

# Chapitre 7

## Problèmes de flots.

### 7.1 Exemple.

Un réseau électrique est formé de lignes reliant des noeuds (transformateurs, centre de redistributions,...), chaque ligne a une capacité de transport maximale. On veut faire passer une quantité de courant maximale entre le lieu de production (la source, supposée unique) et le lieu de consommation (la cible, aussi supposée unique).

### 7.2 Notions de base sur les graphes

- Un graphe orienté  $G=(V,E)$  est formé d'un ensemble de sommets  $V = \{x_1, \dots, x_n\}$  et d'un ensemble d'arcs  $E = \{1, 2, \dots\}$ . L'arc  $k$  reliant  $x_i$  à  $x_j$  est aussi noté  $(x_i, x_j)$ .
- un **chemin** est une suite  $x_1, x_2, \dots, x_p$  tel que  $(x_i, x_{i+1}) \in E$  pour  $i = 1, \dots, p - 1$ .
- une **chaîne** est une suite  $x_1, x_2, \dots, x_p$  tel que  $(x_i, x_{i+1}) \in E$  (arc avant) ou  $(x_{i+1}, x_i) \in E$  (arc arrière) pour  $i = 1, \dots, p - 1$ .
- **Successeurs** de  $x \in V$  c'est l'ensemble  $Suc(x) = \{y \mid (x, y) \in E\}$ .
- **Prédécesseurs** de  $x \in V$  c'est l'ensemble  $Pred(x) = \{y \mid (y, x) \in E\}$ .
- A tout arc  $k$  (ou  $(x, y)$ ) on associe sa **capacité**  $c_k$  (ou  $c(x, y)$ ) un nombre  $> 0$ .

On considèrera uniquement des graphes tels que :

1. il existe un sommet source  $s$  (c.a.d. que  $Pred(s) = \emptyset$ ).
2. il existe un sommet cible  $t$  (c.a.d. que  $Suc(t) = \emptyset$ ).
3. il existe un chemin de  $s$  à  $t$  (sinon le problème de flot n'a pas de solution).

## 7.3 Flots et coupe

### 7.3.1 Flot

Un flot  $f$  de valeur  $v$  est une fonction qui à chaque arc  $(x, y)$  associe une valeur  $f(x, y)$  telle que :

$$\begin{aligned} \forall x \in V, x \neq s, t \quad & \sum_{z \in \text{Pred}(x)} f(z, x) = \sum_{y \in \text{Suc}(x)} f(x, y) \\ & v = \sum_{y \in \text{Suc}(s)} f(s, y) \\ & v = \sum_{z \in \text{Pred}(t)} f(z, t) \end{aligned}$$

Un flot est **admissible** ssi  $\forall x, y \ f(x, y) \leq c(x, y)$ . L'arc  $(x, y)$  est saturé si  $f(x, y) = c(x, y)$ . Le problème posé est de trouver un flot admissible de valeur maximale.

### 7.3.2 Flot et programmation linéaire

Le problème de flot maximal est équivalent au problème de P.L. suivant :

$$\begin{aligned} \text{Max} \quad & v \\ & 0 \leq f(x, y) \leq c(x, y) \quad \forall (x, y) \in E \\ & \sum_{y \in \text{Pred}(x)} f(y, x) = \sum_{z \in \text{Suc}(x)} f(x, z) \quad \forall x \neq s, t \\ & v = \sum_{x \in \text{Suc}(s)} f(s, x) \\ & v = \sum_{y \in \text{Pred}(t)} f(y, t) \end{aligned}$$

### 7.3.3 Coupe minimale et flot maximal

Une coupe est donnée par un ensemble de sommet  $W$  tel que  $\begin{cases} s \in W \\ t \notin W \end{cases}$

La coupe  $C$  est définie par  $C = \{(x, y) \mid x \in W, y \in \overline{W}\}$

**Proposition 1** *Soit  $C$  une coupe, alors tout chemin de  $s$  à  $t$  contient un arc de  $C$ .*

La capacité de la coupe  $C$  associée à  $W$  est  $c = \sum_{(x,y) \in E, x \in W, y \in \overline{W}} c(x, y)$ .

