Chapitre 4

Problème de cheminement dans un graphe

4.1 Introduction

Les problèmes de cheminement dans les graphes (en particulier la recherche d'un plus court chemin) comptent parmi les problèmes les plus anciens de la théorie des graphes et les plus importants par leurs applications.

On peut citer, entre autres : les problèmes de tournées, Problèmes d'investissement et de gestion de stock, les problèmes d'optimisation de réseaux (réseaux routiers et réseaux de télécommunications), problèmes d'intelligence artificielle et reconnaissance des formes, méthodes de traitement numérique du signal et problèmes d'ordonnancement.

Par exemple : Il a d'ailleurs donné lieu aux développements de sites Internet qui se proposent de déterminer pour vous le meilleur itinéraire que ce soit en distance, en temps ou en coût. Il suffit de taper "recherche d'itinéraire" sur un navigateur pour s'en convaincre.

On imagine aisément que ce type de problème est modélisable par un graphe et par la recherche sur ce graphe du "meilleur chemin".

Le problème posé est de déterminer dans un graphe value (orienté ou non orienté) le plus court chemin (chemin de longueur ou de coût minimum).

Définition 4.1 (Graphes valués) :

Un graphe G = (X,U) est valué s'il est muni d'une application :

$$F : U \rightarrow R$$
$$(x, y) \rightarrow c(x,y)$$

Appelée valuation, on notera G = (X, U, c) un graphe muni d'une valuation c.

La valuation d'un arc ou d'une arête peut représenter une distance, un temps, un coût, . . .

A partir de la valuation d'un arc/arête, on peut facilement définir la valuation (longueur) d'un chemin/une chaîne : c'est la somme des valuations des arcs/arêtes qui composent ce chemin/chaîne.

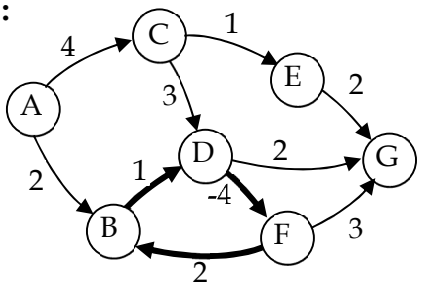
Définition 4.2 (Longueur d'un chemin) :

La longueur (coût) d'un chemin C dans un graphe orienté (non orienté) est égale à la somme des coûts des arcs (arêtes) comptés dans leur ordre de multiplicité dans le chemin (le nombre de fois qu'on parcourt ces arcs-arêtes-), on le note **w(C)**.

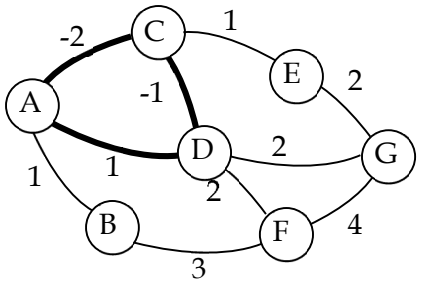
Définition 4.3 (Circuit, cycle absorbant) :

On appelle circuit (cycle) absorbant un circuit (cycle) de coût strictement négatif.

Exemple :



Circuit Absorbant (B,D,F) de coût 1-4+2=-1



Cycle Absorbant (A,D,C) de coût -2-1+1=-2

FIG 4.1 Circuit (cycle) absorbant

La présence d'un circuit (cycle) absorbant qui entraîne une diminution de la longueur du chemin.

S'il existe un circuit (cycle) absorbant, il est clair qu'il n'existe pas de plus court chemin entre toute paire de sommets. Inversement, s'il existe un plus court chemin alors il n'existe pas de circuit (cycle) absorbant

Il existe 3 types de problèmes :

- **Problème A :** x et y deux sommets donnés de G = (X, U, c), il s'agit de trouver un plus court chemin de x à y.
- **Problème B :** x étant fixé, il s'agit de trouver un plus court chemin de x à n'importe quel sommet y, $y \in X$.
- **Problème C :** il s'agit de trouver un plus court chemin entre toute paire de sommets.

Le problème A est plus difficile que le problème B, donc on ne résout jamais directement le problème A, on passe par la résolution du problème B.

De nombreux problèmes concrets se traduisent par la recherche d'un chemin de longueur minimale, ces problèmes a de nombreuses applications pratiques car le coût c_{ij} peut s'interpréter aussi bien comme un coût de transport sur l'arc/arête (x,y), comme les dépenses de construction de l'arc/arête (x,y), comme le temps nécessaire pour parcourir de l'arc/arête (x,y), etc.

Selon les propriétés du graphe traité et selon le problème considéré, il existe de nombreux algorithmes permettant l'obtention d'une solution, parmi eux on peut citer :

Algorithme	Type de Problème	Hypothèses
Bellman	d'un sommet à tous les autres	- Graphe sans circuit
Dijkstra	d'un sommet à tous les autres	- Longueurs positives
Ford	d'un sommet à tous les autres	
Dantzig	entre tous les couples de sommets	- Graphe sans circuit absorbant
Floyd	entre tous les couples de sommets	

L'algorithmes étudié ici est celui de Bellman–Ford–Moore.

4.2 Algorithme de Bellman–Ford–Moore

L'algorithme de Ford, aussi appelé algorithme de Bellman–Ford–Moore, est un algorithme qui calcule des plus courts chemins depuis un sommet à tous les autres sommets

4.2.1 Principe

L'idée est de parcourir tous les sommets jusqu'on ne puisse plus les améliorer et calcule des plus courts chemins depuis un sommet $\mathbf{x_0}$ à tous les autres sommets ou les résultats seront stockés de la façon suivante :

- Le tableau **d** contient les plus courts chemins de $\mathbf{x_0}$ à chaque sommet du graphe x_i.

