Chapitre 5 : Flot maximal dans un graphe

J.-F. Scheid, B. Pinçon

v2.1

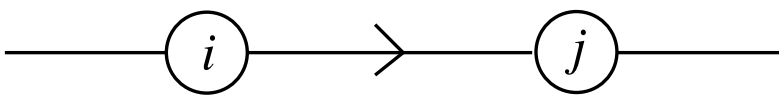
Plan du chapitre

1) Graphe

Graphe $G = (E, \Gamma)$

- E : ensemble fini des **sommets**
- Γ : ensemble fini de couples ordonnés (i, j) avec $i, j \in E$.
  Les éléments de Γ sont appelés les **arêtes** du graphe

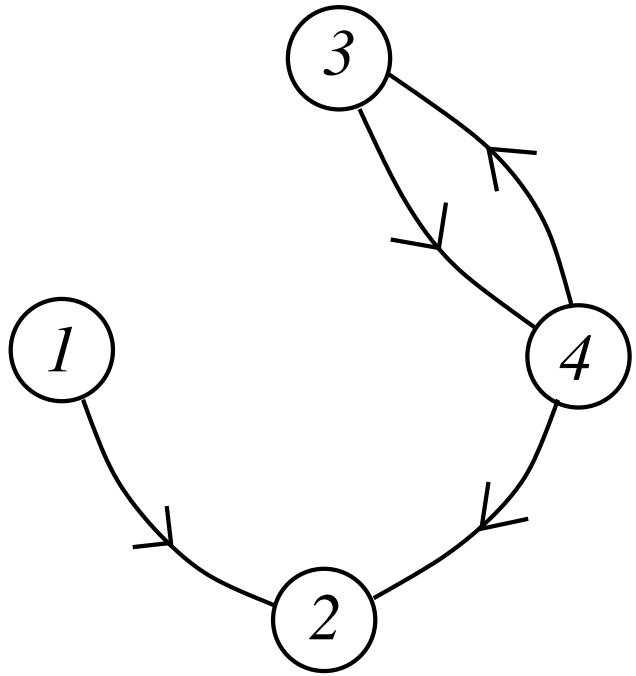
Notation :



Remarque : Les graphes considérés sont tous orientés : les arêtes (i, j) et (j, i) sont distinctes.

Exemple de graphe : $E = \{1, 2, 3, 4\}$

$\Gamma = \{(1, 2), (3, 4), (4, 2), (4, 3)\}$



Exemples de modélisation par des graphes :

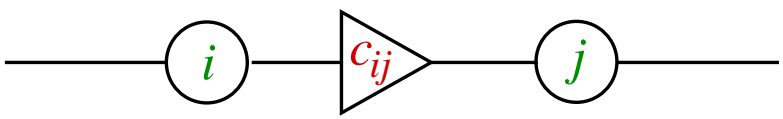
- réseau routier : les sommets sont les intersections des routes, les arêtes représentent les routes.
- cheminement dans un réseau informatique.
- Web modélisé par un graphe. Les sommets sont les pages Web et les arêtes sont les liens hypertexte entre ces différentes pages.

2) Graphe valué

$G = (E, \Gamma, c)$ est un **graphe valué** si (E, Γ) est un graphe auquel on associe une fonction positive $c : \Gamma \to \mathbb{R}^+$ appelée **capacité**.

La capacité de l'arête (i, j) est notée c_{ij}.

Notation :



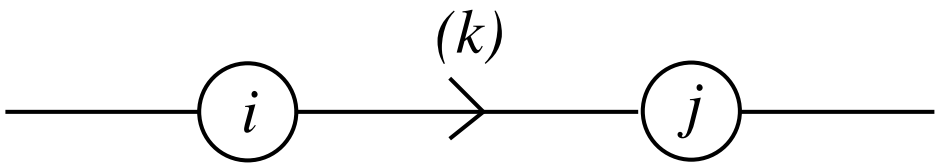
Exemple : La capacité c_{ij} représente par exemple la longueur du tronçon de route (i, j), le nombre max. de voitures par unité de temps entre deux villes i et j, la bande passante maximale entre les serveurs i et j...

3) Représentation d'un graphe

a) Matrice d'incidence sommet-arête

Soit un graphe **sans boucle** c-à-d sans arête (i, i), avec n sommets et m arêtes. On définit A la **matrice d'incidence** de taille $n \times m$:

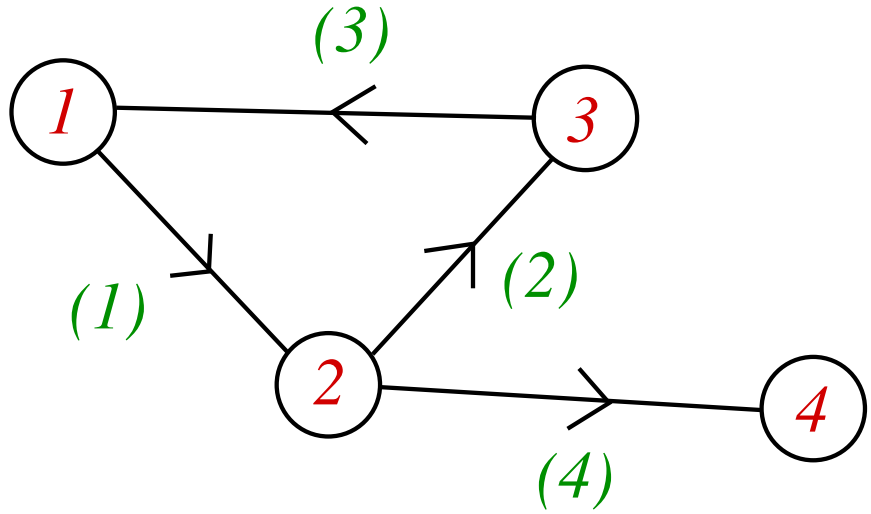
$$a_{ik} = \begin{cases} -1 & \text{si le sommet } i \text{ est l'extrémité } \textbf{initiale} \text{ de l'arête } k \\ +1 & \text{si le sommet } i \text{ est l'extrémité } \textbf{terminale} \text{ de l'arête } k \\ 0 & \text{sinon} \end{cases}$$



$$a_{ik} = -1$$
$$a_{jk} = +1$$

Exemple : $E = \{1, 2, 3, 4\}$

$\Gamma = \{(1, 2), (2, 3), (3, 1), (2, 4)\}$



Matrice d'incidence $A =$

	(1,2)	(2,3)	(3,1)	(2,4)
1	-1	0	+1	0
2	+1	-1	0	-1
3	0	+1	-1	0
4	0	0	0	+1

b) Matrice d'adjacence sommet-sommet

Matrice booléenne A de taille $n \times n$ (n sommets)

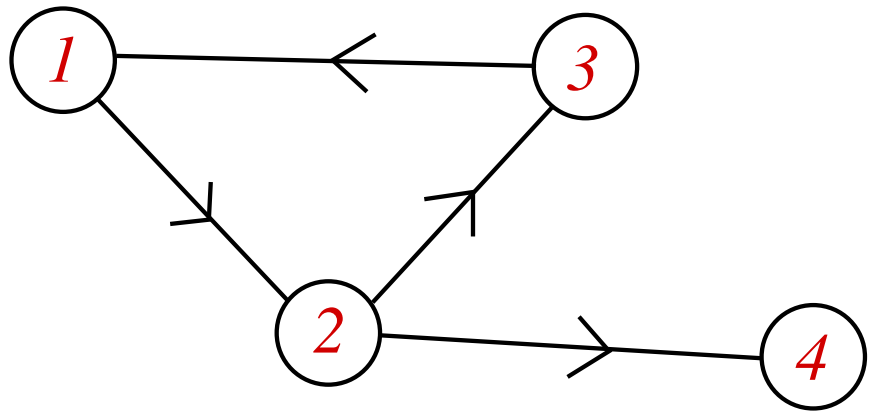
$$a_{ij} = \begin{cases} 1 & \text{si l'arête } (i,j) \text{ existe dans le graphe} \\ 0 & \text{sinon} \end{cases}$$

Variante pour un graphe valué par $\{c_{ij}\}$:

$$a_{ij} = \begin{cases} c_{ij} & \text{si l'arête } (i,j) \text{ existe dans le graphe} \\ 0 & \text{sinon} \end{cases}$$

Exemple : $E = \{1, 2, 3, 4\}$

$\Gamma = \{(1, 2), (2, 3), (3, 1), (2, 4)\}$



Matrice d'adjacence $A =$

	1	2	3	4
1	0	1	0	0
2	0	0	1	1
3	1	0	0	0
4	0	0	0	0

c) Listes d'adjacence : successeurs et prédécesseurs

Pour chaque sommet i du graphe, on définit

- la liste de ses **successeurs** $S(i)$: liste des sommets j tq l'arête (i, j) existe dans le graphe.
- la liste de ses **prédécesseurs** $P(i)$: liste des sommets j tq l'arête (j, i) existe dans le graphe.

c) Listes d'adjacence : successeurs et prédécesseurs

Pour chaque sommet i du graphe, on définit

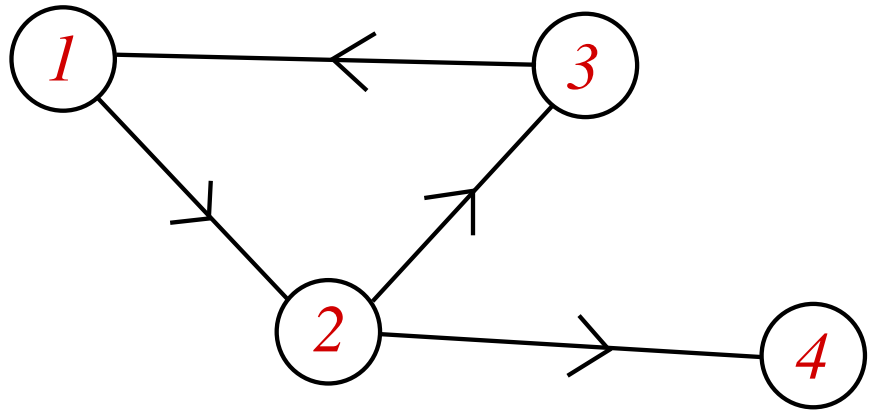
- la liste de ses **successeurs** $S(i)$: liste des sommets j tq l'arête (i, j) existe dans le graphe.
- la liste de ses **prédécesseurs** $P(i)$: liste des sommets j tq l'arête (j, i) existe dans le graphe.

Un sommet sans prédécesseur est appelé une **source**.

Un sommet sans successeur est appelé un **puits**

Exemple : $E = \{1, 2, 3, 4\}$

$\Gamma = \{(1, 2),\ (2, 3),\ (3, 1),\ (2, 4)\}$



sommet	successeur S	prédécesseur P
1	2	3
2	3, 4	1
3	1	2
4	–	2

4) Flot dans un graphe

Problèmes de circulation d'objets (voiture, information ...) dans un réseau (routier, informatique ...).

Définition

Soit $G = (E, \Gamma, c)$ un graphe valué comportant un seul sommet source s et un seul sommet puits t.

- Un **flot** de s à t est une fonction $f : \Gamma \to \mathbb{R}$ tq

$$\sum_{i \in P(j)} f_{ij} = \sum_{k \in S(j)} f_{jk} \quad \text{où} \quad f_{ij} \overset{def}{=} f(i,j)$$

pour tout sommet $j \neq s, t$. On dit qu'il y a conservation du flux au sommet j ("ce qui rentre égale ce qui sort").

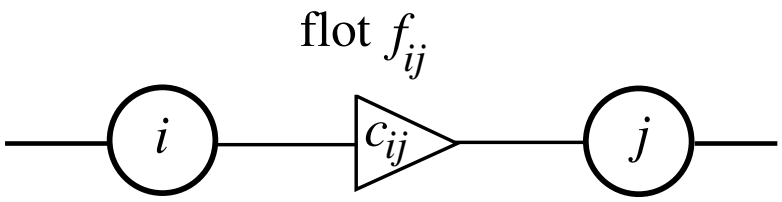
La valeur $f_{ij} \overset{def}{=} f(i,j)$ est le **flot dans l'arête** (i,j).

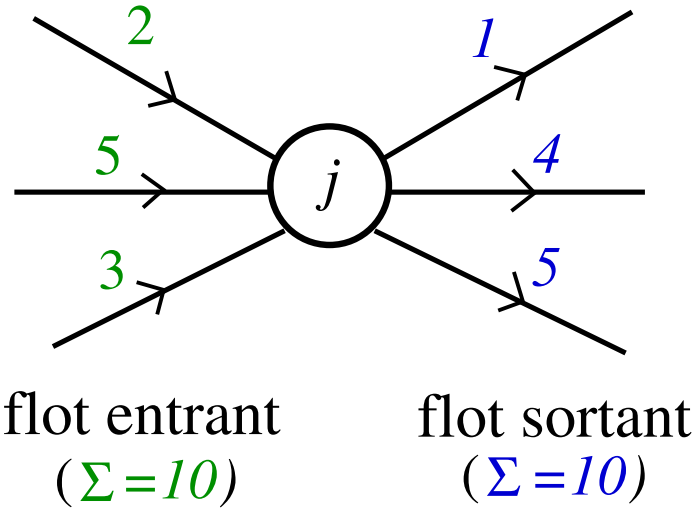
Définition (suite)

- Le flot est dit **réalisable** si pour toute arête $(i, j) \in \Gamma$, on a

$$0 \leq f_{ij} \leq c_{ij}$$

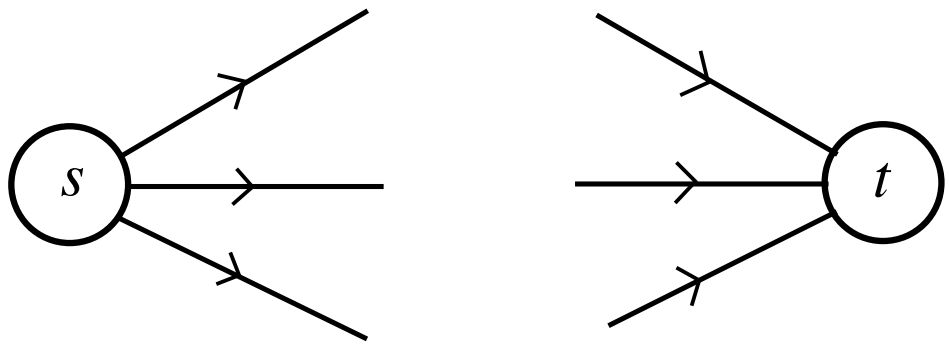
- La quantité $v = \sum_{i \in P(t)} f_{it}$ est la **valeur du flot** de s à t.





Remarque (rappels) :

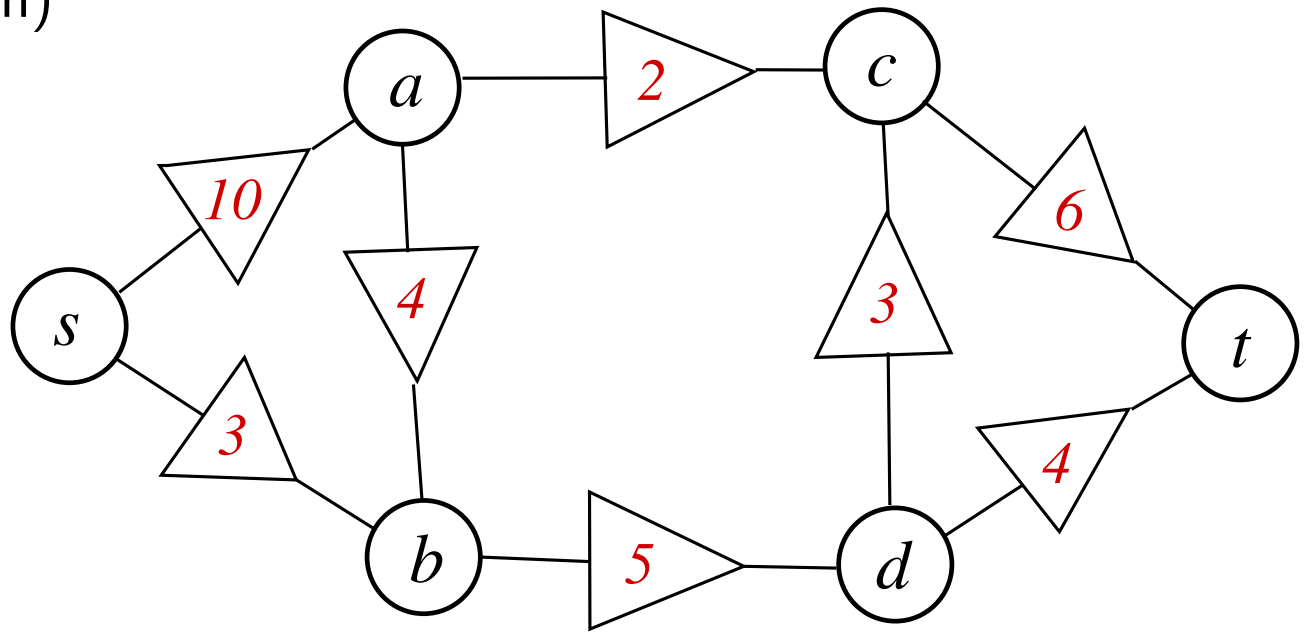
- $S(i)$: ensemble des sommets j **successeurs** du sommet i c-à-d tq l'arête (i, j) existe dans le graphe.
- $P(i)$: ensemble des sommets j **prédécesseurs** du sommet i c-à-d tq l'arête (j, i) existe dans le graphe.
- Une **source** s (resp. un **puits** t) est un sommet ne possédant pas de prédécesseur (resp. de successeur).



