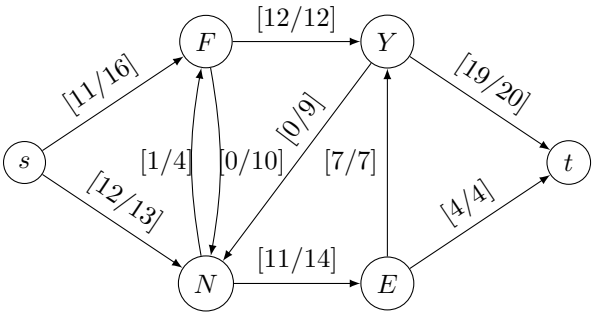
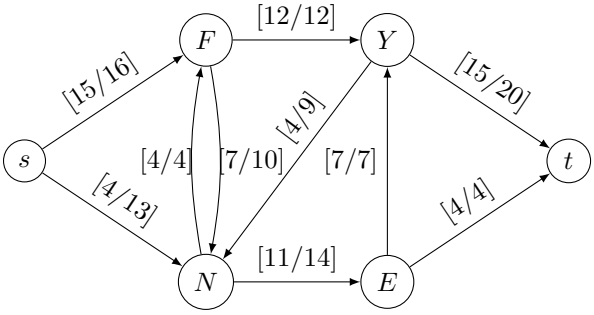
Le problème du flot maximum

Recherche opérationnelle
Dimitri Watel - ENSIIE

2024

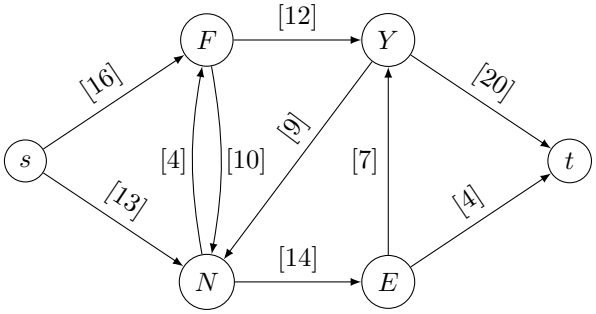
Le problème du flot maximum est un problème classique en recherche opérationnelle dont l'objectif est de modéliser une problématique de déplacement optimal dans un graphe. Les applications sont multiples, les plus classiques étant un réseau de fluide (gaz, liquide) d'où le problème sort son nom, un réseau de transport, un réseau d'énergie ou de télécommunication. Nous allons décrire le problème, expliquer comment on peut le résoudre et démontrer l'exactitude de l'algorithme proposé.





1 Définition du problème

Dans le graphe G suivant où on a identifié une source s et un puits t, chaque arc a est associé à un entier $c(a)$. Ces entiers sont les *capacités* des arcs. On parle de *réseau de flot* ou *réseau de transport* (G, s, t, c).



On cherche à faire passer de l'eau de s à t en suivant les arcs du graphe, sachant que

- la quantité d'eau qui entre dans un nœud doit être identique à celle qui sort du nœud, sauf en s et t,
- la quantité d'eau qui passe par un arc ne peut excéder la capacité de cet arc.

Ainsi dans l'exemple ci-dessus, on pourrait avoir les solutions suivantes.

Plus formellement, un flot f est une fonction de A vers $\mathbb{N}$. Il est dit *admissible* s'il respecte les deux contraintes suivantes:

- pour tout arc $a \in A$, $f(a) \in [0, c(a)]$ (contrainte de capacité)
- pour tout nœud $v \in V$ **exceptés** s **et** t, $\sum_{a \in \gamma^-(v)} f(a) = \sum_{a \in \gamma^+(v)} f(a)$ (contrainte de conservation)

Parmi les deux solutions de l'exemple, la première est plus intéressante, on constate que le flot sortant de s est plus grand et donc, on a pu faire passer plus de flot de s vers t. On appelle *valeur* la quantité de flot sortant de s (ou de manière équivalente la quantité entrant en t). Attention! Rien n'empêche le flot de boucler et de revenir en s, dans ce cas, ce flot va ressortir une deuxième fois de s, mais il ne faut pas le compter deux fois dans la valeur. Ainsi, on définit la valeur v d'un flot f avec

