Chapitre 3

Problèmes du plus court chemin

3.1 Introduction

Les problèmes de chemins optimaux sont très fréquents dans les situations de tous les jours. On les rencontre dès qu'il s'agit de rechercher des chemins entre deux points d'un réseau, de façon à minimiser un coût, une durée, etc. Ils apparaissent aussi comme des sous-problèmes à de nombreux problèmes combinatoires notamment dans les problèmes d'ordonnancement.

Plusieurs algorithmes permettent de résoudre efficacement ce problème. Il existe, néanmoins, une version plus élaborée appelée "problème du voyageur de commerce". Dans ce dernier, le but est de construire un chemin ou un circuit hamiltonien ayant un coût minimal. Malgré la similarité apparente, le problème du voyageur de commerce est plus complexe à résoudre et ne sera pas traité dans ce chapitre.

Il faut, néanmoins, veiller à différencier le problème du plus court chemin de la construction de l'arbre de couverture de poids minimal. Dans la plupart des cas, la construction de cet arbre n'implique pas que les chemins soient minimaux.

3.2 Définitions

3.2.1 Notion de réseau

Un réseau est un graphe orienté $G = (X, E)$ muni d'une fonction $d : E \leftarrow \mathbb{R}$ qui associe à chaque arc e sa longueur $d(e)$. On note un tel réseau par $R = (X, E, d)$ (il s'agit alors d'un graphe valué dont les valeurs sont des réels).

En pratique, $d(e)$ peut matérialiser un coût de transport, une distance, une durée, etc. Il peut également s'agir d'une récompense dans le cas d'une valeur négative. La figure 3.1 montre un exemple

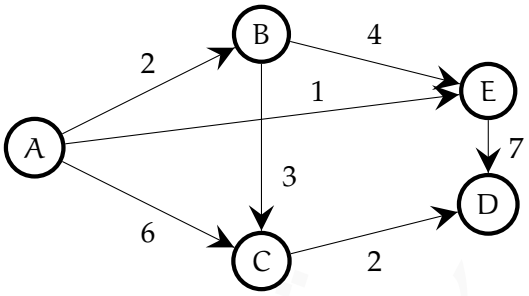

Figure 3.1 : un exemple d'un réseau

d'un réseau.

3.2.2 Coût d'un chemin

Le coût d'un chemin dans un réseau R est égal à la somme des coûts des arcs comptés dans leur ordre de multiplicité (le nombre de fois que l'on passe par ces arcs).

Le problème du plus court chemin consiste à calculer des chemins de coût minimal. Dans le réseau de la figure 3.1, le plus court chemin du sommet A au sommet D est le chemin (A, B, C, D) de coût $L = d(A, B) + d(B, C) + d(C, D) = 2 + 3 + 2 = 7$. Dans ce chemin, l'ordre de multiplicité de chaque arc est de 1.

3.2.3 Circuit absorbant

Un circuit est dit *absorbant* si la somme des coûts de ses arcs est négative. Les chemins de coût minimal n'ont un sens que si le réseau n'a pas de circuit absorbant. En réalité, si un tel circuit existe, alors on peut diminuer le coût d'un chemin indéfiniment en tournant en rond dans ce circuit (le coût minimal sera $-\infty$).

Dans la figure 3.2, le coût du circuit (A, B, C, A) est de -1, c'est donc un circuit absorbant.

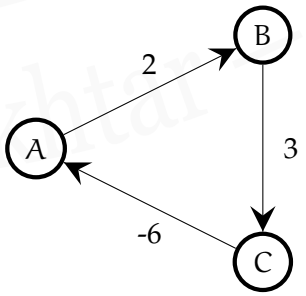

Figure 3.2 : exemple d'un circuit absorbant

3.2.4 Problème considéré

En absence de circuit absorbant, on peut restreindre la recherche du plus court chemin aux seuls chemins élémentaires (rappel : chaque sommet apparaît une seule fois). On distingue 3 types de problèmes notés ici par : A, B et C :

A : trouver un plus court chemin entre deux sommets particuliers x et y dans $R(X, E, d)$.

B : trouver un plus court chemin entre n'importe quels deux sommets x et y du réseau.

