# Plus courts chemins

Les graphes $G = (V, A)$ considérés dans ce document sont supposés connexes, orientés et une longueur $\ell(u, v)$ est associée à chaque arc $(u, v) \in A$. Le but de ce chapitre est de déterminer des plus courts chemins reliant certaines paires de sommets. Si le graphe considéré n'est pas orienté, on peut remplacer chaque arête $[u, v]$ par deux arcs $(u, v)$ et $(v, u)$, et on a alors $\ell(u, v) = \ell(v, u)$, cette longueur étant celle associée à l'arête $[u, v]$.

Dans ce qui suit, nous noterons $f_u(v)$ la longueur d'un plus court chemin reliant $u$ à $v$, et $p_u(v)$ le prédécesseur de $v$ sur un tel chemin. En traçant le graphe contenant tous les arcs $(p_u(v), v)$, pour un $u$ fixé, on obtient l'arborescence des plus courts chemins de $u$ vers les autres sommets du graphe, $u$ étant la racine d'une telle arborescence.

Soit $u$ un sommet pour lequel on désire connaître la longueur des plus courts chemins le reliant aux autres sommets du graphe. Lorsque **toutes les longueurs sont positives**, on peut utiliser l'**algorithme de Dijkstra** (1959) qui peut être décrit comme suit.

Poser $S \leftarrow \emptyset$, $f_u(u) \leftarrow 0$ et $f_u(v) \leftarrow \infty$ pour tout $v \neq u$ ;
Tant que $|S| \leq |V| - 1$ faire
    Déterminer un sommet $v \notin S$ tel que $f_u(v) \leq f_u(w)$ pour tout $w \notin S$ ;
    $S \leftarrow S \cup \{v\}$ ;
    Pour tout $w \notin S$ faire
        Si $f_u(w) > f_u(v) + \ell(v, w)$ alors $f_u(w) \leftarrow f_u(v) + \ell(v, w)$ et $p_u(w) \leftarrow v$.

La complexité de cet algorithme est $O(|A| + |V| \log |V|)$. Il ne doit pas être utilisé dans le cas où les longueurs sont quelconques, c'est-à-dire positives ou négatives.

Si toutes les longueurs sont unitaires (c'est le cas lorsque la longueur d'un chemin est son nombre d'arcs), l'algorithme de Dijkstra peut être accéléré comme suit, ce qui donne une complexité $O(|A|)$.

Poser $S \leftarrow \emptyset$, $f_u(u) \leftarrow 0$, $k \leftarrow -1$ et $f_u(v) \leftarrow \infty$ pour tout $v \neq u$ ;
Tant que $|S| \leq |V| - 1$ faire
    $k \leftarrow k + 1$ ;
    Déterminer l'ensemble $V_k$ des sommets $v$ tels que $f_u(v) = k$ et poser $S \leftarrow S \cup V_k$ ;
    Pour tout $w \notin S$ qui a au moins un prédécesseur dans $V_k$ faire
        Choisir un prédécesseur $v$ de $w$ dans $V_k$ et poser $f_u(w) \leftarrow k + 1$ et $p_u(w) \leftarrow v$.

Lorsque les arcs ont des **longueurs quelconques**, on peut utiliser l'**algorithme de Bellman-Ford** (1956) dont la complexité est $O(|A||V|)$ et qui peut être décrit comme suit.

Poser $f_u(u) \leftarrow 0$, $f_u(v) \leftarrow \infty$ pour tout $v \neq u$, *changement* $\leftarrow$ *vrai* et *compteur* $\leftarrow 0$ ;
Tant que *changement* $=$ *vrai* faire
    *changement* $\leftarrow$ *faux* et *compteur* $\leftarrow$ *compteur* $+ 1$;
    Pour tout sommet $v$ dans $G$ faire
        Pour tout sommet $w$ tel que $(v, w) \in A$ faire
            Si $f_u(w) > f_u(v) + \ell(v, w)$ alors
                $f_u(w) \leftarrow f_u(v) + \ell(v, w)$, $p_u(w) \leftarrow v$ et *changement* $\leftarrow$ *vrai* ;
    Si *changement* $=$ *vrai* et *compteur* $= |V|$ alors STOP : il existe un circuit de longueur $< 0$.

Lorsqu'un circuit de longueur négative est détecté, on peut le retracer à l'aide des valeurs $p_u(v)$. En effet, ce circuit apparaîtra dans le sous-graphe partiel de $G$ ne contenant que les arcs $(p_u(v), v)$.
Le temps de calcul de cet algorithme dépend fortement de l'ordre dans lequel les sommets sont considérés. Si $v$ apparaît avant $w$ lorsque $f_u(v) < f_u(w)$, l'algorithme déterminera les plus courts chemins dès le premier passage dans le 'tant que', alors que plusieurs passages sont nécessaires autrement. Mais un tel ordre n'est bien sûr pas connu, car on n'aurait alors pas besoin de déterminer des plus courts chemins.