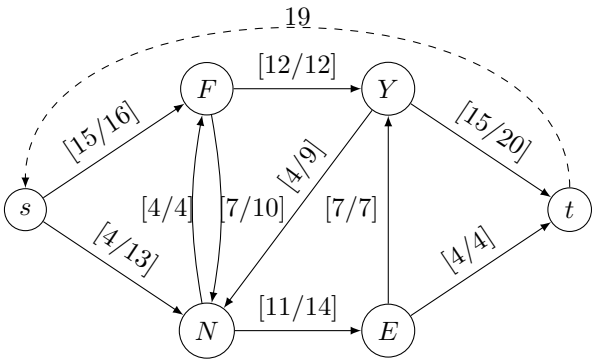
$$v = \sum_{a \in \gamma^-(t)} f(a) - \sum_{a \in \gamma^+(t)} f(a)$$
$$v = \sum_{a \in \gamma^+(s)} f(a) - \sum_{a \in \gamma^-(s)} f(a)$$

Le problème du flot maximum se décrit ainsi:

Problème 1. Connaissant un réseau de flot (G, s, t, c) trouver un flot admissible f de valeur v maximum dans ce réseau.

Un flot maximisant v est dit optimal ou maximum.

Avant de passer à la résolution du problème, on peut simplifier un détail. On remarque que la contrainte de conservation ne concerne pas s et t, mais que le flot sortant de s est égal au flot entrant en t. En rajoutant un arc fictif entre s et t dont le flot est égal à v, on obtient un flot admissible qui respecte la contrainte de conservation sur tous les nœuds.

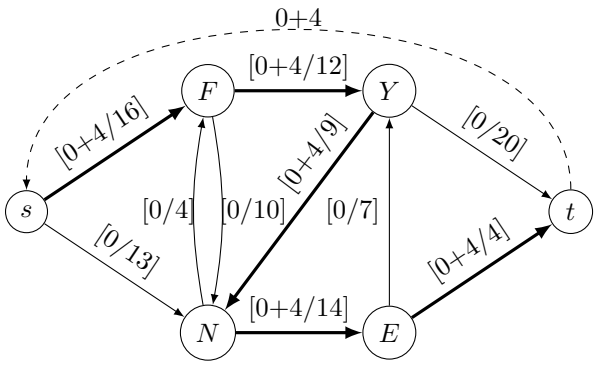


2 L'algorithme de Ford Fulkerson

2.1 Principe de l'algorithme

L'algorithme de Fold Fulkerson injecte du flot dans le réseau petit à petit jusqu'à trouver un flot maximum. L'algorithme démarre avec un flot déjà admissible. La solution la plus simple consiste à partir du flot nul.

L'algorithme recherche *une chaîne augmentante*. Il s'agit d'une chaîne de G reliant s et t dans laquelle on peut injecter du flot. Un exemple simple de chaîne augmentante est un chemin reliant s à t dans lequel chaque arc est non saturé (au sens où le flot associé n'a pas atteint sa capacité). Dans l'exemple suivant, on a un chemin augmentant qui augmente la valeur de 4.