II. Problème de flot maximal dans un graphe

1) Introduction

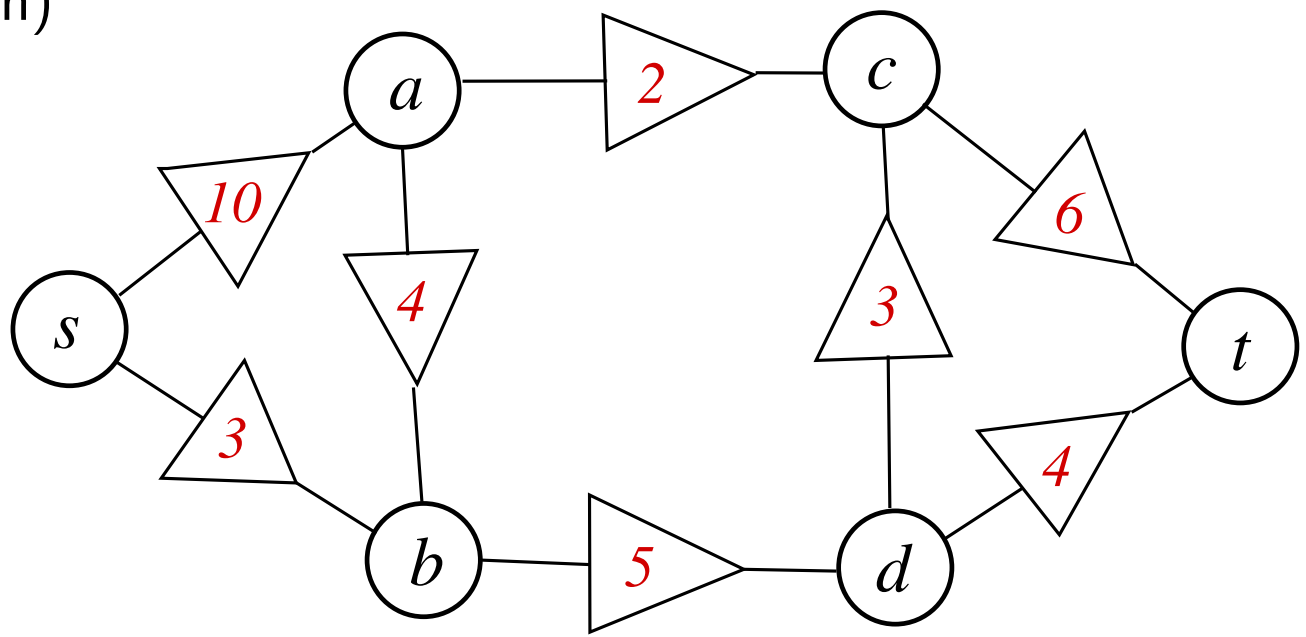
On veut par ex. trouver le trafic maximal entre deux villes d'un réseau routier dont on connait la capacité (nb de voiture par heure sur chaque tronçon)



II. Problème de flot maximal dans un graphe

1) Introduction

On veut par ex. trouver le trafic maximal entre deux villes d'un réseau routier dont on connait la capacité (nb de voiture par heure sur chaque tronçon)



Problème de flot maximal

Etant donné un graphe valué possédant une seule source et un seul puits , trouver un flot réalisable maximal (i.e. dont la valeur est maximale).

Flot maximal et programmation linéaire

$$\max_{f_{ij},v} [F = v]$$

Flot maximal et programmation linéaire

conservation des flux en chaque sommet :

$$\max_{f_{ij},v} [F = v]$$

$$\begin{cases} \sum_{k \in S(s)} f_{sk} - v = 0 \quad (\text{source } s) \\ -\sum_{i \in P(j)} f_{ij} + \sum_{k \in S(j)} f_{jk} = 0, \quad \forall j \neq s, t \\ -\sum_{i \in P(t)} f_{it} + v = 0 \quad (\text{puits } t) \end{cases}$$

Flot maximal et programmation linéaire

$$\max_{f_{ij},v} [F = v]$$

conservation des flux en chaque sommet :

$$\begin{cases} \displaystyle\sum_{k \in S(s)} f_{sk} - v = 0 \quad (\text{source } s) \\ \displaystyle -\sum_{i \in P(j)} f_{ij} + \sum_{k \in S(j)} f_{jk} = 0, \quad \forall j \neq s, t \\ \displaystyle -\sum_{i \in P(t)} f_{it} + v = 0 \quad (\text{puits } t) \end{cases}$$

respect des capacités : $f_{ij} \leq c_{ij}$ pour toute arête $(i, j) \in \Gamma$

Flot maximal et programmation linéaire

$$\max_{f_{ij},v} [F = v]$$

conservation des flux en chaque sommet :

$$\begin{cases} \sum_{k \in S(s)} f_{sk} - v = 0 \quad (\text{source } s) \\ -\sum_{i \in P(j)} f_{ij} + \sum_{k \in S(j)} f_{jk} = 0, \quad \forall j \neq s, t \\ -\sum_{i \in P(t)} f_{it} + v = 0 \quad (\text{puits } t) \end{cases}$$

respect des capacités : $f_{ij} \leq c_{ij}$ pour toute arête $(i,j) \in \Gamma$

contrainte de signe : $f_{ij} \geq 0$ pour toute arête $(i,j) \in \Gamma$

Remarque : les inconnues sont les f_{ij} et la valeur v du flot.

Flot maximal et programmation linéaire

$$\max_{f_{ij},v} [F = v]$$

conservation des flux en chaque sommet :

$$\begin{cases} \sum_{k \in S(s)} f_{sk} - v = 0 \quad (\text{source } s) \\ -\sum_{i \in P(j)} f_{ij} + \sum_{k \in S(j)} f_{jk} = 0, \quad \forall j \neq s, t \\ -\sum_{i \in P(t)} f_{it} + v = 0 \quad (\text{puits } t) \end{cases}$$

respect des capacités : $f_{ij} \leq c_{ij}$ pour toute arête $(i,j) \in \Gamma$

contrainte de signe : $f_{ij} \geq 0$ pour toute arête $(i,j) \in \Gamma$

Remarque : les inconnues sont les f_{ij} et la valeur v du flot.

Flot maximal et programmation linéaire

Ecriture matricielle (n sommets et m arêtes)

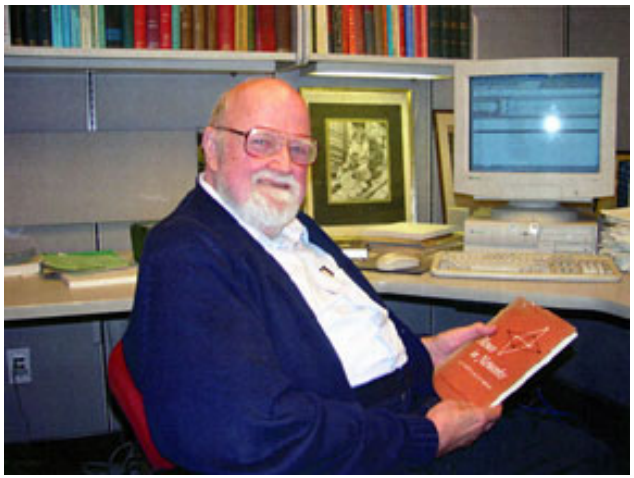
$$\max_{\mathbf{f},v} [F = v]$$

$$\begin{cases} A\mathbf{f} + \mathbf{v} = 0 \\ \mathbf{f} \leq \mathbf{c} \\ \mathbf{f} \geq 0 \end{cases}$$

A est la matrice d'incidence du graphe, de taille $n \times m$,

$$\mathbf{f} = \begin{pmatrix} (f_{sk})_{k \in S(s)} \\ \vdots \\ f_{ij} \\ \vdots \\ (f_{it})_{i \in P(t)} \end{pmatrix} \in \mathbb{R}^m; \quad \mathbf{v} = \begin{pmatrix} +v \\ 0 \\ \vdots \\ 0 \\ -v \end{pmatrix} \in \mathbb{R}^n$$

2) Théorème de Ford-Fulkerson



L. R. Ford (1927– 2017)

D. R. Fulkerson (1924–1976)

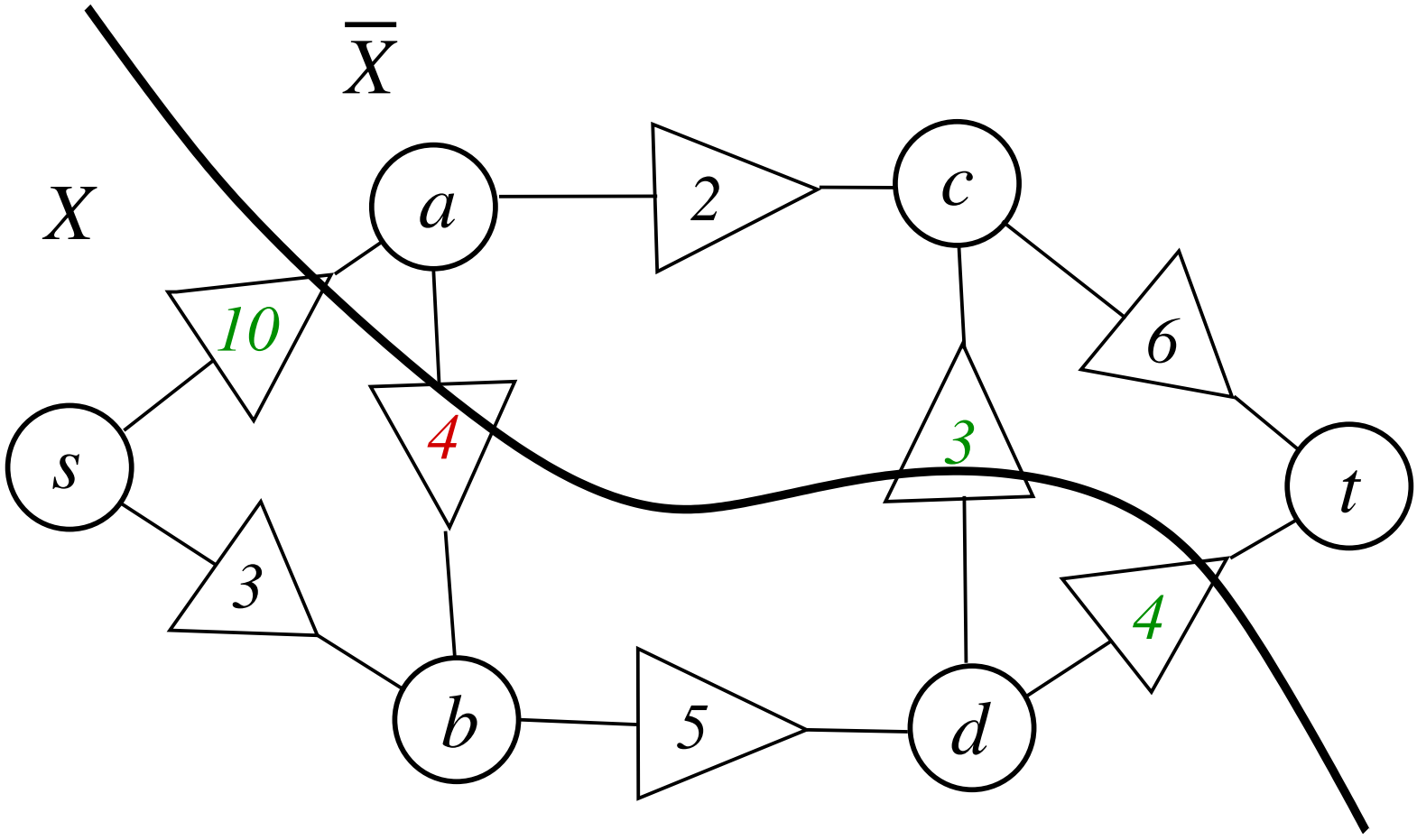
- Ford, L. R., Jr. ; Fulkerson, D. R. (1956), *Maximal flow through a network*, Canadian Journal of Mathematics 8 : 399–404.
- L. R. Ford ; D. R. Fulkerson (1962). Flows in Networks.

Définition

Une **coupe** d'un graphe valué $G = (E, \Gamma, c)$ possédant un seul sommet source s et un seul sommet puits t, est une partition des sommets notée $(X, \overline{X})$ telle que :

- $E = X \cup \overline{X}$
- $X \cap \overline{X} = \emptyset$
- $s \in X$ et $t \in \overline{X}$

La **capacité de la coupe** est définie par $c(X, \overline{X}) = \sum_{\substack{i \in X \\ j \in \overline{X}}} c_{ij}$



Capacité de la coupe $c(X, \overline{X}) = 10 + 3 + 4 = 17$.

On peut comparer la valeur d'un flot avec la capacité d'une coupe du graphe.

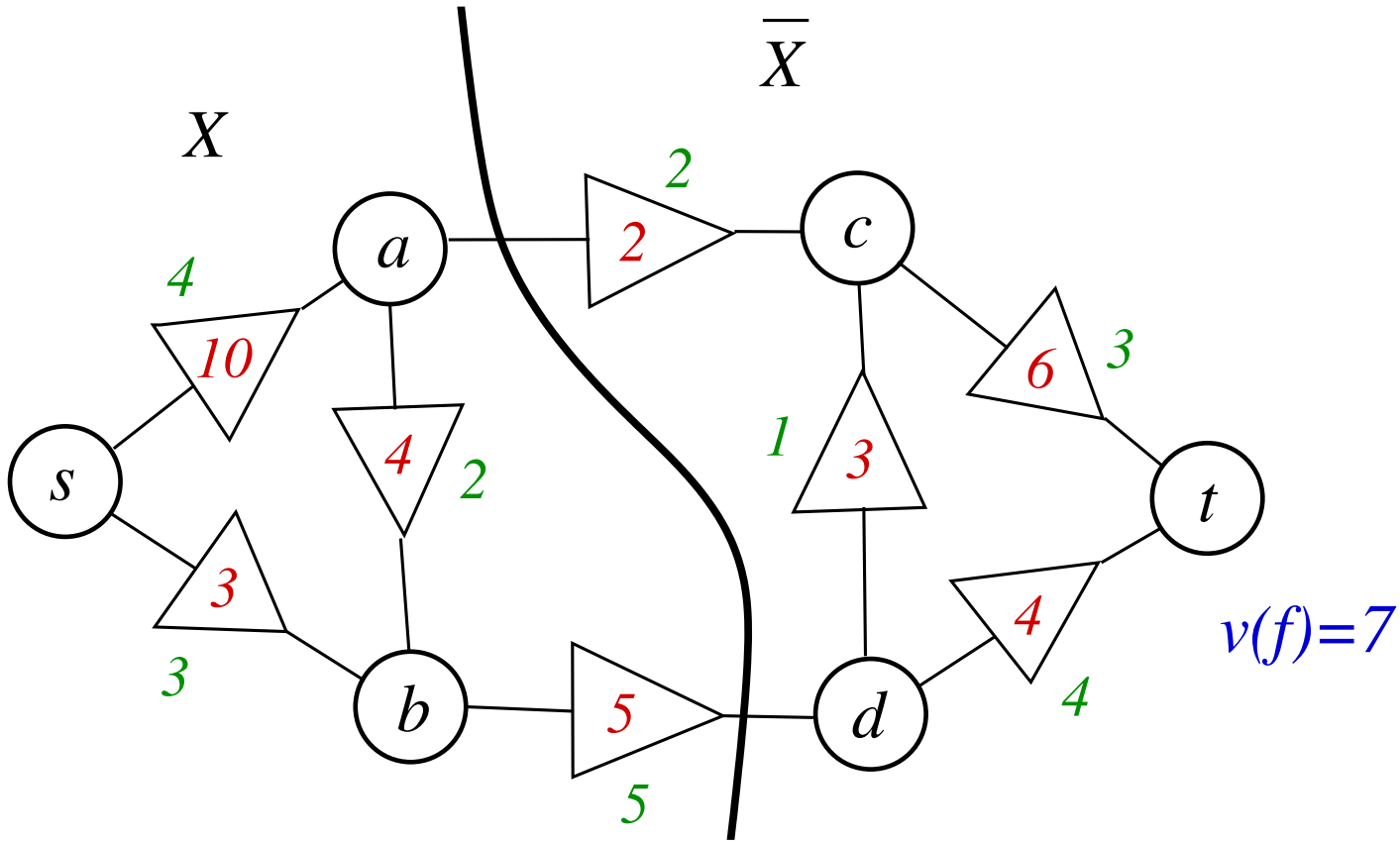
Théorème de Ford-Fulkerson

Soit $G = (E, \Gamma, c)$ un graphe valué. Pour tout flot réalisable f et toute coupe $(X, \overline{X})$, on a

$$v(f) \leq c(X, \overline{X})$$

où $v(f)$ est la valeur du flot f.