Lorsque le graphe $\boldsymbol{G}$ **n'a pas de circuit**, alors que les longueurs peuvent être quelconques, on peut utiliser l'algorithme suivant, dont la complexité est $O(|A|)$. Lorsqu'on recherche les plus courts chemins de $u$ vers les autres sommets du graphe, en supposant que $u$ est une racine, on peut ôter les arcs entrant en $u$.

Poser $f_u(u) \leftarrow 0$ et $S \leftarrow \{u\}$ ;
Pour tout $v \neq u$ faire $d(v) \leftarrow$ nombre d'arcs $(w, v) \in A$ tels que $w \neq u$ ;.
Tant que $|S| < |V|$ faire
    Déterminer l'ensemble $W$ des sommets $v \notin S$ tels que $d(v) = 0$ et poser $S \leftarrow S \cup W$ ;
    Pour tout $v \in W$ faire
        $f_u(v) \leftarrow \min\{f_u(w) + \ell(w, v) \mid (w, v) \in A\}$ ;
        Pour tout $w$ tel que $(v, w) \in A$ faire $d(w) \leftarrow d(w) - 1$ ;

Cet algorithme crée une partition des sommets en couches successives $V_1, \ldots, V_k$ telle que tous les prédécesseurs d'un sommet de $V_i$ sont dans des ensembles $V_j$ avec $j < i$ : à chaque nouveau passage dans le 'tant que', on a un nouvel ensemble $W$ qui est la couche suivante, la première couche étant constituée du seul sommet $u$.

Nous allons maintenant nous intéresser au problème consistant à déterminer les longueurs des **plus courts chemins entre toutes les paires de sommets** de $G$, avec des longueurs quelconques. L'algorithme ci-dessous, proposé par **Floyd-Warshall** en 1962, suppose que les sommets sont numérotés de 1 à $|V|$ et utilise un compteur $k$ tel que à la fin du $k$-ième passage dans la boucle, $L_{ij}$ correspond à la longueur d'un plus court chemin de $i$ vers $j$, avec des sommets intermédiaires faisant partie de $\{1, \ldots, k\}$. Initialement, lorsque $k = 0$, on a donc $L_{ii} = 0$ pour tout sommet $i$, $L_{ij} = \ell(i, j)$ si $(i, j) \in A$, et $L_{ij} = \infty$ autrement.

Pour tout $i = 1, \ldots, |V|$ faire
    Pour tout $j = 1, \ldots, |V|$ faire
        $$L_{ij} \leftarrow \begin{cases} 0 & \text{si } i = j \\ \ell(i, j) & \text{si } (i, j) \in A \\ \infty & \text{sinon.} \end{cases}$$
Pour tout $k = 1, \ldots, |V|$ faire
    Pour tout $i = 1, \ldots, |V|$ faire
        Pour tout $j = 1, \ldots, |V|$ faire
            $L_{ij} \leftarrow \min\{L_{ij}, L_{ik} + L_{kj}\}$ ;
        Si $L_{ii} < 0$ on peut stopper l'algorithme : il existe un circuit de longueur $< 0$.

Remarquons que tous les algorthmes présentés dans ce document sont polynomiaux. Ils permettent de déterminer des plus courts chemins ou des plus longs chemins (si on multiplie par -1 chaque longueur). Cependant, ils supposent tous que ces chemins n'ont pas besoin d'être élémentaires. En d'autres termes, il est possible de passer plusieurs fois par un même sommet. C'est la raison pour laquelle ces algorithmes sont stoppés lorsqu'on détecte un circuit de longueur négative : en effet, on pourrait alors circuler à l'infini dans de tels circuits, ce qui signifie que des plus courts chemins auraient une longueur égale à $-\infty$.

La situation est très différente lorsqu'on exige que les chemins déterminés soient élémentaires. La détermination des plus courts chemins entre un sommet $u$ et les autres devient alors un problème NP-dur. Si on savait le résoudre, on pourrait déterminer très facilement si un graphe $G$ contient un chemin hamiltonien, alors que ce probème est connu comme étant NP-complet : il suffirait d'attribuer des longueurs 1 à chaque arc de $G$, de déterminer ensuite les plus longs chemins entre toute paire de sommets, et on pourrait conclure que $G$ a un chemin hamiltonien si et seulement il existe un sommet à distance $|V| - 1$ d'un autre sommet.