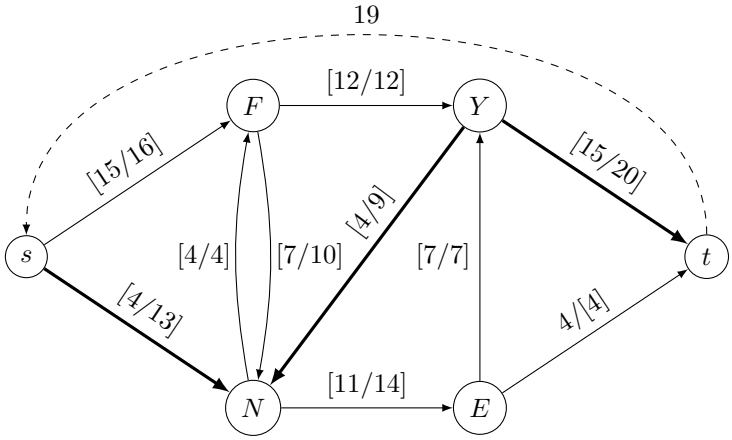


Cependant, cette version simple des chaînes augmentantes a ses limites. Dans l'exemple de valeur 19 en fin de première section, il n'existe aucun chemin de la sorte. Pourtant il existe une solution de valeur 23 (comme indiqué dans un des exemples précédents). L'algorithme de Ford-Fulkerson arrive a outrepasser cette difficulté et à trouver le flot maximum en utilisant une version plus généralisée des chaînes augmentante.

Une *chaîne augmentante* μ dans le réseau (G, s, t, c) muni d'un flot admissible f est une **chaîne** (non orientée) reliant s à t telle que:

- pour tout arc a de μ orienté de s vers t, $f(a) < c(a)$. On note μ^+ ces arcs.
- pour tout arc a de μ orienté de t vers s, $f(a) > 0$. On note μ^- ces arcs.

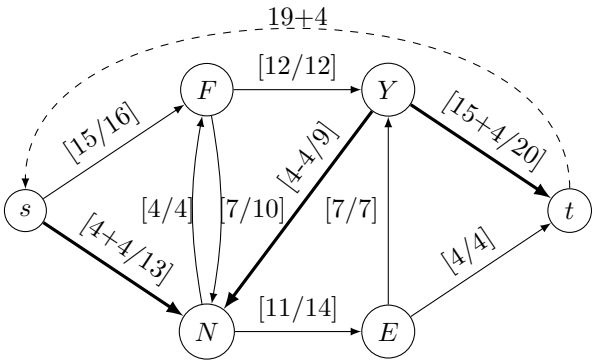
Dans l'exemple suivant, on a représenté la chaîne $sNYt$ (on peut remarquer que la chaîne ne tient pas compte de l'orientation). Les arcs sN et Yt sont dans le sens de s vers t, ces arcs ne sont pas saturés. Et l'arc YN est dans le sens de t vers s, cet arc n'est pas vide. On a donc bien une chaîne augmentante.



Une fois qu'on dispose d'une chaîne augmentante, on peut augmenter la valeur du flot en augmentant $f(a)$ si $a \in \mu^+$ et en diminuant $f(a)$ si $a \in \mu^-$. Le delta de variation doit être le même sur tous les arcs ce qui garanti qu'on respecte bien la contrainte de conservation. Et il faut respecter la contrainte de capacité, donc ne pas avoir

d'arc dépassant la capacité ou d'arc qui se retrouve avec un flot négatif.

$$v \to v + \min_{a \in \mu} \begin{cases} c(a) - f(a) & \text{if } a \in \mu^+ \\ f(a) & \text{if } a \in \mu^- \end{cases}$$



L'algorithme de Ford Fulkerson est le suivant:

Algorithme 1 Algorithme de Ford Fulkerson

Entrées: (G, s, t, c) a flow network
$f \leftarrow$ un flot nul sur tous les arcs.
Tant que il existe une chaîne augmentante μ dans G
Fairethere exists an augmenting path μ on f
$dv \leftarrow \min\left(\min_{a \in \mu^+}\{c(a) - f(a)\}; \min_{a \in \mu^-}\{f(a)\}\right)$
$\forall a \in \mu^+, f(a) \leftarrow f(a) + dv$
$\forall a \in \mu^-, f(a) \leftarrow f(a) - dv$

2.2 Trouver une chaîne augmentante

Il existe deux méthodes équivalentes: le réseau résiduel et l'algorithme de marquage.