Le théorème de Ford-Fulkerson permet de savoir si un flot est maximal ou non. Par exemple :



$v(f) = 7$ et $c(X, \overline{X}) = 7 \Rightarrow$ flot maximal.

Démonstration du théorème de Ford-Fulkerson

Convention : si l'arête (i, j) n'existe pas dans le graphe, on pose $f_{ij} = 0$.

$$\Rightarrow v(f) = \sum_{j \in S(s)} f_{sj} = \sum_{j \in E} f_{sj}.$$

On montre que

$$v(f) = \sum_{\substack{i \in X \\ j \in \overline{X}}} f_{ij} - \sum_{\substack{i \in X \\ k \in \overline{X}}} f_{ki}$$

flot de X à $\overline{X}$ — flot de $\overline{X}$ à X

$$\Rightarrow v(f) \leq \sum_{\substack{i \in X \\ j \in \overline{X}}} c_{ij} = c(X, \overline{X})$$

Démonstration de la relation $v(f) = \sum_{\substack{i\in X\\ j\in\overline{X}}} f_{ij} - \sum_{\substack{i\in X\\ k\in\overline{X}}} f_{ki}$.

D'après la conversation des flux : $\sum_{j\in E} f_{ij} - \sum_{k\in E} f_{ki} = 0$ pour tout $i \neq s, t$.

On somme sur $i \in X, i \neq s$: $\sum_{\substack{i\in X\\ i\neq s}} \left(\sum_{j\in E} f_{ij} - \sum_{k\in E} f_{ki} \right) = 0$

soit

$$\sum_{i\in X} \left(\sum_{j\in E} f_{ij} - \sum_{k\in E} f_{ki} \right) - \underbrace{\sum_{j\in E} f_{sj}}_{=v(f)} - \sum_{k\in E} \underbrace{f_{ks}}_{=0} = 0.$$

On obtient

$$v(f) = \sum_{\substack{i\in X\\ j\in X}} \cancel{f_{ij}} + \sum_{\substack{i\in X\\ j\in\overline{X}}} f_{ij} - \sum_{\substack{i\in X\\ k\in X}} \cancel{f_{ki}} - \sum_{\substack{i\in X\\ k\in\overline{X}}} f_{ki} = \sum_{\substack{i\in X\\ j\in\overline{X}}} f_{ij} - \sum_{\substack{i\in X\\ k\in\overline{X}}} f_{ki}$$

□

3) Coupe minimale

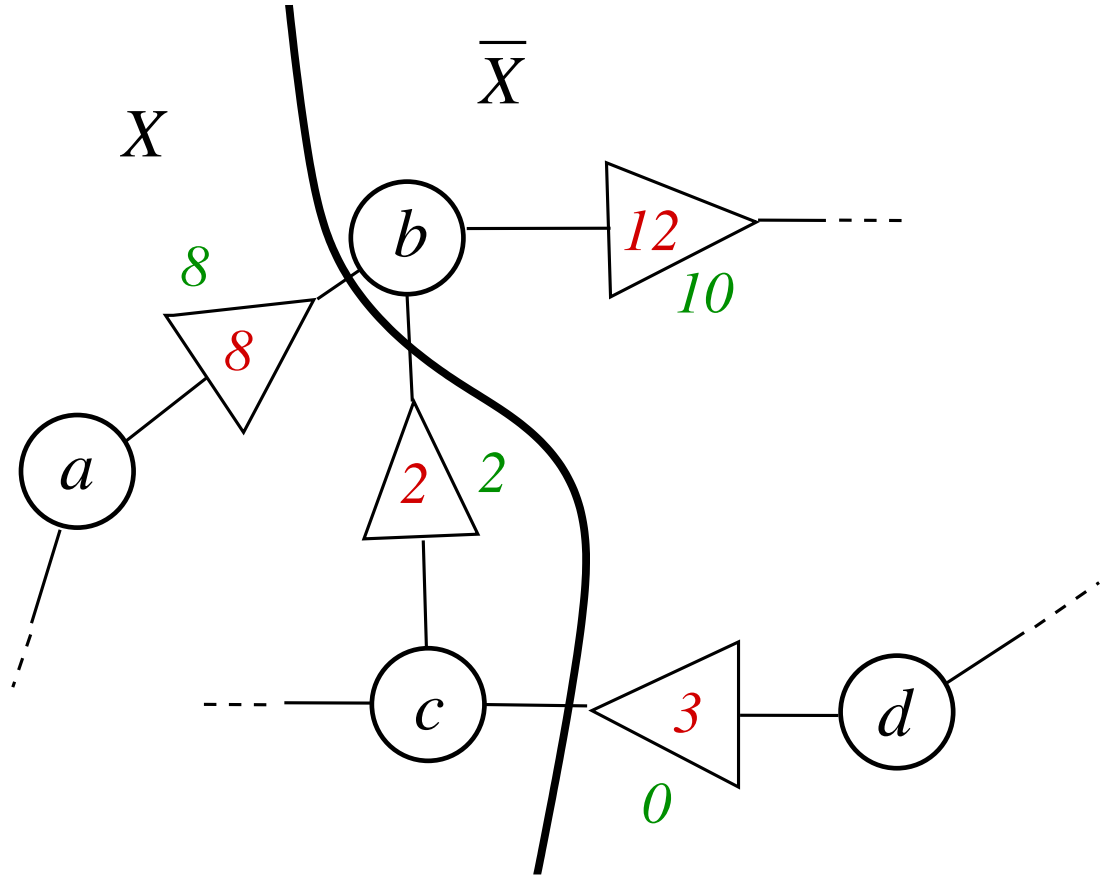
Le Théorème de Ford-Fulkerson admet un corollaire qui donne une condition suffisante pour avoir un flot maximal.

On dit qu'une arête $(i, j) \in \Gamma$ est **saturée** si $f_{ij} = c_{ij}$ et qu'elle est **insaturée** si $f_{ij} = 0$

Définition

Une coupe $(X, \overline{X})$ est dite **minimale** pour f si toute arête de X vers $\overline{X}$ est saturée et toute arête de $\overline{X}$ vers X est insaturée.

Coupe minimale $(X, \overline{X})$



- arêtes (a, b) et (c, b) **saturées**
- arête (d, c) **insaturée**

Proposition

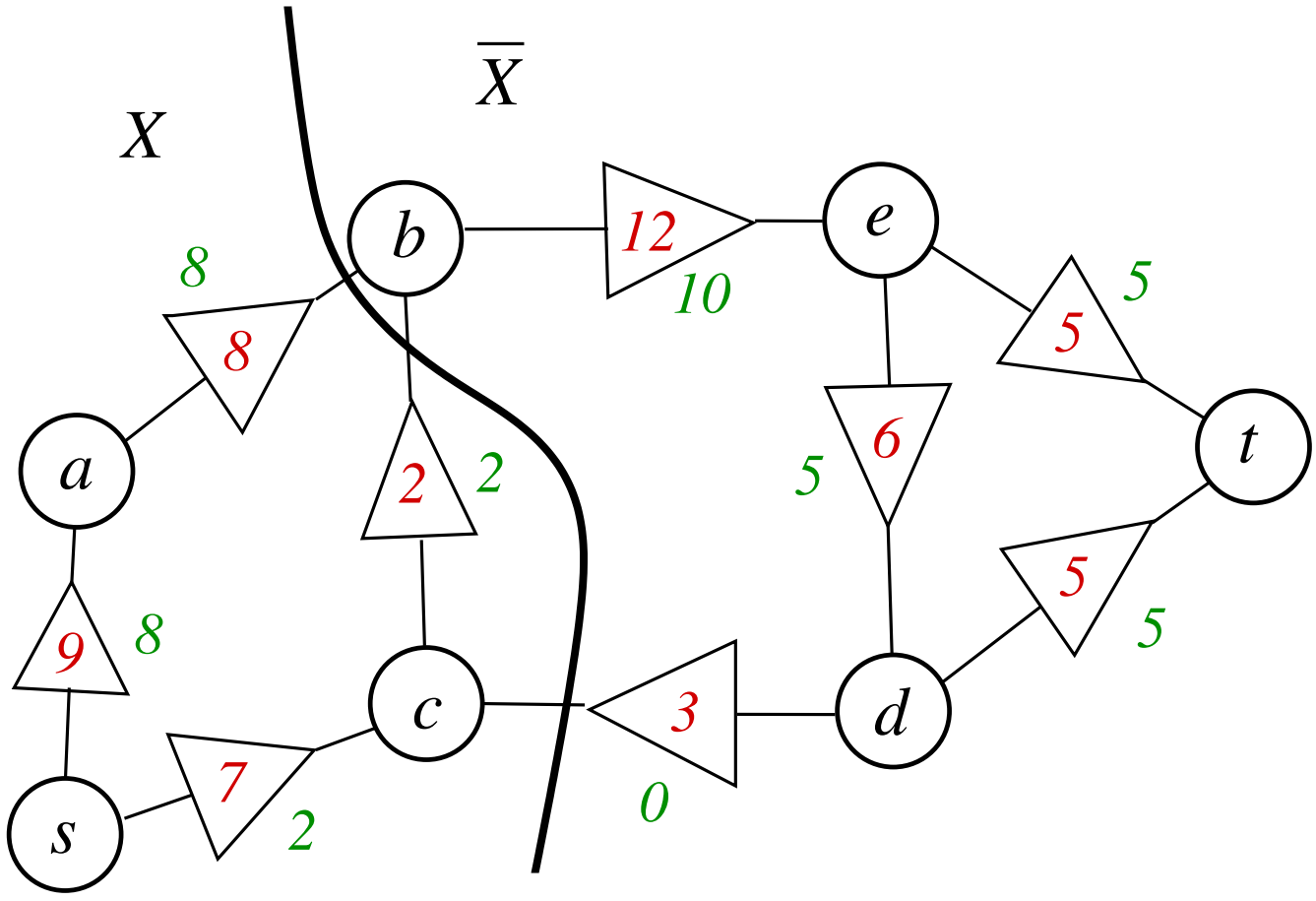
S'il existe une coupe minimale pour un flot f, alors ce flot est maximal.

Preuve. A partir de la formule établie dans le th. de Ford-Fulkerson :

$$v(f) = \sum_{\substack{i \in X \\ j \in \overline{X}}} \overbrace{f_{ij}}^{=c_{ij}} - \sum_{\substack{i \in X \\ k \in \overline{X}}} \overbrace{f_{ki}}^{=0} = c(X, \overline{X})$$

$\Rightarrow$ $v(f)$ est maximal. $\square$

Coupe minimale / flot maximal



$\Rightarrow$ flot maximal $v(f) = c(X, \overline{X}) = 10$

1) Condition nécessaire et suffisante de flot maximal

Définition 1.

Une chaîne d'un graphe est une suite de sommets

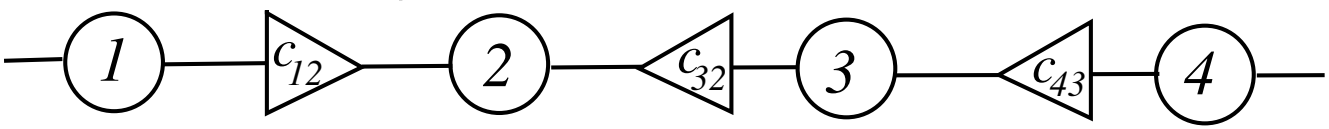
$$\mathcal{C} = (i_1, i_2, \cdots, i_p, i_{p+1}, \cdots, i_q)$$

reliés entre eux par des arêtes c'est-à-dire tels que

$$(i_p, i_{p+1}) \in \Gamma \quad \text{ou} \quad (i_{p+1}, i_p) \in \Gamma$$

(arête **directe**) (arête **inverse**)

Une chaîne ne tient pas compte de l'orientation des arêtes reliant les sommets (chaîne $\neq$ chemin).

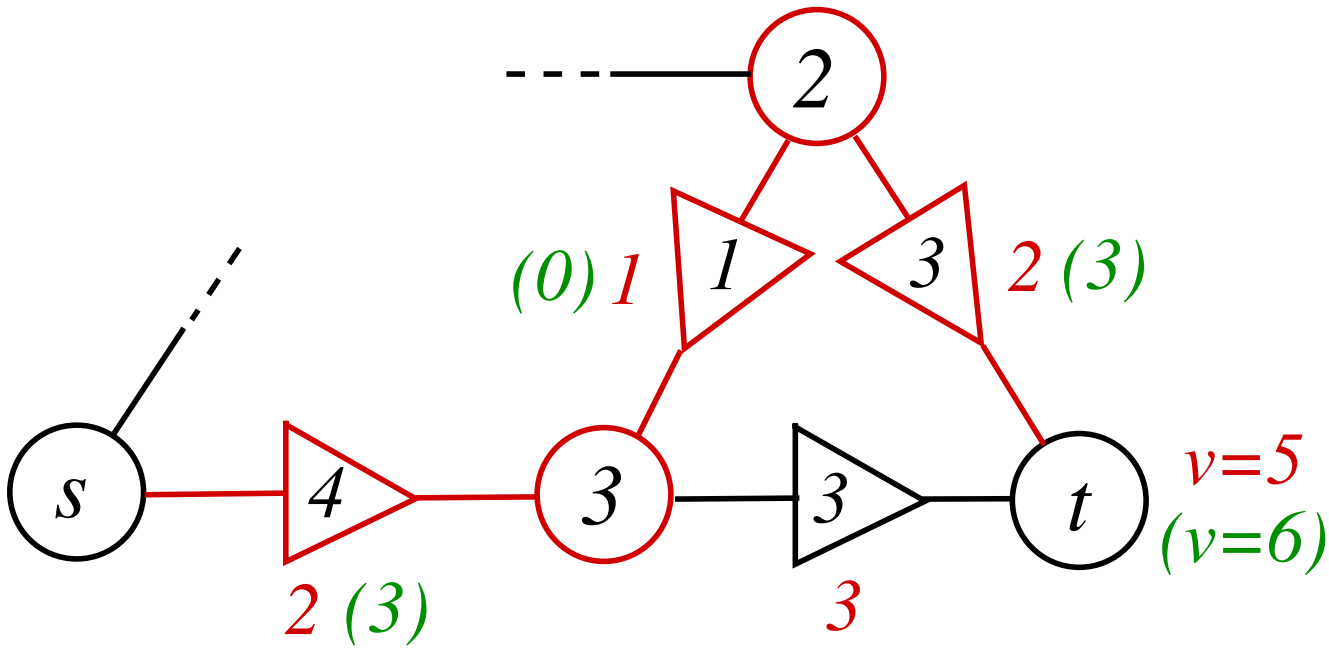


Définition 2.

Soit $G = (E, \Gamma, c)$ un graphe valué possédant une seule source s et un seul puits t. Une chaîne $\mathcal{C} = (s, i_1, i_2, \cdots, i_p, i_{p+1}, \cdots, i_q, t)$ est dite **améliorante** pour un flot réalisable f donné si :

- $f(i_p, i_{p+1}) < c(i_p, i_{p+1})$ si $(i_p, i_{p+1}) \in \Gamma$ (arête directe)
- $f(i_{p+1}, i_p) > 0$ si $(i_{p+1}, i_p) \in \Gamma$ (arête inverse)

Remarque : ce qui compte ici, ce sont les inégalites strictes.



L'algorithme de Ford-Fulkerson permet de trouver un flot maximal par recherche de chaînes améliorantes. Il est basé sur le résultat suivant :

Théorème

Un flot réalisable est maximal si et seulement s'il n'existe pas de chaîne améliorante.

L'algorithme de Ford-Fulkerson permet de trouver un flot maximal par recherche de chaînes améliorantes. Il est basé sur le résultat suivant :

Théorème

Un flot réalisable est maximal si et seulement s'il n'existe pas de chaîne améliorante.

Preuve.

***i*) Condition nécessaire**

Soit f un flot réalisable maximal. On suppose par l'absurde qu'il existe une chaîne améliorante $\mathcal{C} = (s, i_1, i_2, \cdots, i_p, i_{p+1}, \cdots, i_q, t)$. On note

$$\varepsilon_1 = \min\{c(i_p, i_{p+1}) - f(i_p, i_{p+1}) \text{ tel que } (i_p, i_{p+1}) \in \Gamma \text{ (arête directe)}\}$$

$$\varepsilon_2 = \min\{f(i_{p+1}, i_p) \text{ tel que } (i_{p+1}, i_p) \in \Gamma \text{ (arête inverse)}\}$$

$$\rightarrow \boxed{\varepsilon = \min\{\varepsilon_1, \varepsilon_2\} > 0}$$

ε représente l'amélioration qu'on peut apporter au flot.