Un réseau routier peut être considéré comme un graphe avec des coûts positifs sur chaque arc. Les sommets représentent des croisements de routes et chaque arc du graphe est associé à un segment de route entre deux croisements. Le coût d'un arc peut correspondre à la longueur du segment de route associé, au temps nécessaire pour traverser ce segment ou à tout autre coût lié au parcours de ce segment de route. En utilisant des graphes orientés, il est également possible de modéliser des rues en sens unique. Les algorithmes que nous avons présentés peuvent donc naturellement être implantés dans des systèmes GPS. Il existe en fait de nombreuses autres applications très utiles en pratique. En voici deux.

Considérons un **problème d'ordonnancement de tâches, avec contraintes de précédences**. Plus précisément, considérons un ensemble $\mathcal{T} = \{T_1, \ldots, T_n\}$ de $n$ tâches avec des temps d'exécution $p_1, \ldots, p_n$. Aucune tâche ne peut être interrompue, et on suppose qu'elles sont soumises à des contraintes de type $t_j - t_i \geq a_{ij}$, où $t_i$ dénote l'heure à laquelle la tâche $T_i$ débute. Ainsi, si $a_{ij} > 0$ cela signifie qu'au démarrage de $T_i$, il faut attendre au minimum $a_{ij}$ unités de temps avant de débuter $T_j$. Aussi, si $a_{ij} < 0$, cela signifie que $T_i$ doit démarrer au plus tard $-a_{ij}$ unités de temps après le début de $T_j$. On cherche alors, par exemple, à déterminer un horaire de durée totale minimale. On peut aussi être intéressé à déterminer l'heure de démarrage au plus tôt de chaque tâche, ou l'heure de démarrage au plus tard de chaque tâche, sans que cela ne retarde la fin de l'exécution des $n$ tâches.

Pour ce faire, on construit un *graphe conjonctif* $G$ dont les sommets sont $\mathcal{T} \cup \{\textit{début}, \textit{fin}\}$. Pour chaque contrainte $t_j - t_i \geq a_{ij}$, on crée un arc $(T_i, T_j)$ de longueur $a_{ij}$. Pour chaque tâche $T_i$, on rajoute un arc $(\textit{début}, T_i)$ de longueur 0 et un arc $(T_i, \textit{fin})$ de longueur $p_i$. Notons $L(i, j)$ la longueur d'un plus long chemin de $i$ vers $j$ dans ce graphe. La longueur $L(\textit{début}, \textit{fin})$ est alors la durée minimale de l'exécution des $n$ tâches. Aussi, l'heure de démarrage au plus tôt de $T_i$ est $L(\textit{début}, T_i)$, alors que l'heure de démarrage au plus tard de $T_i$, sans que cela n'augmente le temps nécessaire à l'exécution des $n$ tâches est $L(\textit{début}, \textit{fin}) - L(T_i, \textit{fin})$.

Considérons maintenant un **problème de gestion de stock** sur un horizon fini de $n$ périodes, dans lequel les commandes sont passées à intervalles réguliers et les demandes sont uniformément réparties durant chaque période. Plus précisément, notons $s_k$ le niveau du stock au début de la période $k$ (on suppose que $s_1$ est connu). Soit $x_k$ la quantité (supposée entière) à commander au début de la période $k$ et $c_k(x)$ le coût d'une commande de $x$ unités au début de la période $k$. On suppose que la demande entre le début de la période $k$ et la suivante est $d_k$. Finalement, notons $h_k(s)$ le coût de stockage à payer durant la période $k$ pour un stock moyen de niveau $s$, en supposons que la demande est uniforme. En d'autres termes, le stock moyen durant la période $k$ est $s_k + x_k - \frac{1}{2}d_k$. La capacité de stockage est limitée à $C$ unités en tout temps. En supposant que le stock final $s_{n+1}$ est imposé, on désire déterminer la politique de commandes qui minmisera le coût total.

On peut formuler ce problème en termes de plus plus court chemin dans un graphe. À cet effet, créons un sommet $(k, s)$ pour chaque début de période $k$ $(1 \leq k \leq n+1)$ et pour chaque niveau de stock $s$ admissible au début de la période $k$. Pour chaque sommet $(k, s)$ et pour chaque commande $x$ admissible au début de la période $k$ (sachant que le stock actuel est de niveau $s$), créons un arc de $(k, s)$ vers $(k+1, s+x-d_k)$ de coût $c_k(x) + h_k(s + x - \frac{1}{2}d_k)$. Une politique optimale de stockage est alors obtenue en déterminant un plus court chemin de $(1, s_1)$ vers $(n+1, s_{n+1})$.

Pour terminer, mentionnons que l'arborescence des plus courts chemins reliant un sommet $r$ aux autres sommets du graphe n'est pas nécessairement une arborescence de racine $r$ et de coût minimum, tel que le montre l'exemple ci-dessous.