**Théorème de la coupe minimale.** On note  $f(H, K) = \sum_{x \in H, y \in K} f(x, y)$ .

**Proposition 2** *Pour tout flot  $f$  de valeur  $v$ , pour toute coupe  $C$  (donnée par  $W$ ),  $v = f(W, \overline{W}) - f(\overline{W}, W)$*

La preuve se fait par récurrence sur  $|W|$ .

1.  $|W| = 1$  (donc  $W = \{s\}$ ). Dans ce cas  $f(W, \overline{W}) = \sum_{x \in \text{Suc}(s)} f(s, x)$  et  $f(\overline{W}, W) = 0$  (pas d'arc entrant sur  $s$ ) ce qui donne le résultat.

2. On suppose le résultat vrai pour une coupe  $W$  et on considère  $W' = W \cup \{x\}$  (on note  $+$  et  $-$  l'union et la soustraction ensembliste).

$$f(W', \overline{W'}) - f(\overline{W'}, W') = f(W + x, \overline{W} - x) - f(\overline{W} - x, W + x)$$

d'où

$$f(W', \overline{W'}) - f(\overline{W'}, W') = f(W, \overline{W}) + f(x, \overline{W}) - f(W, x) - f(\overline{W}, W) + f(x, W) - f(\overline{W}, x)$$

d'où

$$f(W', \overline{W'}) - f(\overline{W'}, W') = f(W, \overline{W}) - f(\overline{W}, W) + f(x, \overline{W}) + f(x, W) - f(W, x) - f(\overline{W}, x)$$

Mais  $f(x, V) = f(x, \overline{W}) + f(x, W)$  et  $f(V, x) = f(W, x) + f(\overline{W}, x)$  d'où le résultat car  $f(V, x) = f(x, V)$  et par hypothèse d'induction  $v = f(W, \overline{W}) - f(\overline{W}, W)$ .

On a  $c = f(W, \overline{W}) \geq f(W, \overline{W}) - f(\overline{W}, W) = v$ . Par conséquent la capacité d'une coupe est toujours supérieure ou égale à la valeur de tout flot (notamment à celle d'un flot maximal). Une égalité induit que le flot est maximal et que la capacité de la coupe est minimale. L'algorithme de Ford-Fulkerson nous donnera un flot maximal qui correspond à une coupe minimale, ce qu'on résumera en : Une *coupe minimale* correspond à un *flot maximal*

## 7.4 Algorithme de Ford-Fulkerson

### 7.4.1 Description de l'algorithme

**Chaine augmentant le flot.** Soit  $C$  une chaine  $s = x_1, x_2, \dots, x_n = t$ , alors on pose  $\epsilon_1 = \min_i \text{arc avant de } C(c_i - f_i)$  et  $\epsilon_2 = \min_i \text{arc arriere de } C(f_i)$  ( $f_i$  et  $c_i$  flots et capacité de  $i$ ). Si  $\epsilon = \min(\epsilon_1, \epsilon_2)$  est strictement positif, alors on peut augmenter le flot de  $\epsilon$  sur  $C$  : on ajoute  $+\epsilon$  sur un arc avant de  $C$  et  $-\epsilon$  sur un arc arriere de  $C$ .

**Algorithme de calcul d'un flot maximal.** Principe : partir d'un flot et chercher s'il existe une chaine sur laquelle on peut augmenter le flot. Si oui augmenter le flot et recommencer, sinon, alors le flot est optimal. Cet algorithme est du à Ford et Fulkerson.

1. Initialiser le flot à 0 (ou à un flot admissible).
2. Tant que c'est possible faire
  - chercher une chaine augmentant le flot de  $\epsilon > 0$
  - augmenter le flot de  $\epsilon$  sur cette chaine.
  - fait

Procédure de recherche d'une chaine augmentant le flot :

- (a) Initialisation.
  - $Marque = \{s\}$  et  $marque(s) = +\infty$
  - $Vu = \emptyset$

(b) Itération.