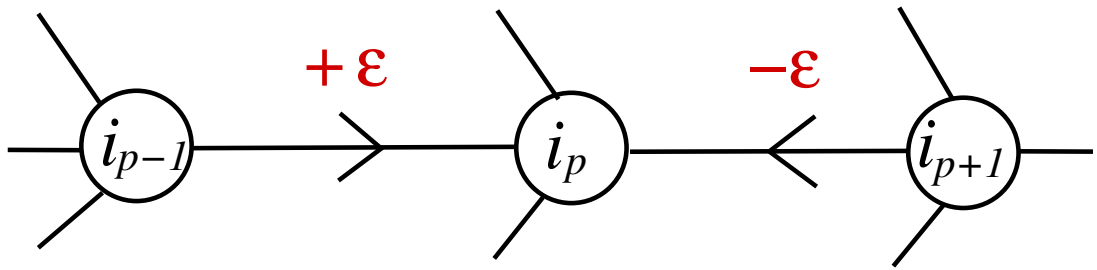
Nouveau flot f' qui coïncide avec f en dehors de la chaîne améliorante. Sur les arêtes de la chaîne :

- Si $(i_p, i_{p+1}) \in \Gamma$ (arête directe) alors

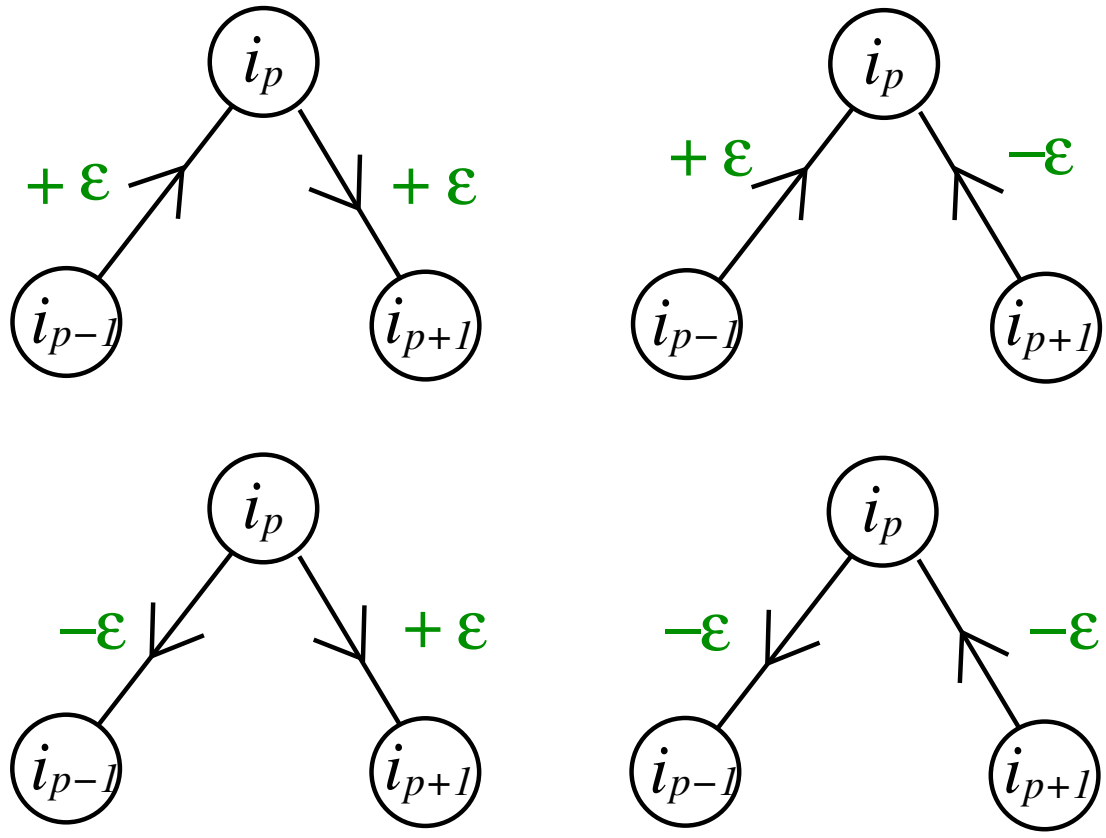
$$f'(i_p, i_{p+1}) = f(i_p, i_{p+1}) + \varepsilon$$

- Si $(i_{p+1}, i_p) \in \Gamma$ (arête inverse) alors

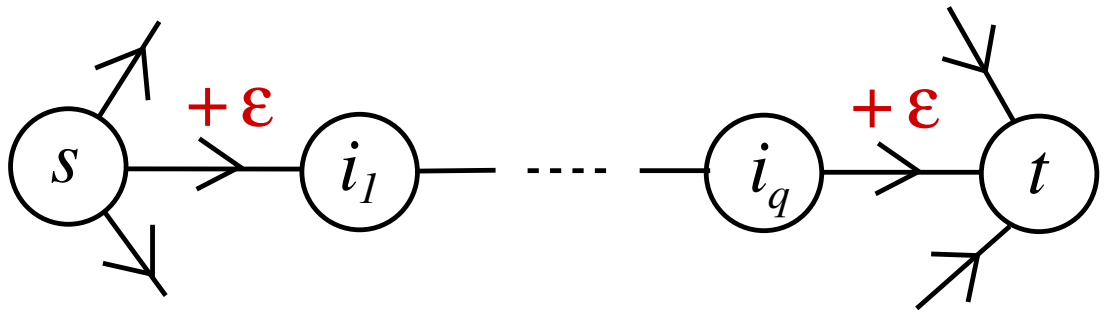
$$f'(i_p, i_{p+1}) = f(i_p, i_{p+1}) - \varepsilon$$



- Le nouveau flot f' est bien **réalisable**. En particulier, il y a conservation des flux en chaque sommet : 4 cas possibles



- Le nouveau flot f' est augmenté de $+\varepsilon$ quand il arrive au puits t : $v(f') = v(f) + \varepsilon$.



$\Rightarrow$ le flot f n'est pas maximal $\Rightarrow$ contradiction. $\square$

ii) **Condition suffisante**

On suppose qu'il n'existe pas de chaîne améliorante. On va montrer que le flot est maximal en trouvant une coupe $(X, \overline{X})$ telles que $v(f) = c(X, \overline{X})$ (Th. Ford-Fulkerson).

Construction de la coupe $(X, \overline{X})$

X est l'ensemble des sommets qui sont **marqués** de la façon suivante :

1. On marque la source s
2. A partir de tous les sommets i marqués, marquer tous les sommets j non encore marqués tels que

$$f(i,j) < c(i,j) \text{ (arête directe)} \quad \text{ou} \quad f(j,i) > 0 \text{ (arête inverse)}$$

3. Recommencer en 2) jusqu'à ce qu'il n'y ait plus de marquage possible.

$\Rightarrow$ A l'issue du marquage, on a $v(f) = c(X, \overline{X})$. $\square$

Remarque : le puits t ne peut pas être marqué sinon il y aurait une chaîne améliorante.

2) Algorithme de Ford-Fulkerson

- Initialisation par un flot initial réalisable ($f = 0$)
- Tant que le flot n'est pas maximal
  - Marquage de la source s
  - Tant qu'on marque des sommets

    Pour tout sommet marqué i

    Marquer les sommets j non marqués tq $f(i,j) < c(i,j)$ ou $f(j,i) > 0$

    Fin pour

    Fin Tant que
  - Si le puits t n'est pas marqué alors le flot est maximal
    Sinon amélioration du flot

  Fin Tant que

Amélioration du flot

- trouver une chaîne qui a permis de marquer t et calculer $\varepsilon = \min(\varepsilon_1, \varepsilon_2) > 0$ avec

$$\varepsilon_1 = \min\{c(i_p, i_{p+1}) - f(i_p, i_{p+1}) \text{ avec } (i_p, i_{p+1}) \in \Gamma \text{ (arête directe)}\}$$

$$\varepsilon_2 = \min\{f(i_{p+1}, i_p) \text{ avec } (i_{p+1}, i_p) \in \Gamma \text{ (arête inverse)}\}$$

- trouver le nouveau flot f' :
  - Si $(i_p, i_{p+1}) \in \Gamma$ (arête directe) alors $f'(i_p, i_{p+1}) = f(i_p, i_{p+1}) + \varepsilon$
  - Si $(i_{p+1}, i_p) \in \Gamma$ (arête inverse) alors $f'(i_p, i_{p+1}) = f(i_p, i_{p+1}) - \varepsilon$

3) Parcours de graphe

- **Parcours profondeur (DFS)**

3) Parcours de graphe

- **Parcours profondeur (DFS)**

  Exploration en profondeur des chemins : pour chaque sommet, on prend et on marque le premier sommet successeur jusqu'à ce qu'un sommet n'ait plus de successeur ou bien que tous ses successeurs soient déjà marqués.

3) Parcours de graphe

- **Parcours profondeur (DFS)**

  Exploration en profondeur des chemins : pour chaque sommet, on prend et on marque le premier sommet successeur jusqu'à ce qu'un sommet n'ait plus de successeur ou bien que tous ses successeurs soient déjà marqués.

  On utilise généralement **une pile** pour l'exploration des sommets.

3) Parcours de graphe

- **Parcours profondeur (DFS)**

  Exploration en profondeur des chemins : pour chaque sommet, on prend et on marque le premier sommet successeur jusqu'à ce qu'un sommet n'ait plus de successeur ou bien que tous ses successeurs soient déjà marqués.

  On utilise généralement **une pile** pour l'exploration des sommets.

  Utilisations :

  - Pour un graphe non-orienté, calcul des *composantes connexes*.
  - Pour un graphe orienté sans cycle, *tri topologique* des sommets : ordre des sommets tel qu'un sommet est toujours visité avant ses successeurs. En dépilant, on obtient un tri topologique (en ordre inverse).

III. Algorithme de Ford-Fulkerson

- **Parcours largeur (BFS)**

III. Algorithme de Ford-Fulkerson

- **Parcours largeur (BFS)**

  A partir d'un sommet, on liste et on marque tous les sommets successeurs non encore marqués, jusqu'à ce qu'un sommet n'ait plus de successeur ou bien que tous ses successeurs soient déjà marqués.

- **Parcours largeur (BFS)**

  A partir d'un sommet, on liste et on marque tous les sommets successeurs non encore marqués, jusqu'à ce qu'un sommet n'ait plus de successeur ou bien que tous ses successeurs soient déjà marqués.

  On utilise généralement **une file** (liste FIFO) pour l'exploration des sommets.

- **Parcours largeur (BFS)**

  A partir d'un sommet, on liste et on marque tous les sommets successeurs non encore marqués, jusqu'à ce qu'un sommet n'ait plus de successeur ou bien que tous ses successeurs soient déjà marqués.

  On utilise généralement **une file** (liste FIFO) pour l'exploration des sommets.

  Utilisations :

  - Le parcours en largeur explore tous les sommets accessibles depuis le sommet initial $\Rightarrow$ calcul des *composantes connexes*.
  - Recherche du plus court chemin (nb minimum d'arêtes) entre deux sommets.

Algorithme BFS et plus court chemin

Initialement, tous les sommets sont non-marqués et la file est vide.

Marquer et insérer le sommet s de départ dans la file.
Initialisation de la distance $D(s) = 0$.

Tant que la file n'est pas vide
- Supprimer le sommet P situé en tête de file.
- Pour chaque successeur non marqué Q de P,
  - Marquer et insérer Q dans la file
  - Calcul de la distance de Q à s : $D(Q) = D(P) + 1$.

Fin Pour

Fin Tant que

Algorithme BFS et plus court chemin

Initialement, tous les sommets sont non-marqués et la file est vide.

Marquer et insérer le sommet s de départ dans la file.
Initialisation de la distance $D(s) = 0$.

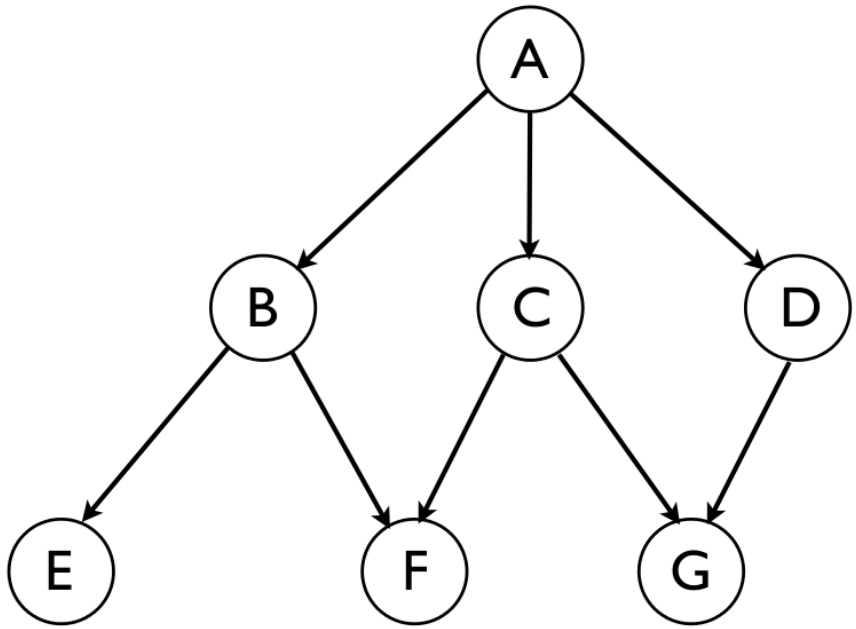
Tant que la file n'est pas vide
- Supprimer le sommet P situé en tête de file.
- Pour chaque successeur non marqué Q de P,
  - Marquer et insérer Q dans la file
  - Calcul de la distance de Q à s : $D(Q) = D(P) + 1$.

Fin Pour

Fin Tant que

☞ On obtient la liste des sommets accessibles à partir de s (sommets marqués) et D est la distance la plus courte (nb minimum d'arêtes) de chaque sommet à s.

Exemple de parcours DFS / BFS



Parcours profondeur DFS : A,B,E,F,C,G,D (*tri topologique A,D,C,G,B,F,E*)

Parcours largeur BFS : A,B,C,D,E,F,G

Remarque. Dans la recherche d'une chaîne améliorante de l'algorithme de Ford Fulkerson, le parcours du graphe se fait en considérant non pas les successeurs d'un sommet mais les sommets voisins **accessibles au sens d'une chaîne améliorante**.

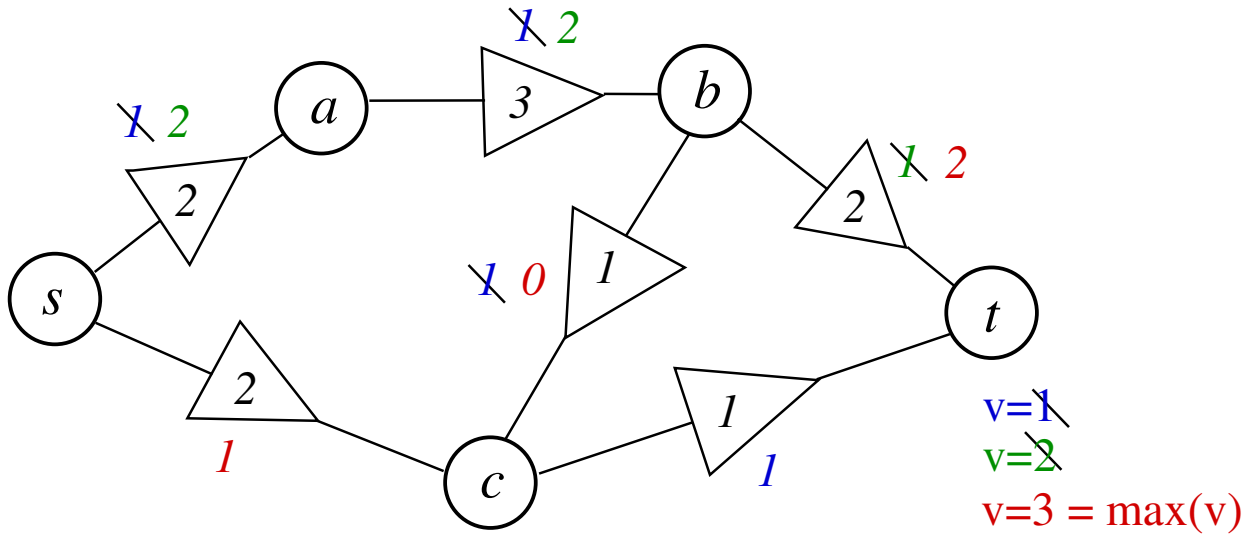
4) Un exemple

Dans la pratique, on peut utiliser plusieurs listes/tableaux :