Soit $(G = (V, A), s, t, c)$ un réseau de transport et un flot f réalisable. On définit le *réseau résiduel* comme un graphe $H = (V, B)$ et un poids $\omega : B \to \mathbb{N}$ sur les arcs de H tels que:

- pour tout arc $a = (u, v) \in A$ tel que $f(a) < c(a)$, on ajoute un arc $b = (u, v) \in B$
- pour tout arc $a = (u, v) \in A$ tel que $f(a) > 0$, on ajoute un arc $b = (v, u) \in B$

Lemme 2.1. *Il existe une chaîne augmentante dans G si et seulement s'il existe un chemin de s vers t dans H.*

Proof. Soit une chaîne P reliant s et t dans G. On note w_i le i-ieme nœuds de P, avec $s = w_0$.

P est une chaîne augmentante
$\Leftrightarrow$ pour tout $i \in [\![0, |P| - 1]\!]$
$\begin{cases} (w_i, w_{i+1}) \text{ est un arc de } G \text{ non saturé, ou} \\ (w_{i+1}, w_i) \text{ est un arc de } G \text{ non vide} \end{cases}$
$\Leftrightarrow$ pour tout $i \in [\![0, |P| - 1]\!]$ (w_i, w_{i+1}) est un arc de H
$\Leftrightarrow$ P est un chemin de H

□

L'algorithme de marquage est le suivant

Algorithme 2 Algorithme de marquage

Marquer la source avec $+$
Pour $(u, v) \in A$ **Faire**
Si u est marqué, v est non marqué et $f(u, v) < c(u, v)$ **Alors**
marquer v avec $+(u)$
Sinon Si v est marqué, u est non marqué et $f(u, v) > 0$, **Alors**
marquer u avec $-(v)$
Recommencer la ligne 2 si un nœud a été marqué

Lemme 2.2. *t est marqué si et seulement s'il existe une chaîne augmentante dans G*

Proof. A l'exception de s, chaque nœud marqué l'est avec un autre nœud de G qui a lui-même été marqué à une itération précédente. Ainsi, en *remontant* le marquage, on arrive toujours en s. Supposons que t soit marqué et notons $P = (s = w_0, w_1, w_2, \ldots, t = w_{|P|})$ la suite de nœud marqués qui a mené au marquage de t.

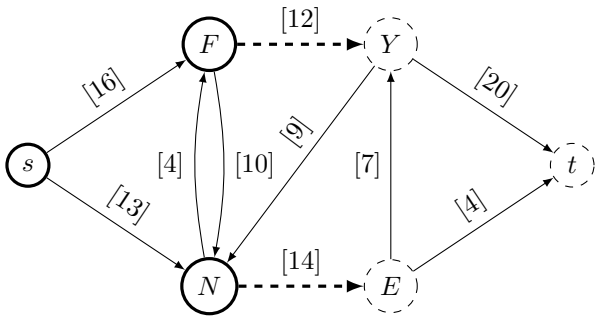
Pour tout $i \in [\![0, |P| - 1]\!]$, puisque w_{i+1} a été marqué avec $+w_i$ ou $-w_i$
$\begin{cases} (w_i, w_{i+1}) \text{ est un arc de } G \text{ non saturé, ou} \\ (w_{i+1}, w_i) \text{ est un arc de } G \text{ non vide} \end{cases}$
Et donc P est une chaîne augmentante.

Si $P = (s = w_0, w_1, w_2, \ldots, t = w_{|P|})$ est une chaîne augmentante. On montre par récurrence que chaque nœud de P est marqué. w_0 est marqué à la première ligne de l'algorithme. Supposons maintenant que w_i est marqué par l'algorithme lors d'une itération. On s'intéresse à l'éxécution de la boucle ligne 2 qui fait suite à ce marquage. Si, lors de cette exécution, w_{i+1} est déjà marqué (par u autre nœud que w_i) alors on a prouvé la propriété de récurrence souhaitée. Sinon, puisque P est une chaîne augmentante, alors
$\begin{cases} (w_i, w_{i+1}) \text{ est un arc de } G \text{ non saturé, ou} \\ (w_{i+1}, w_i) \text{ est un arc de } G \text{ non vide} \end{cases}$
Selon le cas, il est possible de marquer w_{i+1} avec $+w_i$ ou $-w_i$. Et donc w_{i+1} sera nécessairement marqué au plus tard lors de l'itération de la boucle ligne 2 où u et v seront les nœuds w_i et w_{i+1}. Par récurrence, tous les nœuds de P sont marqués, en particulier t. □

