Résolution du problème du plus court chemin

Les problèmes de plus court chemin apparaissent naturellement comme modélisation de problèmes opérationnels (trajet le plus rapide, ou le moins cher, gestion optimale de stock...). On suppose dans la suite qu'on dispose d'un graphe orient G = (V, A) pondéré par une fonction poids p : A → R. Les poids peuvent représenter des longueurs, des durées, mais aussi des coûts ou des bénéfices. C'est pour cela qu'ils peuvent prendre des valeurs positives ou négatives. Dépendamment du signe de p et de l'existence ou pas de circuits dans le graff, on dispose de différents algorithmes. Le poids d'un chemin est la somme des poids de ses arcs. On cherche le chemin de poids le plus petit.

1 Les poids sont positifs : Algorithme de Dijkstra

On suppose dans cette section que la fonction p prend des valeurs strictement positives. On considère un sommet de départ a. L'algorithme de Dijkstra va donner le poids du chemin le plus court de a à tous les autres sommets.

Durant l'algorithme, on maintiendra les variables suivantes :

- Q: ensemble des sommets en cours de traitement.
- λ : V → R+: fonction qui à chaque sommet v donne l'évaluation courante du poids du plus court chemin de a à v.

L'algorithme.

1. Initialisation :
   - Q={a},
   - λ(a) = 0 et λ(v) = +∞ pour les autres sommets v.
2. Boucle sur les éléments v de Q : (on commence par celui qui minimise λ)
   - On retire v de Q.
   - Boucle sur tous les v' successeurs de v :
     - Si λ(v') = +∞, ajouter v' dans Q.
     - Si λ(v) + p(v, v') < λ(v'), alors λ(v') = λ(v) + p(v, v').
3. Critère d'arrêt : On s'arrête quand Q est vide. La fonction λ donne alors les poids des plus courts chemins du sommet a aux autres sommets. Un poids égal à +∞ signifie que le sommet n'est pas atteignable à partir de a.

2 Trace de l'algorithme de Dijkstra

On considère le graphe suivant.

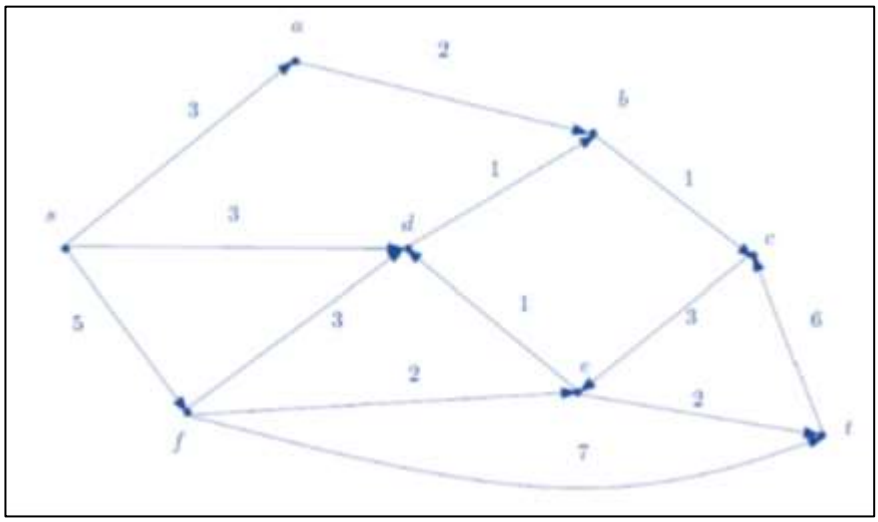

Figure 1: Graphe orienté

L'application de l'algorithme de Dijkstra à ce graphe en partant du sommet s vers les autres sommets donne :