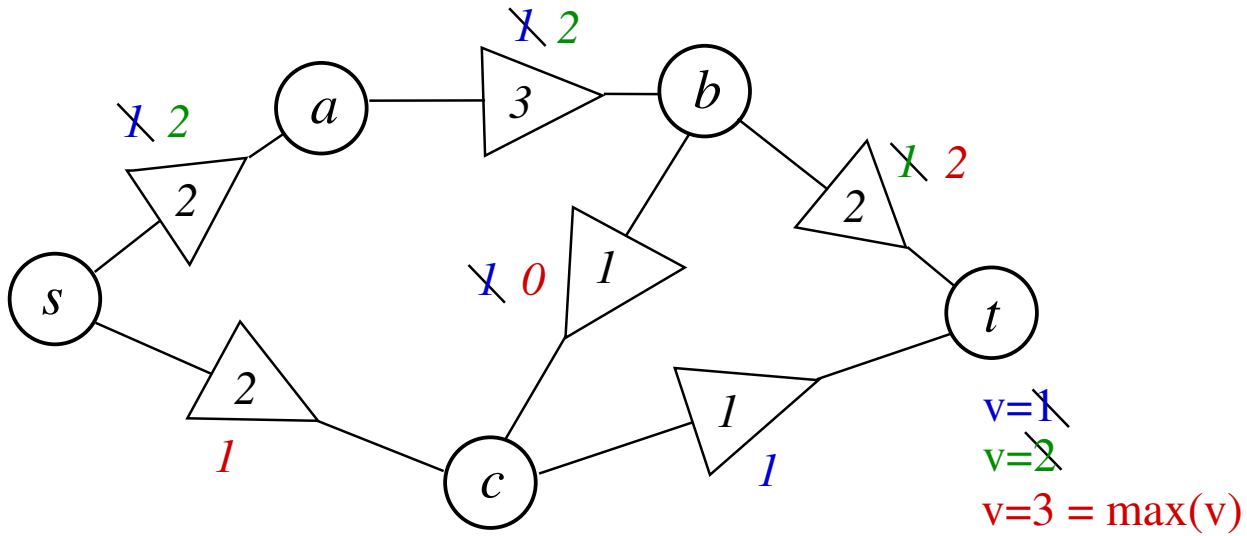
- $\mathbb{E} = (\textit{sommets examinés (marqués) selon un certain parcours})$
- pour un sommet $\mathbb{E}[k] = j$, $\textit{orig}[k]$ indique le sommet précédent dans la future (possible) chaîne en construction.
- $\varepsilon[k]$ est l'amélioration du flot possible jusqu'à $\mathbb{E}[k]$
- le tableau *sens* : posons $\textit{orig}[k] = i$ alors :

  (arête directe) $\textit{sens}[k] = 1$ si $(i, j) \in \Gamma$

  (arête inverse) $\textit{sens}[k] = -1$ si $(j, i) \in \Gamma$



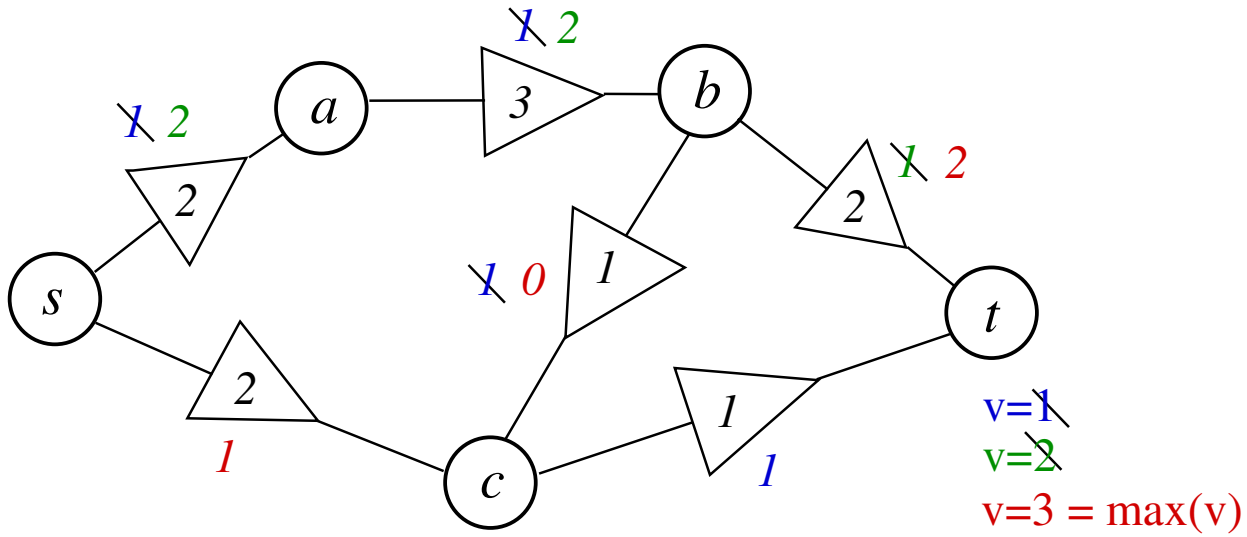
Marquage pile profondeur (DFS)



Marquage pile profondeur (DFS)

Etape 1 :

$\mathbb{E}$	*s*	*a*	*b*	*c*	*t*
orig	–	*s*	*a*	*b*	*c*
ε	∞	2	2	1	$\varepsilon = 1$
sens	–	1	1	1	1



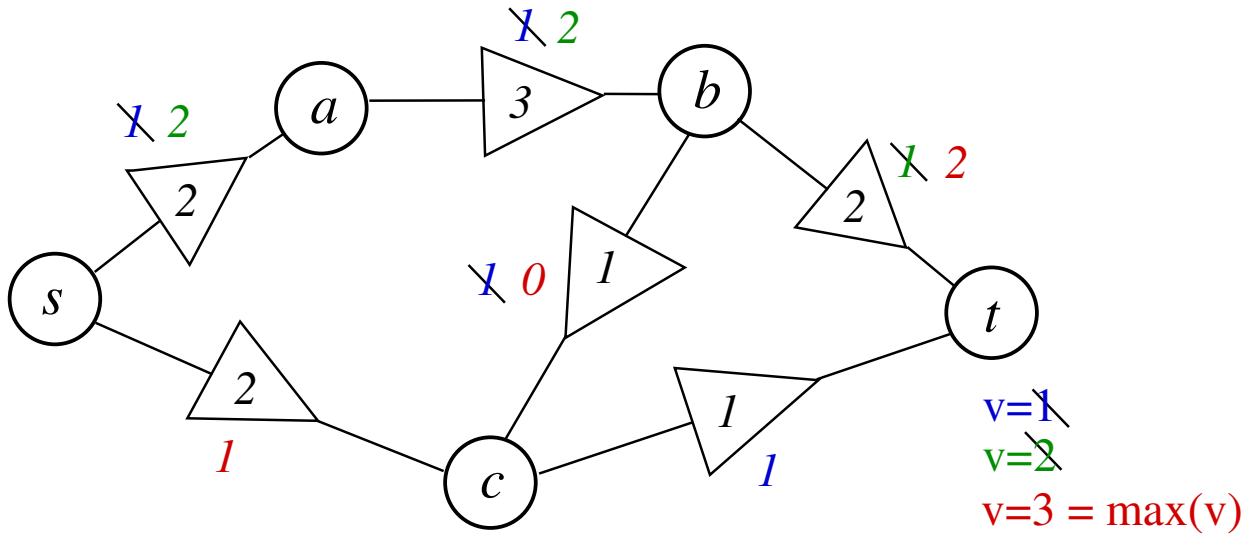
Marquage pile profondeur (DFS)

Etape 1 :

$\mathbb{E}$	s	a	b	c	t
orig	–	s	a	b	c
ε	∞	2	2	1	$\varepsilon = 1$
sens	–	1	1	1	1

Etape 2 :

$\mathbb{E}$	s	a	b	t
orig	–	s	a	b
ε	∞	1	1	$\varepsilon = 1$
sens	–	1	1	1



Marquage pile profondeur (DFS)

Etape 1 :

$\mathbb{E}$	s	a	b	c	t
orig	–	s	a	b	c
ε	∞	2	2	1	$\varepsilon = 1$
sens	–	1	1	1	1

Etape 2 :

$\mathbb{E}$	s	a	b	t
orig	–	s	a	b
ε	∞	1	1	$\varepsilon = 1$
sens	–	1	1	1

Etape 3 :

$\mathbb{E}$	s	c	b	a	t
orig	–	s	c	b	b
ε	∞	2	1	1	$\varepsilon = 1$
sens	–	1	-1	-1	1

on dépile ↑

Etape 4 :

$\mathbb{E}$	s	c
orig	–	s
ε	∞	1
sens	–	1

$\Rightarrow$ pile vide.

on dépile $\uparrow$ $\uparrow$

Le puits t n'est pas marqué $\Rightarrow$ pas de chaîne améliorante $\Rightarrow$ flot maximal

Etape 4 :

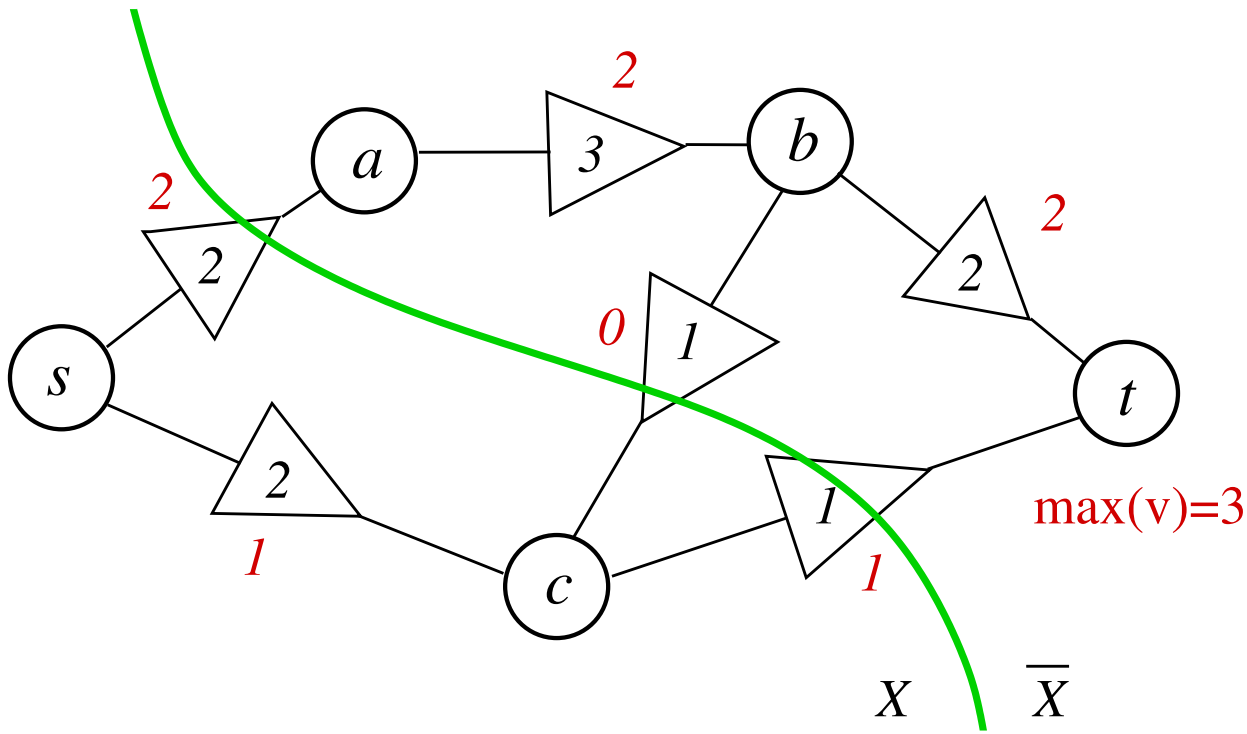
$\mathbb{E}$	s	c
orig	–	s
ε	∞	1
sens	–	1

$\Rightarrow$ pile vide.

on dépile $\uparrow$ $\uparrow$

Le puits t n'est pas marqué $\Rightarrow$ pas de chaîne améliorante $\Rightarrow$ flot maximal

Coupe minimale : $(X, \overline{X})$ avec $X = \{s, c\}$ et $\overline{X} = \{a, b, t\}$.
X est formé des sommets marqués à la dernière étape (pile vide).



5) Finitude et complexité

- Capacités à valeurs entières

  Pour des capacités à valeurs entières, l'algorithme de Ford-Fulkerson converge en un nombre fini d'opérations.

5) Finitude et complexité

- Capacités à valeurs entières

  Pour des capacités à valeurs entières, l'algorithme de Ford-Fulkerson converge en un nombre fini d'opérations.

  Pour un graphe avec n sommets et m arêtes :
  - $\mathcal{O}(m)$ opérations pour la recherche d'une chaîne améliorante et l'amélioration du flot.
  - La capacité d'une coupe est au plus en $\mathcal{O}(n \times C_{\max})$ où $C_{\max}$ est le maximum des capacités des arêtes. Dans le pire des cas, le flot augmente d'une seule unité à chaque fois. Il y a donc au plus $\mathcal{O}(nC_{\max})$ améliorations.

  $\Rightarrow \mathcal{O}(nmC_{\max})$ opérations pour l'algorithme de Ford-Fulkerson.

  Parfois la complexité est donnée en utilisant le flot maximum F^* (qu'on ne connait pas a priori) qui est nécessairement entier, ce qui nous donne une complexité en $\mathcal{O}(mF^*)$.

- Pour des capacités à valeurs réelles et un parcours en largeur (BFS), l'algorithme converge également. Cette variante est appelée algorithme d'Edmonds-Karp (1972).

Dans Ford-Fulkerson en parcours largeur (BFS), alias Edmonds-Karp, on choisit systématiquement une chaine améliorante de **plus court chemin** *de s à t, c'est-à-dire celle avec le moins d'arêtes possibles.*

Cet algorithme se termine toujours (même pour des capacités non entières), avec une complexité en $\mathcal{O}(nm^2)$ (indép. des capacités).

IV. Interlude : codage informatique d'un graphe

Si on considère des graphes *peu denses* (cas le plus fréquent), c'est à dire tels que *la plupart des sommets ne sont reliés (par une arête) qu'à une petite partie des autres sommets*, on utilise *des listes d'adjacence :* pour chaque sommet i, on dispose de la liste $V[i]$ de ses voisins (=sommets qui partagent une arête commune avec le sommet i).

- Si le graphe est orienté (E, Γ) pour chaque sommet $i \in E$, on dispose :
  - de la liste de ses successeurs $S[i]$ ($j \in S[i] \iff (i,j) \in \Gamma$)
  - et le plus souvent, de la liste $P[i]$ de ses prédécesseurs $j \in P[i] \iff (j,i) \in \Gamma$.

- Si de plus les arcs (=arêtes orientées) sont valués, il y a une autre liste associée $Val[i]$ ayant la même "structure" que $S[i]$: si j est le k ème élément de $S[i]$ ($j = S[i][k]$) la valuation de l'arête (i,j) est accessible par $Val[i][k]$.

IV. Interlude : codage informatique d'un graphe

Si $\#E = n$, **l'idée naturelle est de numéroter les noeuds/sommets** du graphe de 1 à n (ou de 0 à $n-1$), et d'utiliser un **tableau** T : $T[i]$ permettant d'accéder aux diverses listes (disons $T[i].S$, $T[i].P$ et $T[i].Val$) associées au sommet i.

Avantages :

- légèreté (faible occupation mémoire) ;
- accès à un sommet donné très rapide.

Inconvénients :

- ajouts et suppressions de sommets et d'arêtes difficiles ;
- répondre à une question du genre : *"Est ce que le noeud j est un successeur du noeud i ?"* demande de balayer la liste des successeurs de i (opération pas si rapide s'il y a suffisamment de successeurs).

IV. Interlude : codage informatique d'un graphe

Autre possibilité : utiliser **une table de hachage** (appelée dictionnaire en python). À chaque sommet est associé un identificateur unique[1]. Si i est un tel identificateur alors de même $T[i]$ permettra d'accéder aux diverses listes qui peuvent elles-même être des dictionnaires.

Avantages :

- ajouts et suppressions de sommets et d'arêtes faciles et rapides ;
- si la liste des successeurs est un dictionnaire, répondre à la question *"Est ce que le noeud j est un successeur du noeud i ?"* est une opération rapide (est ce que j est une clé du dictionnaire $T[i].succ$?).

Inconvénients :

- lourd en mémoire !
- l'accès aux caractéristiques d'un sommet est un peu moins rapide (il faut décoder la clé et un peu plus pour avoir la case mémoire correspondante).

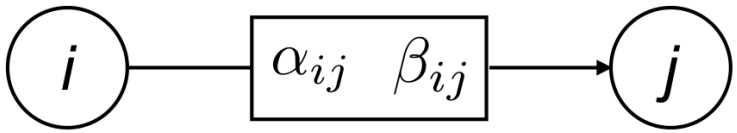
1. Souvent une chaîne de caractères ou tout objet utilisable comme clé par le langage, python accepte beaucoup de choses comme clés...