- Le tableau **Pred** contient les prédécesseurs de chaque sommet dans le plus court chemin à partir de x_0.

Contrairement à les autres algorithme, l'algorithme de Ford est applicable sur tous les types de graphes c.à.d. il autorise la présence de certains arcs/arêtes de poids négatif et permet de détecter l'existence d'un circuit (cycle) absorbant, accessible depuis le sommet source.

4.2.2 Algorithme

Algorithme **Ford (**G : graphe ; x_0 : sommet**)**
Entrées : G(X,U, c) un graphe orienté valué d'ordre n, et c_{ij} longueur de l'arc (i,j).
Sorties : les plus courts chemins entre le sommet $\mathbf{x_0}$ et tout autre sommet.
Début
Etape 1 : (initialisation)
  Pour i=1 à n **faire**
    $d[x_i] \leftarrow +\infty$;
    $Pred[x_i] \leftarrow$null ;
  FPour
  $d[x_0] \leftarrow 0$;
Etape 2 : (Traitement)
  Pour k=1 à n-1 **faire**
    Pour $(u(x_i,x_j) \in U)$ **faire**
      Si $(d[x_j]>d[x_i]+c_{ij})$ **Alors**
        $d[x_j] \leftarrow d[x_i]+c_{ij}$; $Pred[x_j] \leftarrow x_i$;
      FSi
    FPour
  FPour
  Pour $(u(x_i,x_j) \in U)$ **faire**
    Si $(d[x_j]>d[x_i]+c_{ij})$ **Alors**
      Fin (Pas de plus court chemin, il existe cycle (circuit) absorbant)
    FSi
  FPour
Fin

4.2.3 Exemple 1(Sans circuit absorbant)

Appliquons l'algorithme de Ford sur le graphe G de la figure 4.2 suivante afin d'obtenir les plus courts chemins de **A** vers les autres sommets

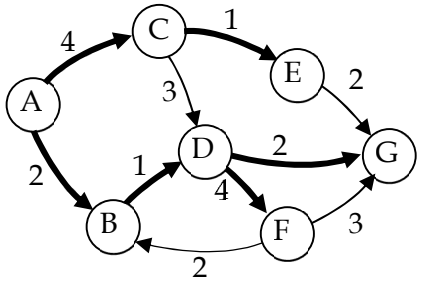


FIG 4.2 Exemple d'application algorithme Ford

U={(B,D), (C,E),(C,D),(D,G),(D,F),(E,G),(F,G),(F,B), (A,C),(A,B)}

	d							Pred						
Itérations	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**A**	**B**	**C**	**D**	**E**	**F**	**G**
Init	0	+∞	+∞	+∞	+∞	+∞	+∞	/	/	/	/	/	/	/
It1 (k=1)	0	2	4	+∞	+∞	+∞	+∞	/	A	A	/	/	/	/
It2 (k=2)	0	2	4	3	5	7	5	/	A	A	B	C	D	D
It3 (k=3)	0	2	4	3	5	7	5	/	A	A	B	C	D	D
It4 (k=4)	0	2	4	3	5	7	5	/	A	A	B	C	D	D
It5 (k=5)	0	2	4	3	5	7	5	/	A	A	B	C	D	D
It6 (k=6)	0	2	4	3	5	7	5	/	A	A	B	C	D	D

L'arborescence des plus courts chemins est celle composée de toutes les arêtes (arcs) en gras de la dernière itération et la valeur d[x] de chaque sommet représente la longueur du plus court chemin entre le sommet de départ x_0=A et le sommet considéré x≠A.

4.2.4 Exemple 2 (Avec circuit absorbant)

Appliquons l'algorithme de Ford sur le graphe G de la figure 4.3 suivante afin d'obtenir les plus courts chemins de **A** vers les autres sommets

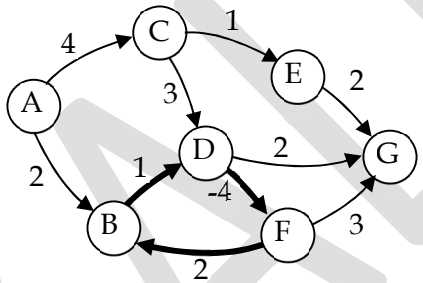


FIG 4.3 Exemple d'application algorithme Ford

U={(B,D), (C,E),(C,D),(D,G),(D,F),(E,G),(F,G),(F,B), (A,C),(A,B)}

	d							Pred						
Itérations	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**A**	**B**	**C**	**D**	**E**	**F**	**G**
Init	0	+∞	+∞	+∞	+∞	+∞	+∞	/	/	/	/	/	/	/
It1 (k=1)	0	2	4	+∞	+∞	+∞	+∞	/	A	A	/	/	/	/
It2 (k=2)	0	1	4	3	5	-1	2	/	F	A	B	C	D	F
It3 (k=3)	0	0	4	2	5	-2	1	/	F	A	B	C	D	F
It4 (k=4)	0	-1	4	1	5	-3	0	/	F	A	B	C	D	F
It5 (k=5)	0	-2	4	0	5	-4	-1	/	F	A	B	C	D	F
It6 (k=6)	0	-3	4	-1	5	-5	-2	/	F	A	B	C	D	F

Pour u(B,D) on d[D]>d[B]+c_{BD} c.ad -1>-3+1 donc pas de plus court chemin et qu'il existe un circuit absorbant (B, D, F)