C : trouver un plus court chemin entre un sommet x (de départ) et tous les sommets y du réseau $R(X, E, d)$.

Ces problèmes sont en réalité liés. Par exemple, un algorithme pour **A** peut être appliqué plusieurs fois pour résoudre **B** ou **C**. On s'intéressera, dans la suite, au problème **C**. Ce dernier admet une solution si :

— x est une racine du réseau.
— Le réseau ne contient pas de circuit absorbant.

La solution sera, en fait, une arborescence des plus courts chemins admettant x comme racine.

3.3 Algorithme de Dijkstra

3.3.1 Conditions

L'algorithme de Dijkstra s'applique lorsque les poids sont tous positifs, ce qui exclut bien sûr la présence des circuits absorbants.

3.3.2 Les algorithmes utilisés

Étant donné la racine x du réseau, l'algorithme de Dijkstra consiste à construire pour chaque sommet y, le coût λy représentant le coût minimal du chemin reliant x à y.

Algorithme de Dijkstra pour trouver le plus court chemin entre x_0 (la racine du graphe) et les autres sommets d'un réseau $R = (X, E, d)$

1 initialiser un ensemble S à x_0 ($S = \{x_0\}$)
2 $\lambda x_0 = 0$: le coût du chemin entre x_0 et x_0 est nul
3 **pour chaque** *sommet* x_i *successeur de* x_0 **faire**
4 mettre $\lambda x_i = d(x_0, x_i)$
5 **pour chaque** *sommet* x_j *non successeur de* x_0 **faire**
6 mettre $\lambda x_j = \infty$
7 **tant que** $S \neq X$ **faire**
8 choisir un sommet $x_k \notin S$ tel que $\lambda x_k = \min_{x_l \notin S} \lambda x_l$
9 $S = S \cup \{x_k\}$
10 **pour chaque** $x_m \in (\Gamma^+(x_k) - S)$ **faire**
`// les successeurs de` x_k `qui ne sont pas dans S`
$\lambda x_m = \min(\lambda x_k + d(x_k, x_m), \lambda x_m)$

Nous allons d'abord calculer les coûts minimaux des chemins entre le sommet A et les autres sommets du graphe de la figure 3.3. Nous construirons ensuite l'arborescence couvrant les chemins minimaux.

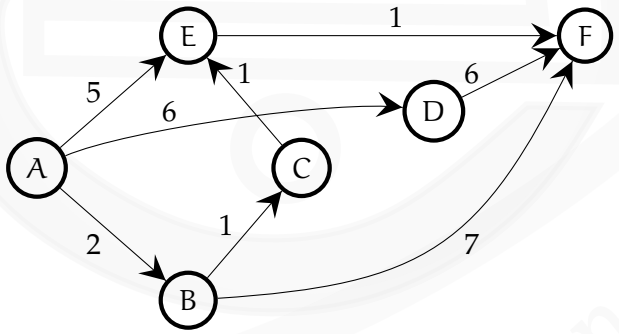

Figure 3.3 : calculer les plus courts chemins à partir de A

S	λA	λB	λC	λD	λE	λF
A	**0**	2	∞	6	5	∞
A, B	0	**2**	3	6	5	9
A, B, C	0	2	**3**	6	4	9
A, B, C, E	0	2	3	6	**4**	5
A, B, C, E, F	0	2	3	6	4	**5**
A, B, C, E, F, D	0	2	3	**6**	4	5

La dernière ligne de la table donne les coûts minimaux des chemins reliant la racine à tous les sommets du réseau. Pour construire l'arborescence des chemins optimaux dans le réseau, l'algorithme suivant est utilisé :

Algorithme de l'arborescence des chemins optimaux d'un réseau $R = (X, E, d)$

1 on part de la racine

2 on retient tout arc (x_i, x_j) tel que $\lambda x_i - \lambda x_j = d(x_i, x_j)$

On obtient alors le graphe de la figure 3.4 où les arcs en rouge représentent l'arborescence des chemins minimaux. En d'autres termes, c'est l'arborescence qui permet de trouver le chemin minimal entre A et n'importe quel sommet du réseau. Rappelons que, dans une arborescence, il existe un seul chemin entre deux sommets quelconques.