V. Flot maximal avec bornes inférieures et supérieures

1) Introduction

- Graphe valué par des capacités inférieures $\{\alpha_{ij}\}$ et supérieures $\{\beta_{ij}\}$: $G = (E, \Gamma, (\{\alpha_{ij}\}, \{\beta_{ij}\}))$

  On note :

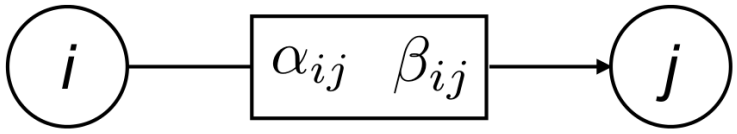


V. Flot maximal avec bornes inférieures et supérieures

1) Introduction

- Graphe valué par des capacités inférieures $\{\alpha_{ij}\}$ et supérieures $\{\beta_{ij}\}$: $G = (E, \Gamma, (\{\alpha_{ij}\}, \{\beta_{ij}\}))$

  On note :



- Problème de flot maximal généralisé :

$$\max_{f_{ij}, v} [F = v]$$

$$\begin{cases} \sum_{k \in S(s)} f_{sk} - v = 0 \quad \text{(source } s\text{)} \\ -\sum_{i \in P(j)} f_{ij} + \sum_{k \in S(j)} f_{jk} = 0, \quad \forall j \neq s, t \\ -\sum_{i \in P(t)} f_{it} + v = 0 \quad \text{(puits } t\text{)} \\ \alpha_{ij} \leq f_{ij} \leq \beta_{ij} \quad \text{pour toute arête } (i,j) \in \Gamma \end{cases}$$

2) Condition nécessaire d'existence d'un flot réalisable pour G.

Proposition

S'il existe un flot $\{f_{ij}\}$ réalisable sur G vérifiant $\alpha_{ij} \leq f_{ij} \leq \beta_{ij}$ pour tout $(i,j) \in \Gamma$, alors pour tout $j \neq s, t$:

$$\sum_{i \in P(j)} \alpha_{ij} \leq \sum_{k \in S(j)} \beta_{jk} \quad (1)$$

$$\sum_{k \in S(j)} \alpha_{jk} \leq \sum_{i \in P(j)} \beta_{ij} \quad (2)$$

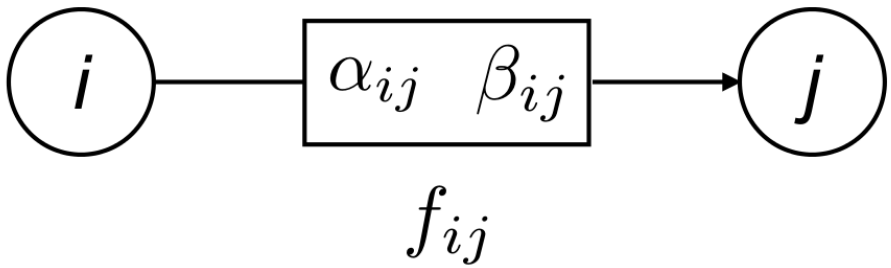
Démonstration. Soit $j \neq s, t$. On somme la relation $\alpha_{ij} \leq f_{ij} \leq \beta_{ij}$ sur $i \in P(j)$ puis sur $k \in S(j)$:

$$\sum_{i \in P(j)} \alpha_{ij} \leq \sum_{i \in P(j)} f_{ij} = \sum_{k \in S(j)} f_{jk} \leq \sum_{k \in S(j)} \beta_{jk}.$$

L'inégalité (2) se montre de la même façon. □

3) Adaptation de Ford-Fulkerson : flot maximal sur G.

On suppose qu'on dispose d'un flot initial réalisable sur G (cf. section suivante). On adapte l'algorithme de Ford-Fulkerson pour la recherche d'une chaine améliorante. On détermine ainsi un flot maximal sur G.



	arête directe	arête inverse
condition d'amélioration :	$f_{ij} < \beta_{ij}$	$f_{ij} > \alpha_{ij}$
amélioration possible :	$\varepsilon_1 = \beta_{ij} - f_{ij} > 0$	$\varepsilon_2 = f_{ij} - \alpha_{ij} > 0$

V. Flot maximal avec bornes inférieures et supérieures

4) Recherche d'un flot réalisable sur G : graphe auxiliaire G'.

Il reste à déterminer un flot réalisable initial f sur G. On ne peut plus prendre $f \equiv 0$ partout sur G !

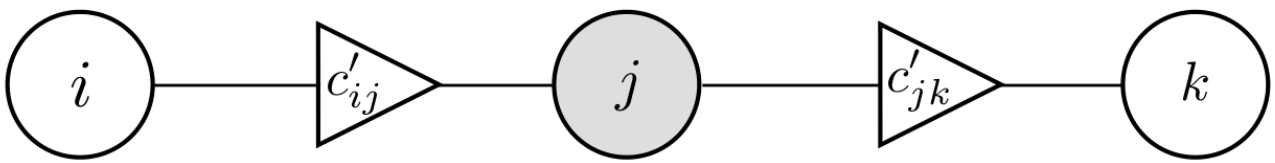
On se ramène au cas d'un flot positif par le changement de variable :

$$\boxed{f'_{ij} = f_{ij} - \alpha_{ij}}, \ \forall (i,j) \in \Gamma.$$

On a alors :

$$0 \leq f'_{ij} \leq c'_{ij} \text{ avec } \boxed{c'_{ij} = \beta_{ij} - \alpha_{ij}}$$

Rmq : telle quelle, l'application $f' : \Gamma \to \mathbb{R}$ **n'est généralement pas un flot**, on perd la conservation du flux :



$$f'_{ij} = f_{ij} - \alpha_{ij} \quad \neq \quad f'_{jk} = f_{jk} - \alpha_{jk}$$

$$(f_{ij} = f_{jk})$$

V. Flot maximal avec bornes inférieures et supérieures

Pour plusieurs prédécesseurs et successeurs, la **conservation** de f' au noeud j s'écrit :

$$\underbrace{\sum_{i \in P(j)} f'_{ij}}_{\textit{flux entrant en } j} \stackrel{?}{=} \underbrace{\sum_{k \in S(j)} f'_{jk}}_{\textit{flux sortant de } j}$$

$$\sum_{i \in P(j)} (f_{ij} - \alpha_{ij}) \stackrel{?}{=} \sum_{k \in S(j)} (f_{jk} - \alpha_{jk})$$

cette égalité sera fausse en général : on a la conservation du flux f en j ($\sum_{i \in P(j)} f_{ij} = \sum_{k \in S(j)} f_{jk}$) mais **sauf cas particulier** :

$$\sum_{i \in P(j)} \alpha_{ij} \neq \sum_{k \in S(j)} \alpha_{jk}$$

V. Flot maximal avec bornes inférieures et supérieures

Idée : on va introduire un **graphe valué auxiliaire** $G' = (E', \Gamma', c')$ avec deux sommets supplémentaires :

- s' (une source) reliée à tout j ($j \neq s, t$) par une capacité $c_{s'j}$;
- t' (un puits), tout j ($j \neq s, t$) est relié à t' par une capacité $c_{jt'}$

On prolonge l'application f' précédente : $f' : \Gamma' \to \mathbb{R}$ (avec $f'_{ij} = f_{ij} - \alpha_{ij}, \forall (i,j) \in \Gamma$).

Avec les nouvelles arêtes (s', j) et (j, t'), **la conservation de f' en j** s'écrit :

$$f'_{s'j} + \sum_{i \in P(j)} f'_{ij} = f'_{jt'} + \sum_{k \in S(j)} f'_{jk} \tag{3}$$

Comme $f'_{ij} = f_{ij} - \alpha_{ij}$, $\forall (i,j) \in \Gamma$, on obtient :

$$f'_{s'j} + \sum_{i \in P(j)} f_{ij} - \sum_{i \in P(j)} \alpha_{ij} = f'_{jt'} + \sum_{k \in S(j)} f_{jk} - \sum_{k \in S(j)} \alpha_{jk}$$

V. Flot maximal avec bornes inférieures et supérieures

Par conservation du flot f en j :

$$\sum_{i \in P(j)} f_{ij} = \sum_{k \in S(j)} f_{jk} \tag{4}$$

il faudrait donc que :

$$f'_{s'j} - \sum_{i \in P(j)} \alpha_{ij} = f'_{jt'} - \sum_{k \in S(j)} \alpha_{jk}$$

On choisit alors :

$$c'_{s'j} = \sum_{i \in P(j)} \alpha_{ij}, \quad \text{et } c'_{jt'} = \sum_{k \in S(j)} \alpha_{jk}$$

Ainsi, si le flot f' est **saturé** dans les arcs (s', j) et (j, t'), on obtient bien **l'équivalence** entre la conservation du flot f' en j (dans G') et la conservation du flot f en j (dans G) avec $f'_{ij} = f_{ij} - \alpha_{ij}$, $\forall (i,j) \in \Gamma$.

Rmq : sans arêtes supplémentaires un tel flot f' est maximum (il est clair que $(X = \{s'\}, \bar{X} = E' \setminus \{s'\})$ et $(X = E' \setminus \{t'\}, \bar{X} = \{t'\})$ sont deux coupures minimales).

Se dessine ainsi un calcul de flot max sur le graphe auxiliaire G' ! Et si ce flot max sature tous les arcs (s', j) et (j, t'), on aura gagné : pour $(i, j) \in \Gamma$, $f_{ij} := f'_{ij} + \alpha_{ij}$ définira bien un flot admissible (réalisable) sur le graphe G. Cependant, on a oublié un petit détail... Que fait-on avec la source s et le puits t du graphe G ?

Réponse : si on considère un flot réalisable f sur G de valeur v, le flot qui sort de la source s est égal au flot qui entre au puits t, soit :

$$v = \sum_{k \in S(s)} f_{sk} = \sum_{i \in P(t)} f_{it} \tag{5}$$

Dans le graphe auxiliaire G', on va :

- confondre ces deux sommets (ou ce qui est équivalent, les relier par un arc (t, s) de capacité infinie) ;
- créer un arc (s', t) de capacité $c'_{s't}$ et un arc (s, t') de capacité $c'_{st'}$.

V. Flot maximal avec bornes inférieures et supérieures

Le même type de calcul que précédemment (p. 57 et 58), montre qu'on obtient une **équivalence** entre l'équation (5) et la conservation de f' au "sommet" $st \in G'$ (via la transformation $f'_{ij} = f_{ij} - \alpha_{ij}$) avec :

$$c'_{s't} = \sum_{i \in P(t)} \alpha_{it}, \quad \text{et } c'_{st'} = \sum_{k \in S(s)} \alpha_{sk}$$

à **condition** que le flot f' soit **saturé** sur ces arcs.

Le calcul du flot maximum sur le graphe G' peut s'effectuer par un algorithme classique (pour flots positifs) comme Edmonds-Karp (ou autre) en partant du flot initial $f'_{ij} = 0, \forall (i,j) \in \Gamma'$ qui est réalisable (nous verrons plus loin une meilleure initialisation pour f'). Une fois le flot maximum obtenu, il faut vérifier la saturation du flot dans certaines arêtes...

V. Flot maximal avec bornes inférieures et supérieures

Résumé de la construction du graphe valué auxiliaire $G' = (E', \Gamma', c')$:

V. Flot maximal avec bornes inférieures et supérieures

Résumé de la construction du graphe valué auxiliaire $G' = (E', \Gamma', c')$:

- On ajoute deux sommets s' et t' : $E' = E \cup \{s', t'\}$

V. Flot maximal avec bornes inférieures et supérieures

Résumé de la construction du graphe valué auxiliaire $G' = (E', \Gamma', c')$:

- On ajoute deux sommets s' et t' : $E' = E \cup \{s', t'\}$
- On ajoute des arêtes reliant s' aux sommets $j \neq s$ et des arêtes reliant les sommets $j \neq t$ à t' :

$$\Gamma' = \Gamma \cup \{(s', j), \forall j \in E, j \neq s\} \cup \{(j, t'), \forall j \in E, j \neq t\}$$

- on relie t à s par un arc de capacité infinie (s et t sont confondus pour G').

V. Flot maximal avec bornes inférieures et supérieures

Résumé de la construction du graphe valué auxiliaire $G' = (E', \Gamma', c')$:

- On ajoute deux sommets s' et t' : $E' = E \cup \{s', t'\}$
- On ajoute des arêtes reliant s' aux sommets $j \neq s$ et des arêtes reliant les sommets $j \neq t$ à t' :

$$\Gamma' = \Gamma \cup \{(s', j), \forall j \in E, j \neq s\} \cup \{(j, t'), \forall j \in E, j \neq t\}$$

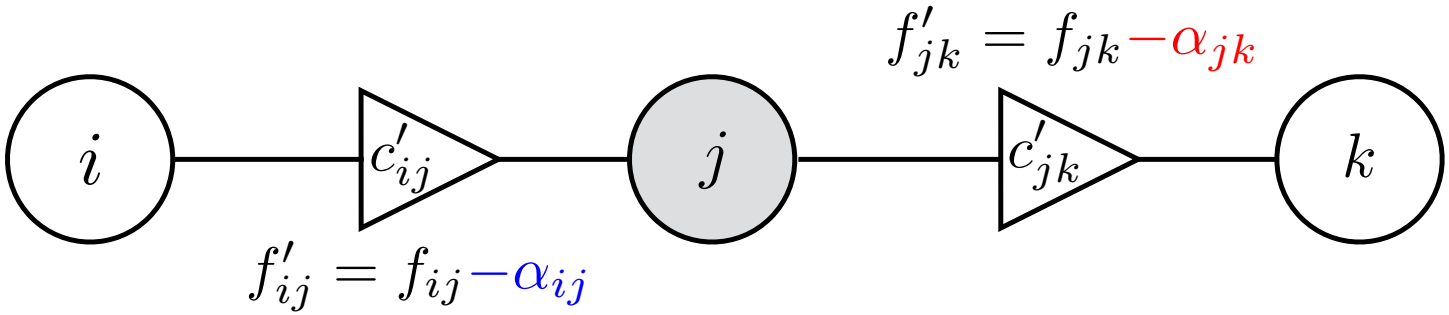
- on relie t à s par un arc de capacité infinie (s et t sont confondus pour G').
- Capacité c' :

$$c'_{ij} = \beta_{ij} - \alpha_{ij}, \quad \forall (i,j) \in \Gamma$$

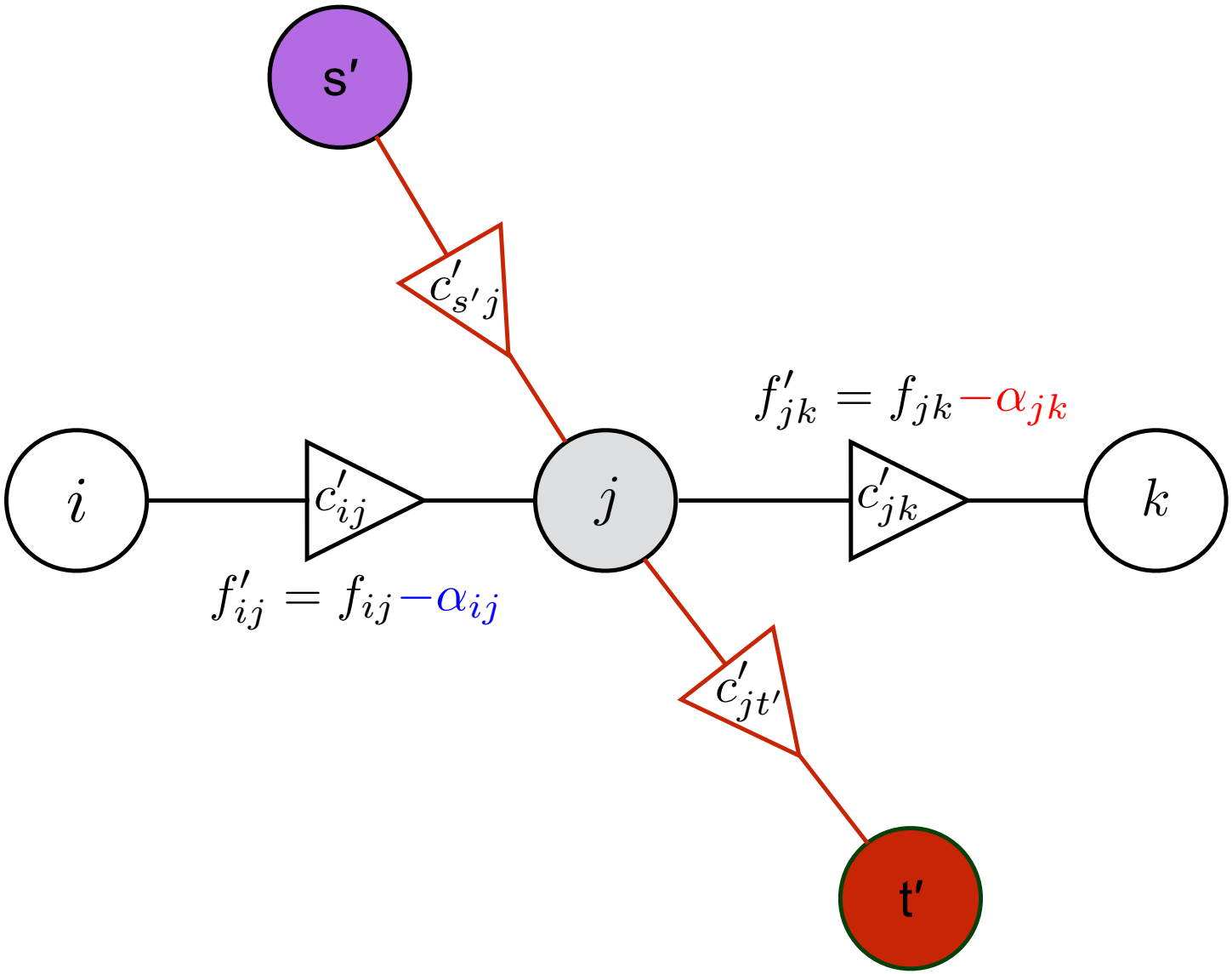
$$c'_{s'j} = \sum_{i \in P(j)} \alpha_{ij}, \quad \forall j \neq s$$