La dernière section prouve que l'algorithme de Ford Fulkerson est correct en montrant que, s'il n'y a plus de chaîne augmentante alors le flot est maximum

3 Le problème de la coupe minimum et théorème de dualité forte

On va s'intéresser à un problème connexe, dit dual du problème du flot maximuù, pour démontrer l'exactitude de l'algorithme. On s'intéresse à des coupes du graphe séparant s et t. Une telle coupe est une partition des nœuds en deux ensembles (S, T) tels que $s \in S$ et $t \in T$. On a représenté ci-après un exemple de coupe où les nœuds en gras correspondent à S et les autres à T. Les deux ensembles n'ont pas besoin d'être de même taille, et peuvent ne contenir que s ou que t.



Connaissant une coupe, on s'intéresse aux arcs allant de S à T (et uniquement ceux là). On calcule ensuite la somme des capacités de ces arcs. Dans l'exemple précédent, ces arcs sont représentés en pointillés. On remarque par exemple que (Y, N) n'est pas compté car cet arc part de T et va vers S. La capacité de cette coupe est donc 26.)

Remarque 1. Même si l'arc fictif est ajouté au réseau, il n'est jamais compté dans la capacité de la coupe car il relie nécessairement T vers S et non l'inverse.

Le problème de la coupe minimum se décrit ainsi:

Problème 2. Connaissant un réseau de flot (G, s, t, c) trouver une coupe (S, T) de capacité minimum dans ce réseau.

Une telle coupe est dite minimum.

Bien que différent du problème de flot maximum, on va démontrer que les deux problèmes sont liés: on va montrer la valeur des solutions optimales dans un même réseau de flot est la même dans les deux problèmes. Si on connait la valeur d'un flot maximum, alors on connait la capacité d'une coupe minimum et inversement. Pour démontrer ce théorème, on va dans un premier temps démontrer un résultat intermédiaire.

Lemme 3.1. *Considérons un réseau de flot* $(G = (V, A), s, t, c)$ *muni de l'arc fictif* (t, s) *et* $W \subset V$ *alors* $\sum_{a \in \gamma^-(W)} f(a) = \sum_{a \in \gamma^+(W)} f(a)$.

Remarque 2. Autrement dit la contrainte de conservation se vérifie sur un sous-ensemble de nœuds.

Proof. Puisque l'arc fictif est présent, la contrainte de conservation est vérifiée en chaque nœud de W.

$$\sum_{a \in \gamma^-(v)} f(a) = \sum_{a \in \gamma^+(v)} f(a) \ \forall v \in W$$
$$\sum_{v \in W} \sum_{a \in \gamma^-(v)} f(a) = \sum_{v \in W} \sum_{a \in \gamma^+(v)} f(a)$$

Considérons un arc $a = (v_1, v_2) \in A$. Si v_1 et v_2 sont dans W alors cet arc apparait dans la somme de gauche et dans la somme de droite. Si on simplifie, il ne reste, à gauche que les arcs (u, v) tels que $u \notin W$, et à droite que les arcs (v, u) tels que $u \notin W$.

$$\sum_{v \in W} \sum_{\substack{(u,v) \in \gamma^-(v) \\ u \notin W}} f(u, v) = \sum_{v \in W} \sum_{\substack{(v,u) \in \gamma^+(v) \\ u \notin W}} f(v, u)$$
$$\sum_{a \in \gamma^-(W)} f(a) = \sum_{a \in \gamma^+(W)} f(a)$$

□

On peut maintenant démontrer une version faible du théorème qu'on souhaite prouver.