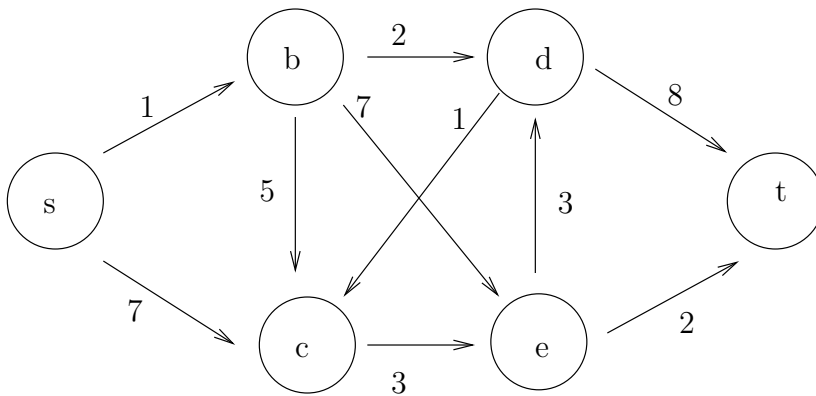
Tant que non fini faire

- i. Choisir  $x$  tel que  $\begin{cases} x \in Marque \\ x \notin Vu \end{cases}$
- ii.  $\forall y \in Suc(x)$  et  $y \notin Marque$   
si  $(x, y)$  non saturé alors  $\begin{cases} Marque = Marque \cup \{y\} \\ marque(y) = \text{Min}(marque(x), c(x, y) - f(x, y)) \end{cases}$
- iii.  $\forall y \in Pred(x)$  et  $y \notin Marque$   
si  $f(y, x) \neq 0$  alors  $\begin{cases} Marque = Marque \cup \{y\} \\ marque(y) = \text{Min}(marque(x), f(y, x)) \end{cases}$
- iv.  $Vu = Vu \cup \{x\}$
- v. fini :=  $t \in Marque$  (et alors  $\epsilon = marque(t)$ )  
ou  
 $t \notin Marque$  mais il n'existe plus de  $x \in Marque$  et  $x \notin Vu$   
(pas de chaine possible)

fait

Retrouver la chaine augmentant le flot se fait en remontant la chaine de marquage.

### 7.4.2 Exemple.



On initialise à 0 (on peut éventuellement trouver à la main un flot meilleur). Plusieurs stratégies sont possibles pour construire *marque* : en largeur, en file,... On choisit ici de construire *Marque* comme une pile (on développe le dernier arrivé), on écrit en même temps la valeur de  $marque(x)$ , les éléments qui sont dans *Vu* sont écrits en gras :

$$\text{Marque} = \{(s, +\infty)\}, Vu = \emptyset$$

$$\text{Marque} = \{(s, +\infty), (b, 1), (c, 7)\}, Vu = \{s\}$$

$$\text{Marque} = \{(s, +\infty), (b, 1), (c, 7), (e, 3)\}, Vu = \{s, c\}$$

$$\text{Marque} = \{(s, +\infty), (b, 1), (c, 7), (e, 3), (t, 2)\}, Vu = \{s, c, e\}$$

On a une chaîne  $s, c, e, t$  augmentant le flot de 1. La chaîne suivante est :  $s, c, e, d, t$  qui augmente de 1. Ensuite la nouvelle chaîne est  $s, b, d, t$  qui augmente de 1 et donne le flot maximal.

### 7.4.3 Correction et Terminaison

#### Correction.

**Proposition 3** *L'algorithme donne un flot maximal quand il termine.*

La preuve de correction se fait en deux étapes :

1. On montre qu'après chaque passage dans l'itération pour tout  $x \in \text{Marque}$ , il existe une chaîne entre  $s$  et  $x$  formée d'éléments de  $Vu$  sur laquelle on peut augmenter le flot de  $\text{marque}(x)$ .
2. On montre que si **fini** est vrai et que  $t \notin \text{Marque}$ , l'ensemble  $\text{Marque}$  des sommets marqués obtenu après la dernière itération est une coupe dont la capacité est celle du flot calculé d'où l'optimalité.