$$c'_{jt'} = \sum_{k \in S(j)} \alpha_{jk}, \quad \forall j \neq t$$

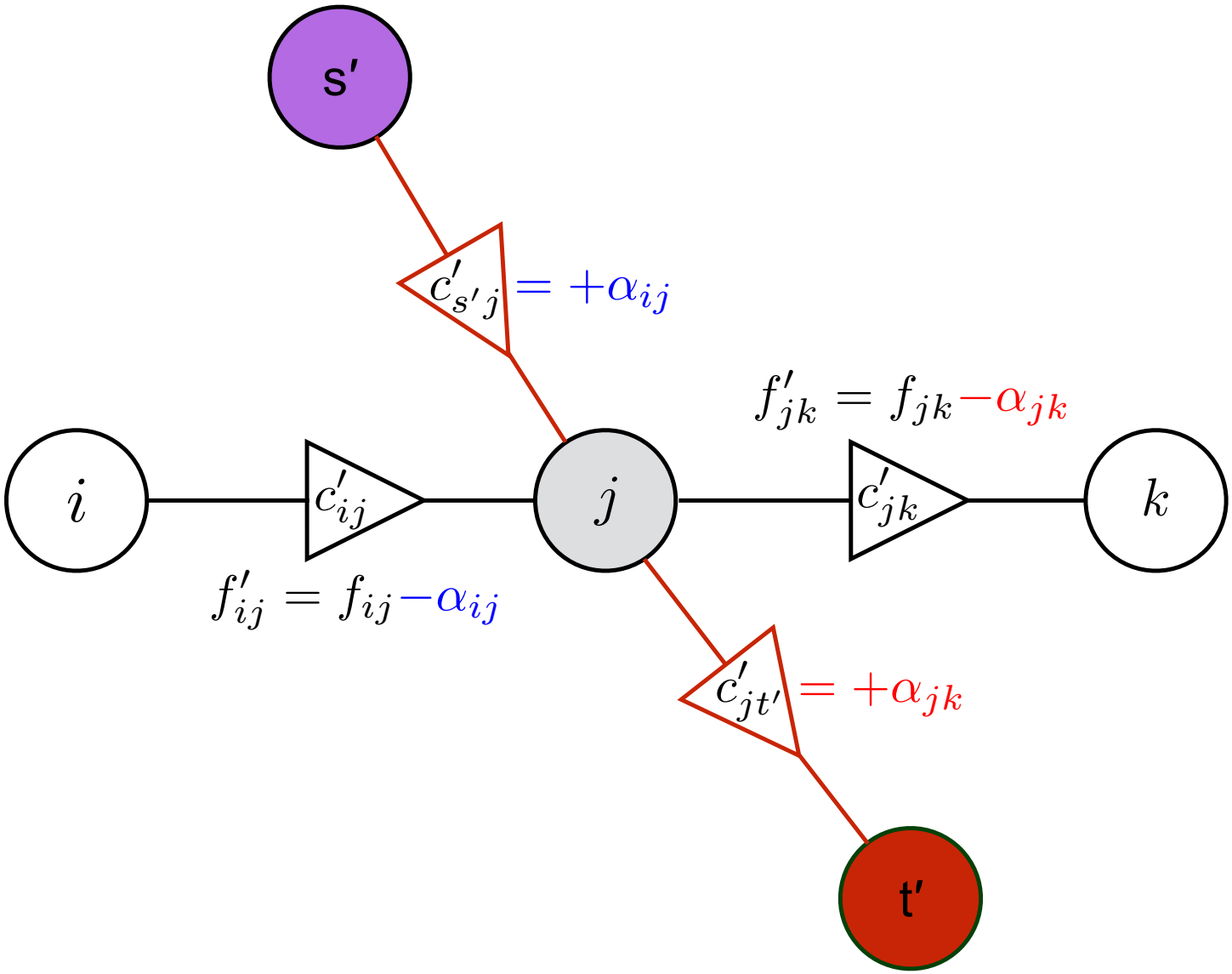
V. Flot maximal avec bornes inférieures et supérieures



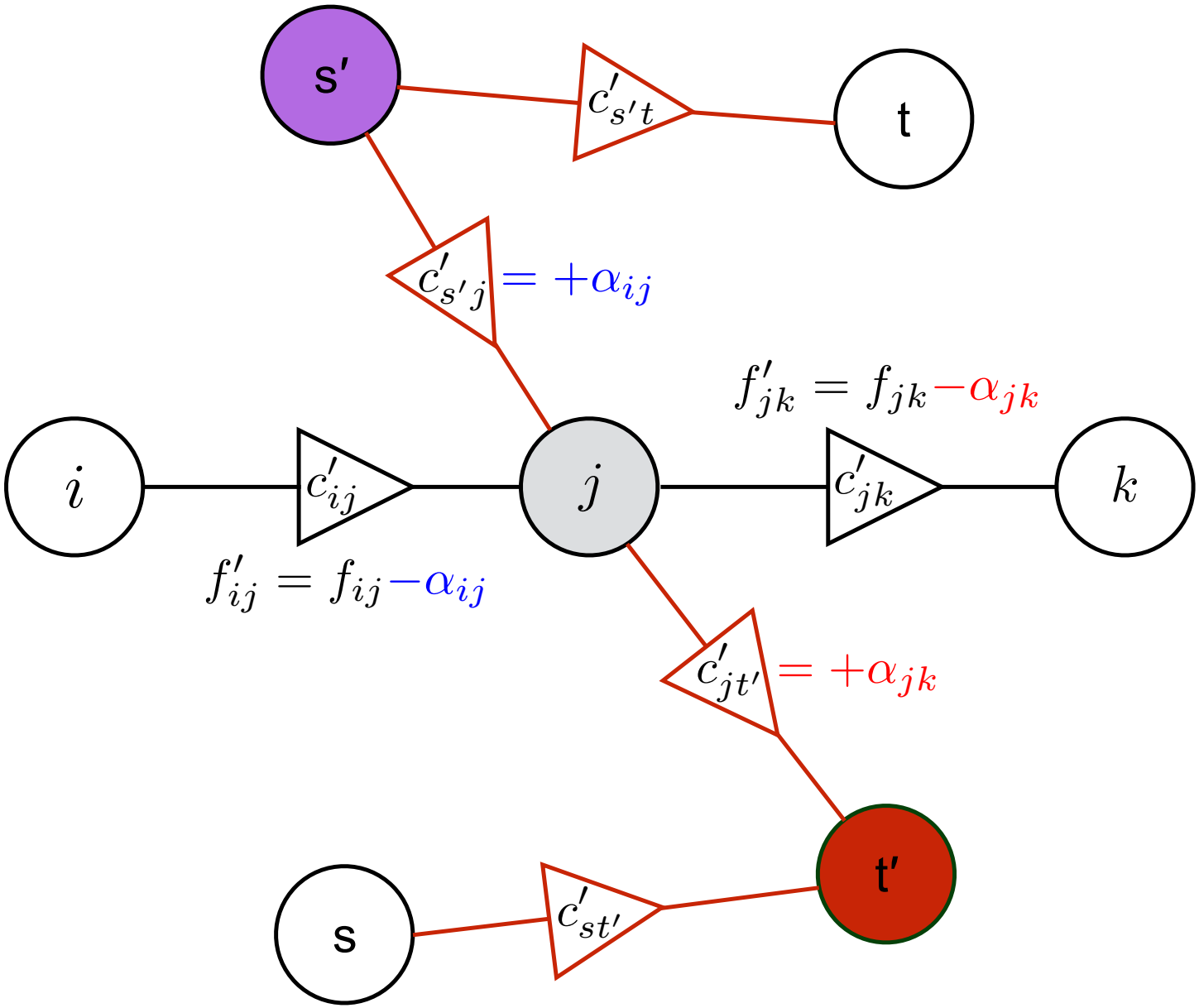
V. Flot maximal avec bornes inférieures et supérieures



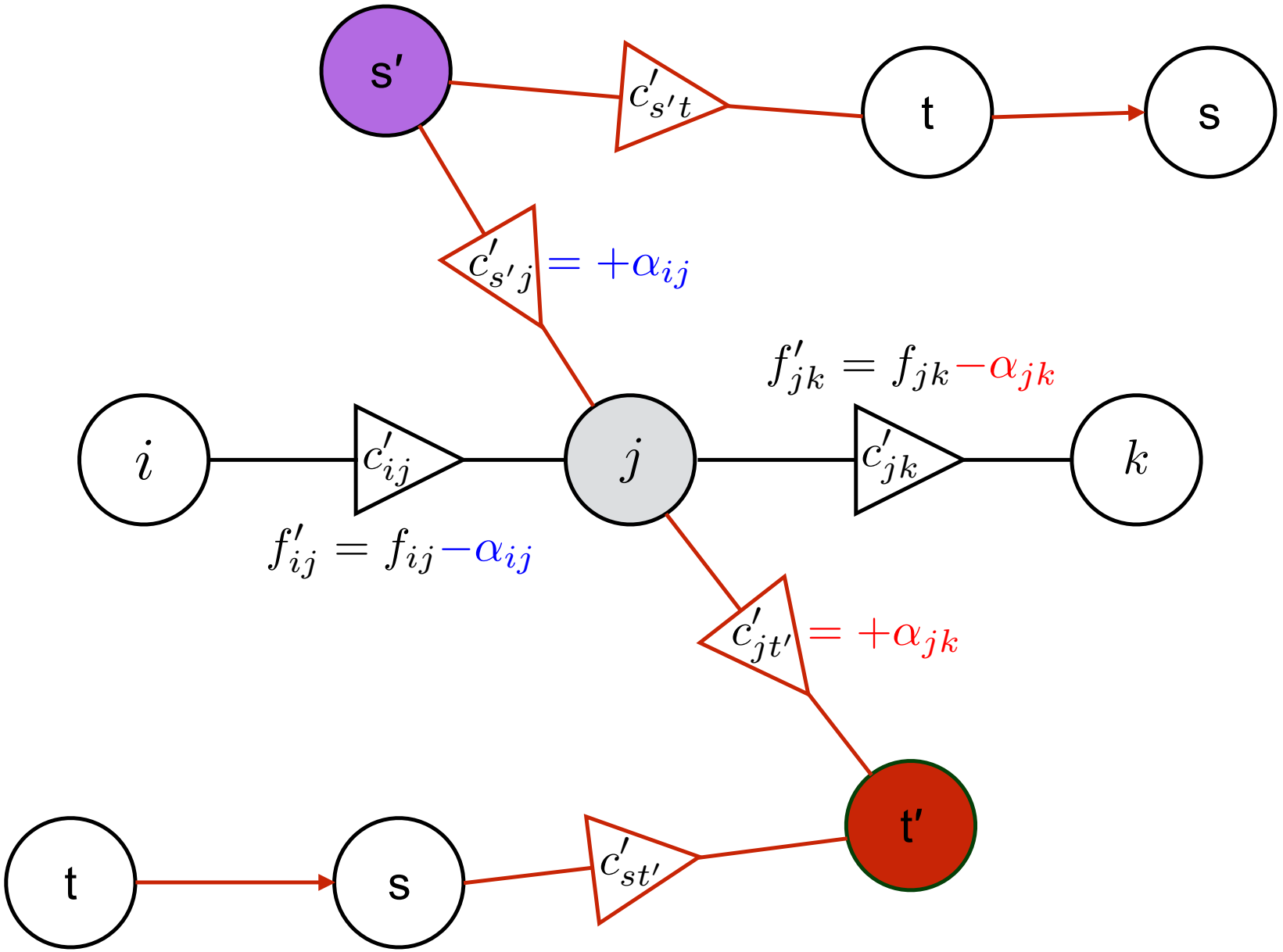
V. Flot maximal avec bornes inférieures et supérieures



V. Flot maximal avec bornes inférieures et supérieures



V. Flot maximal avec bornes inférieures et supérieures



V. Flot maximal avec bornes inférieures et supérieures

Soient les deux applications $f : \Gamma \to \mathbb{R}$ et $f' : \Gamma' \to \mathbb{R}$ telles que :

$$f'_{ij} = f_{ij} - \alpha_{ij}, \quad \forall (i,j) \in \Gamma \tag{6}$$

et :

$$\begin{aligned} f'_{s'j} = c'_{s'j} &:= \sum_{i \in P(j)} \alpha_{ij}, \quad \forall j \in E,\ j \neq s \\ f'_{jt'} = c'_{jt'} &:= \sum_{k \in S(j)} \alpha_{jk}, \quad \forall j \in E,\ j \neq t \end{aligned} \tag{7}$$

On a le :

Théorème (*CNS d'existence d'un flot réalisable sur G*)

l'application f est un flot réalisable sur $G = (E, \Gamma, c)$ $\iff$ l'application f' est un flot réalisable sur $G' = (E', \Gamma', c')$ (et en fait (7) implique que f' est un flot maximum sur G').

Démonstration abrégée

1. f flot réalisable sur $G \Rightarrow f'$ flot réalisable (et maximal) sur G' ?
   Si l'application f est un flot réalisable sur G, elle vérifie :
   - (i) $\alpha_{ij} \leq f_{ij} \leq \beta_{ij}, \forall (i,j) \in \Gamma$,
   - (ii) la conservation du flux en chaque sommet $j \neq s, t$ (i.e. (4)),
   - et (iii) l'égalité (5) (le flot sortant de s est égal au flot entrant en t).

   Par définition de l'application f', i.e. :
   - $f'_{ij} = f_{ij} - \alpha_{ij}$ pour $(i,j) \in \Gamma$
   - et f'_{ij} donné par (7) pour $(i,j) \in \Gamma' \setminus \Gamma$,

   f' vérifie les conditions de conservation en chaque sommet (cf calculs p.57...60) ainsi que $0 \leq f'_{i,j} \leq c'_{i,j}, \forall (i,j) \in \Gamma'$. Il s'agit donc d'un flot réalisable sur G' et par les conditions de saturation (7) c'est un flot maximal sur G'.
2. f' flot réalisable (et maximal) sur $G' \Rightarrow f$ réalisable sur G ?
   À faire.

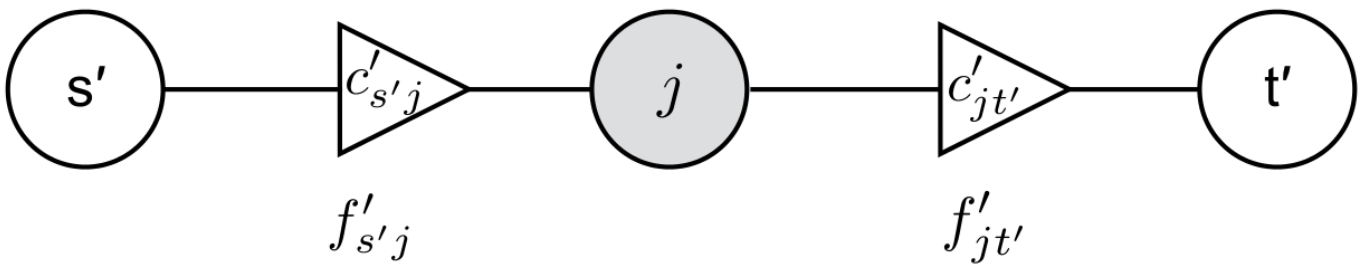
V. Flot maximal avec bornes inférieures et supérieures

Résumé : pour trouver un flot réalisable pour G, on détermine un flot maximal f' pour le graphe auxiliaire G'. Deux possibilités :

1. si f' vérifie les conditions de saturation (7) alors $f_{ij} = f'_{ij} + \alpha_{ij}, \forall (i,j) \in \Gamma$ est un flot réalisable pour G ;
2. sinon cela veut dire qu'il n'existe pas de flot réalisable sur G.

Flot réalisable initial sur le graphe auxiliaire G'.