1. Initialisation :
   - Q = {s},
   - λ(s) = 0 et λ(v) = +∞ sinon.
2. Étape 1 : v = s
   - Q=∅.
   - Successeurs(s) = {a, d, f}.
     - v' = a. λ(a) = 3. Q = {a}.
     - v' = d. λ(d) = 3. Q = {a, d}.
     - v' = f. λ(f) = 5. Q = {a, d, f}.
3. Étape 2 : v = a
   - Q = {d, f}.
   - Successeurs(a) = {b}.
     - v'= b. λ(b) = 5. Q = {d, f, b}.
4. Étape 3 : v = d
   - Q = {b, f}.
   - Successeurs(d) = {b}.
     - v' = b. λ(b) = 4.
5. Étape 4 : v = b
   - Q = {f}.
   - Successeurs(b) = {c}.
     - v' = c. λ(c) = 5. Q = {f, c}.
6. Étape 5 : v = c
   - Q = {f}.
   - Successeurs (c) = {e}.
     - v' = e. λ(e) = 8. Q = {e, f}.
7. Étape 6 : v = f
   - Q = {e}.
   - Successeurs (f) = {e, t}.
     - v' = e. λ(e) = 7.
     - v' = t. λ(t) = 12. Q = {e, t}.
8. Étape 7 : v = e
   - Q = {t}.
   - Successeurs(e) = {d, t}.
     - v' = d. λ(d) = 3 < λ(e) + 1 = 9.
     - v' = t. λ(t) = 9.
9. Étape 8 : v = t
   - Q=∅.
   - Successeurs (t) = {c}.
     - v' = c. λ(c) = 5 < λ(t) + 6 = 15.
10. Critère d'arrêt : On s'arrête car Q = ∅. Les différentes valeurs de λ sont représentées dans le tableau suivant :

s	a	b	c	d	e	f	t
(0)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)
0	(3)	(∞)	(∞)	(3)	(∞)	(5)	(∞)
0	3	(5)	(∞)	(3)	(∞)	(5)	(∞)
0	3	(4)	(∞)	3	(∞)	(5)	(∞)
0	3	4	(5)	3	(∞)	(5)	(∞)
0	3	4	5	3	(8)	(5)	(∞)
0	3	4	5	3	(7)	5	(12)
0	3	4	5	3	7	5	(9)
0	3	4	5	3	7	5	9

3 PROGRAMMATION DE L'ALGORITHME DIJKSTRA

Nous disposons d'un graphe orienté pondéré. Nous souhaitons trouver le plus court chemin entre chaque 2 sommets de ce graphe à l'aide de l'algorithme de Dijkstra. Le but de cet exercice est de programmer l'algorithme de Dijkstra sous Scilab et l'utiliser pour trouver le plus court chemin du sommet s au sommet t du graphe de la figure 1 ci-dessus.

On vous demande de :

1. Créer un répertoire de travail « PCC ». Dans ce répertoire, créer un fichier « Graphe.sce » et un fichier « Dijkstra.sce ».
2. Lancer Scilab et se placer dans le répertoire « PCC ».
3. Nombre de sommets : dans le fichier « Graphe.sce », introduire une variable n qui représente le nombre de sommets dans le graphe.
4. Arcs et poids : dans le fichier « Graphe.sce », introduire une matrice A telle que Aij = 0 si l'arc ij n'appartient pas au graphe et A ij = p(i,j) le poids de l'arc i,j si l'arc i,j appartient au graphe.
5. Sommet de départ : dans le fichier « Graphe.sce », introduire une variable s qui donne l'indice du sommet de départ.
6. Exécuter le fichier "Graphe.sce".

On implémentera dorénavant les commandes dans le fichier « Dijkstra.sce ».

7. Définir une fonction PCC qui prend en argument un graphe A et un sommet de départ s et dont le résultat est un vecteur λ qui donne les poids des plus courts chemins de s aux autres sommets du graphe. Introduire une variable n qui donne le nombre de sommets du graphe et le vecteur Q qui donne les sommets à explorer.
8. Initialisation : initialiser Q et λ.
9. Sommets à explorer : faire une boucle sur les éléments de Q tant que Q n'est pas vide. A chaque itération, définir i, l'indice du sommet de Q qui minimise λ. On peut utiliser les commandes min et find.
10. Exploration du sommet : faire une boucle sur les sommets j successeurs de i. Pour chaque j implémenter les deux commandes suivantes :
    - Si $\lambda(j) = +\infty$, $Q = Q \cup \{j\}$.
    - Si $\lambda(i) + p(i,j) < \lambda(j)$, alors $\lambda(j) = \lambda(i) + p(I,j)$.
11. Application, exécuter le fichier « Dijkstra.sce » : appliquer la fonction PPC à la matrice définie précédemment et donner le poids du plus court chemin entre s et t.
12. Question bonus : changer votre code dans le fichier « Dijkstra.sce » pour qu'il retourne le chemin (la suite des sommets) pour aller de s à un sommet donné en entrée.
13. Zipper les deux fichiers « Graphe_votrenom.sce » et « Dijskstra_votrenom.sce » dans une archive « PCC_votrenom.zip » et déposer la dans le site du cours à la fin de la séance.

Bon courage