ENIEM. Diss. UMMTO,2019.

[2] El Malki, Mohammed, et al. "Interrogation de données hétérogènes dans les systèmes NoSQL orientés graphes." (2018) : 179 – 194.

[3] Hélary, Jean-Michel. "Le plus court chemin." Interstices (2005).

[4] Zemmouk, Salim, and Hocine Korichi. Ordonnancement et optimisation d'un processus de fabrication au sein de l'entreprise ENIEM. Diss. UMMTO,2017.