(a) C'est une coupe ( $s \in \text{Marque}$  et  $t \notin \text{Marque}$ ).

(b) Sa capacité est  $\sum_{(x \in \text{Marque}, y \notin \text{Marque})} c(x, y)$ . Montrons qu'elle vaut  $\sum_{(x \in \text{Marque}, y \notin \text{Marque})} f(x, y)$ .

On remarque que l'ensemble  $\text{Marque}$  est croissant au cours de l'algorithme.

Prenons  $x \in \text{Marque}, y \notin \text{Marque}$  tel que l'arc  $(x, y)$  existe. Le sommet  $x$  a été mis dans  $Vu$  à une itération de la boucle. Lors de cette itération, le sommet  $y$  n'a pas été ajouté à  $\text{Marque}$  (sinon il serait dans cette ensemble à la fin du calcul). Cela signifie que

- soit  $y$  est un successeur et l'arc  $(x, y)$  est saturé,
- soit  $y$  est un prédécesseur et le flot sur  $(y, x)$  est nul.

Comme  $y$  n'est pas dans l'ensemble  $\text{Marque}$  à la fin de l'algorithme, le flot n'est pas modifié sur aucun des arcs  $(z, y)$  ou  $(y, z)$ . Donc le flot vaut la capacité de la coupe, donc est maximal.

#### Terminaison.

**Proposition 4** *Si toutes les capacités sont des nombres rationnels, l'algorithme termine.*

On se ramène à des capacités entières en calculant le ppcm des dénominateurs et prenant comme unité de mesure  $1/\text{ppcm}$ . Dans le cas des capacités entières le flots est augmenté d'un nombre entier d'unités (calcul de  $\epsilon$ ) à chaque étape et donc on termine après un nombre fini d'étapes (borné par le min de (somme des capacités sortant de la source,

somme des capacités entrant sur la cible)). On en tire une borne sur le nombre d'itération de recherche d'une chaîne augmentant le flot pour des capacités entières : elle est de  $K$  si  $K$  est la capacité maximale.

Par contre l'exemple pathologique suivant, avec des capacités irrationnelles, montre que l'algorithme peut ne pas terminer et dans cet exemple il ne converge même pas vers le flot optimal! Toutes les implantations utilisant des nombres flottants (ou des entiers) ceci ne se produit pas en pratique.

- Sommets :  $V = \{s, (x_i)_{i=1,\dots,4}, (y_i)_{i=1,\dots,4}, t\}$
- Arcs spéciaux :  $A_1 = (x_1, y_1)$  de capacité  $a_0 = 1$  ,  $A_2 = (x_2, y_2)$  de capacité  $a_1 = (\sqrt{5} - 1)/2$ ,  $A_3 = (x_3, y_3)$  et  $A_4 = (x_4, y_4)$  de capacité  $a_2 = a_0 - a_1$ .
- Arcs non spéciaux :  $(s, x_i)$  et  $(y_i, t)$ ,  $(y_j, x_i)$ ,  $(x_i, y_j)$  et  $(y_i, y_j)$  pour  $i \neq j$  tous de capacité  $C \gg a_i$  (qu'on va calculer ensuite).

On montre alors que :

**Proposition 5** *A l'étape  $n$  de l'algorithme on peut trouver une chaîne qui augmente le flot de  $\epsilon_n$  qui vérifie la récurrence  $\epsilon_{n+2} = \epsilon_n - \epsilon_{n+1}$ ,  $\epsilon_0 = a_0$ ,  $\epsilon_1 = a_1 = (\sqrt{5} - 1)/2$ .*

Le flot augmentera indéfiniment en tendant vers la valeur  $C = \sum_{n \geq 0} a_n = 2/(3 - \sqrt{5})$  car on a  $\epsilon_n = (\sqrt{5} - 1)/2$ .

On peut facilement voir que le flot maximal est en fait  $4C$ !