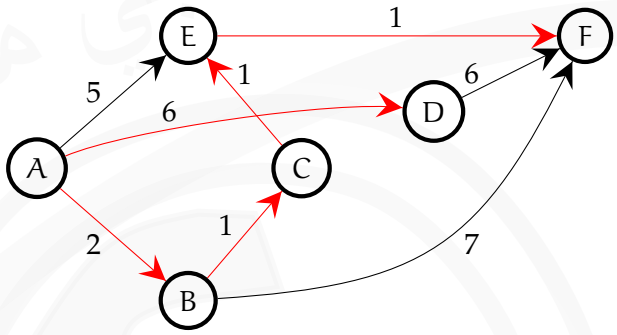

Figure 3.4 : arborescence des chemins optimaux A

3.4 Algorithme de Bellman

3.4.1 Conditions

L'algorithme de Bellman s'appelle plutôt Bellman-Ford, mais nous utilisons ici uniquement "Bellman" pour le désigner afin d'éviter toute confusion avec les algorithmes de flot. On peut définir plusieurs versions de cet algorithme, le point commun est qu'ils s'appliquent aussi à des graphes ayant des arcs à poids négatifs (à condition que le graphe ne contienne pas de circuits absorbants).

Dans ce chapitre, on considère uniquement une version de l'algorithme de Bellman s'appliquant à un graphe sans circuit.

3.4.2 Les algorithmes utilisés

L'algorithme de Bellman, présenté ici, est basé sur l'algorithme de classement des sommets d'un graphe orienté. Il utilise ensuite les rangs de chaque sommet pour établir les plus courts chemins.

Algorithme de Bellman pour trouver le plus court chemin entre la racine x_0 et les autres sommets d'un réseau $R = (X, E, d)$ sans circuit

1 ordonner le graphe selon l'algorithme vu en premier chapitre
2 $n = ordre(G)$
3 pour tout sommet x de rang 1, mettre $\lambda x = 0$
4 **pour** $j = 1..n$ **faire**
5 $\quad \lambda x_j = min(\lambda x_i + d(x_i, x_j))$ avec $x_i \in \Gamma^-(x_j)$

Appliquons l'algorithme au graphe de la figure 3.3. Le classement des sommets du graphe est donné par la figure 3.5.

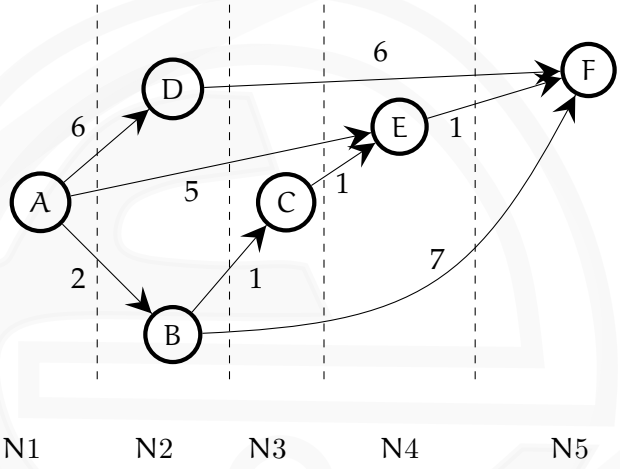

Figure 3.5 : classement des sommets du réseau

L'application de l'algorithme se fait comme suit (les sommets ont été ordonnés selon leurs rangs) :

Coût minimal	**Calcul**
λA	0 (c'est la racine)
λB	$\lambda A + d(A, B) = 2$
λD	$\lambda A + d(A, D) = 6$
λC	$\lambda B + d(B, C) = 3$
λE	$min(\lambda A + d(A, E), \lambda C + d(C, E)) = min(5, 4) = 4$
λF	$min(\lambda E + d(E, F), \lambda D + d(D, F), \lambda B + d(B, F)) = min(5, 12, 9) = 5$

La construction de l'arborescence des chemins optimaux se fait de la même façon que l'algorithme de Dijkstra.