Théorème 3.1. *Connaissant un réseau de flot* $(G = (V, A), s, t, c)$*, un flot admissible* f *de valeur* v *et une coupe* (S, T) *de capacité* $c(S, T)$*, alors* $v \leq c(S, T)$.

Proof. D'après le lemme 3.1,

$$\sum_{a \in \gamma^-(S)} f(a) = \sum_{a \in \gamma^+(S)} f(a)$$

Un des arcs entrant en S est l'arc fictif (t, s), de flot v. Puisque tous les autres arcs de $\gamma^-(S)$ sont de flot positif, on a

$$v \leq \sum_{a \in \gamma^+(S)} f(a)$$

Puisque tous les arcs ont un flot inférieur à leur capacité, on a

$$v \leq \sum_{a \in \gamma^+(S)} c(a)$$

Et, enfin, puisque tout arc sortant de S va dans T, cette dernière somme est la capacité de la coupe.

$$v \leq c(S, T)$$

□

Si, connaissant un flot, on trouve une coupe dont la capacité est égale à la valeur du flot, alors ce flot est nécessairement maximum (et la coupe minimum). On va montrer que c'est toujours possible et que, à partir du flot résultant de l'algorithme de Ford Fulkerson, on peut construire assez simplement cette coupe.

Théorème 3.2. *Connaissant un réseau de flot $(G = (V, A), s, t, c)$, soit v la valeur d'un flot maximum et C la capacité d'une coupe minimum alors $v = C$.*

Théorème 3.3. *L'algorithme de Ford Fulkerson renvoie un flot maximum.*

Proof. La preuve des deux théorèmes se fait conjointement. Considérons le flot f de valeur v renvoyé à l'issue de l'algorithme. Par construction, le réseau de flot, muni de f, ne dispose plus de chaîne augmentante. D'après le lemme 2.2, si on applique l'algorithme de marquage, le nœud t n'est pas marqué. Le nœud s étant toujours marqué, on obtient une coupe (S, T) en mettant dans S tous les nœuds marqués et dans T les autres.

D'après le lemme 3.1,

$$\sum_{a \in \gamma^-(S)} f(a) = \sum_{a \in \gamma^+(S)} f(a)$$

Un des arcs entrant en S est l'arc fictif (t, s), de flot v. Tous les autres arcs de $\gamma^-(S)$ sont des arcs (u, v) où $u \in T$ et $v \in S$. Puisque v est marqué et u ne l'est pas, alors $f(u, v) = 0$ (sinon u aurait été marqué avec " $-v$").

$$v = \sum_{a \in \gamma^+(S)} f(a)$$

Tous les arcs de $\gamma^+(S)$ sont des arcs (u, v) où $u \in S$ et $v \in T$. Puisque u est marqué et v ne l'est pas, alors $f(u, v) = c(u, v)$ (sinon v aurait été marqué avec " $+u$").

$$v = \sum_{a \in \gamma^+(S)} c(a)$$

Et, enfin, puisque tout arc sortant de S va dans T, cette dernière somme est la capacité de la coupe.

$$v = c(S, T)$$

D'après le théorème 3.1, tout flot a une valeur inférieur à $c(S, T)$, donc à v. De même toute coupe a une capacité plus grande que v dont que $c(S, T)$. On en déduit donc que le flot f renvoyé par l'algorithme de Ford Fulkerson est un flot maximum et que (S, T) est une coupe minimum. □

Pour conclure cette partie, appuyons donc juste sur la méthode décrite dans la preuve du théorème 3.3 pour trouver une coupe minimum:

Algorithme 3 Trouver une coupe minimum

Appliquer l'algorithme de Ford-Fulkerson
Version 1
Après la dernière itération utiliser l'algorithme de marquage.
$S \leftarrow$ les nœuds marqués
$T \leftarrow$ les nœuds non marqués
Renvoyer (S, T)
Version 2
Après la dernière itération construire le réseau résiduel H.
$S \leftarrow$ les nœuds accessible depuis s dans H
$T \leftarrow$ les autres nœuds
Renvoyer (S, T)