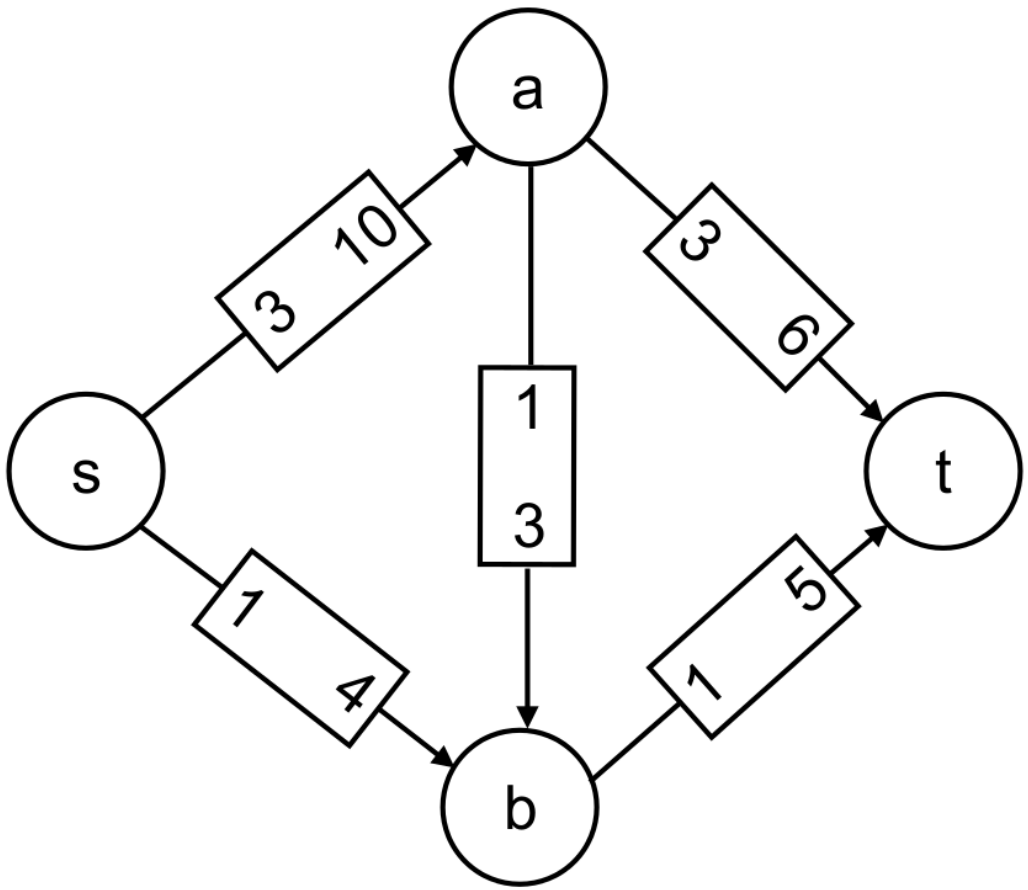
On peut choisir le flot nul $f' \equiv 0$ mais on peut aussi faire un peu mieux :



$$f'_{s'j} = f'_{jt'} = \min(c'_{s'j},\ c'_{jt'}) \quad \text{pour tout } j \in E$$
$$f'_{ij} = 0 \quad \text{pour tout } (i,j) \in \Gamma$$

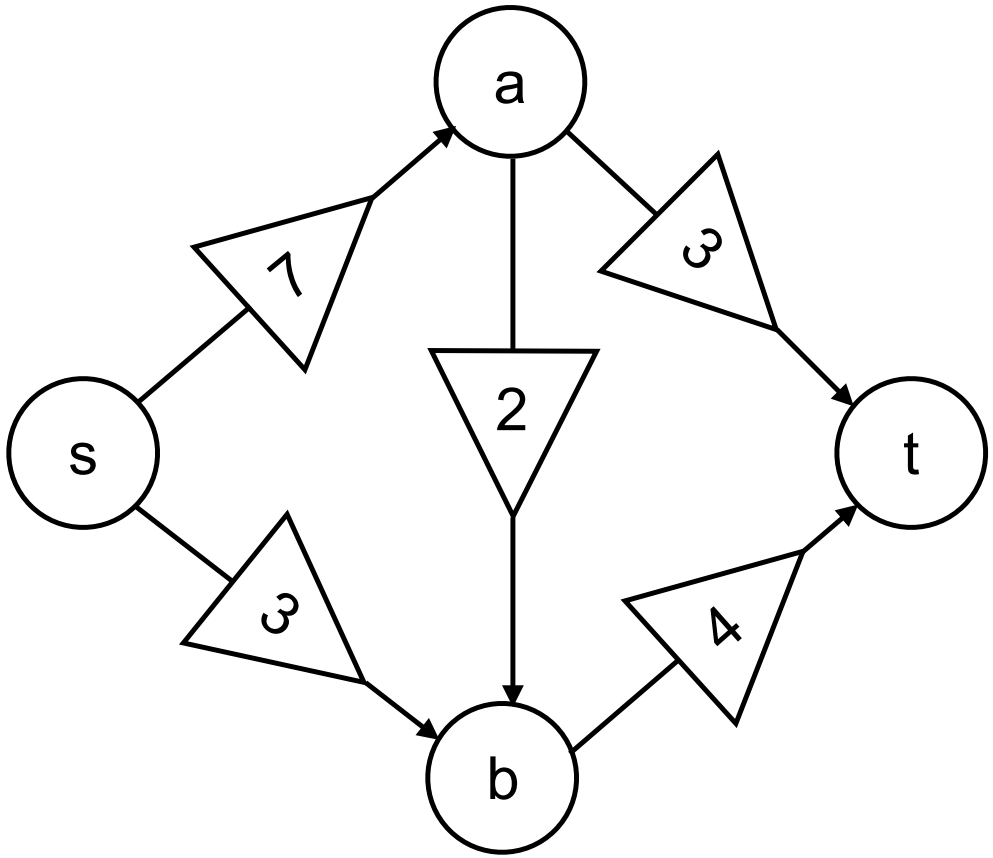
V. Flot maximal avec bornes inférieures et supérieures

Exemple de détermination d'un flot réalisable initial sur G.



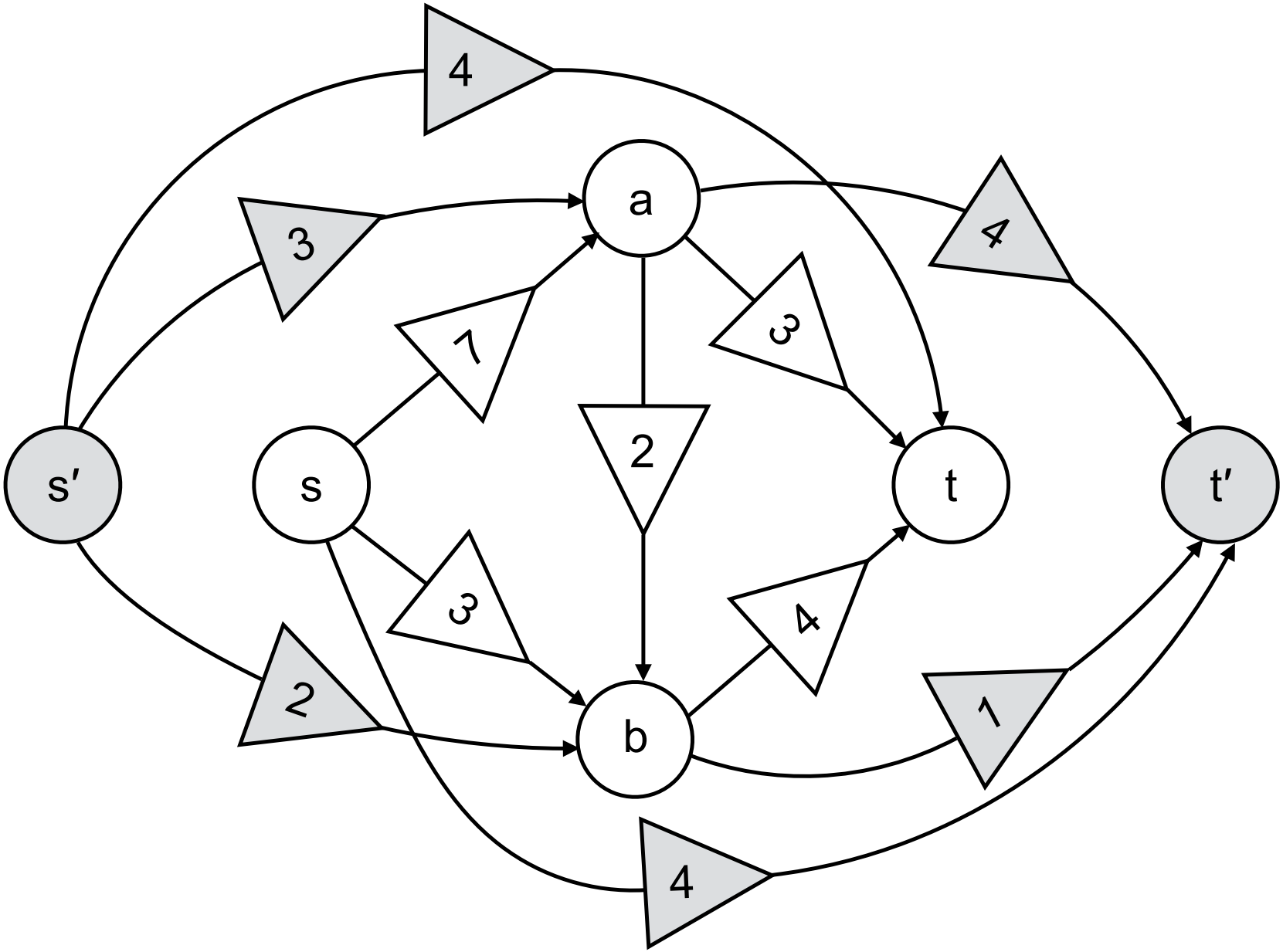
V. Flot maximal avec bornes inférieures et supérieures

a) Graphe auxiliaire G' :



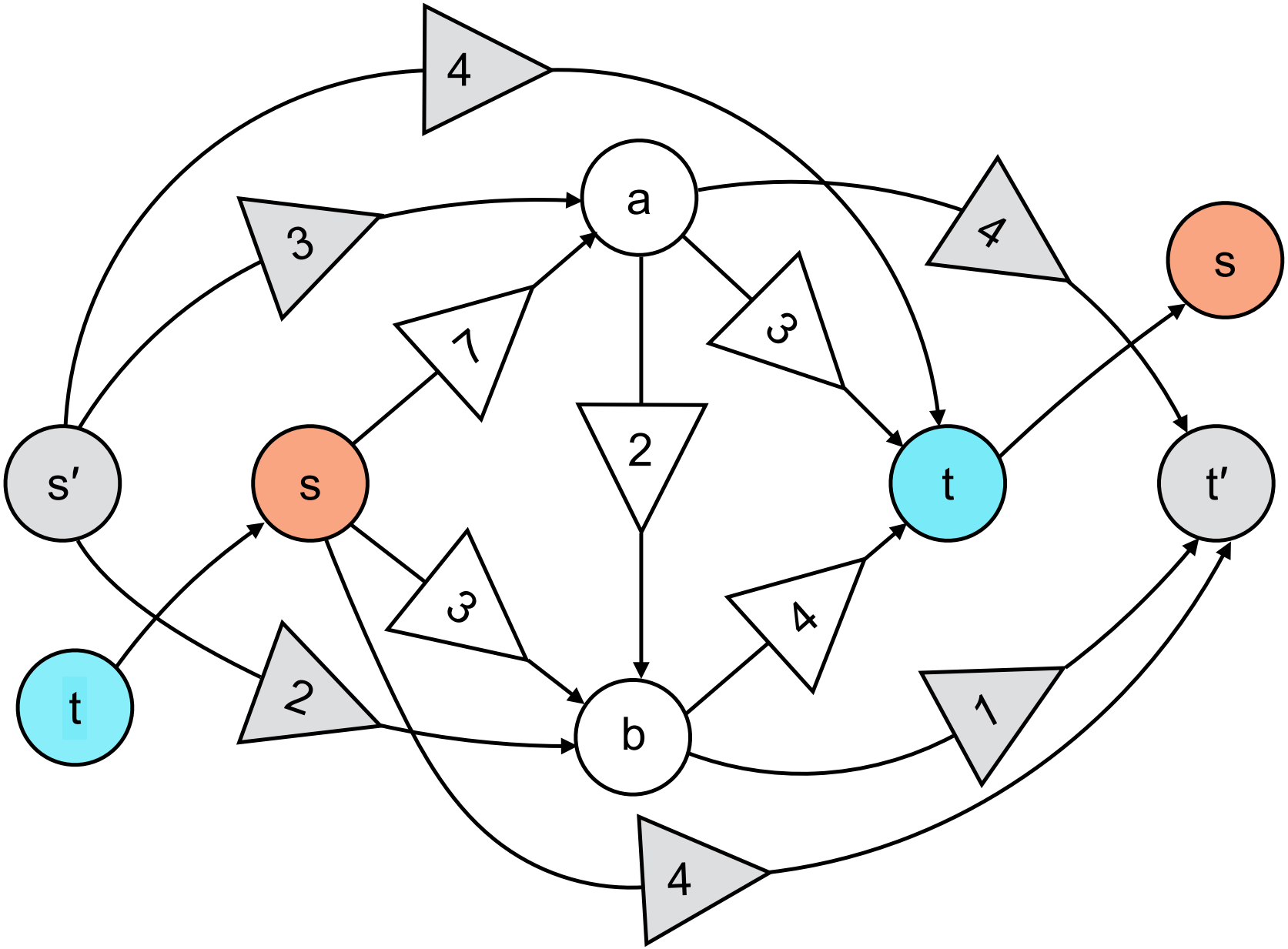
V. Flot maximal avec bornes inférieures et supérieures

a) Graphe auxiliaire G' :



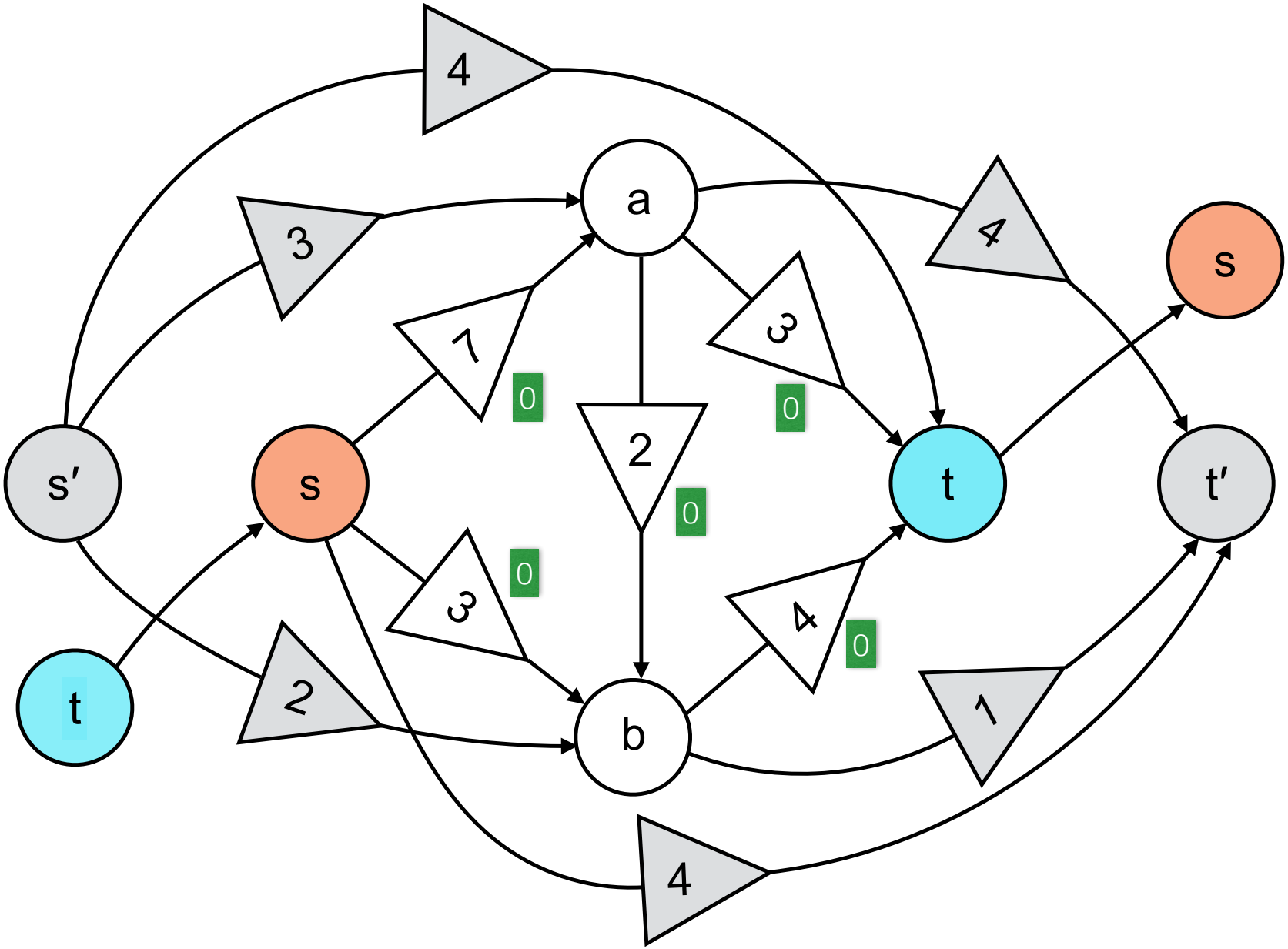
V. Flot maximal avec bornes inférieures et supérieures

a) Graphe auxiliaire G' :



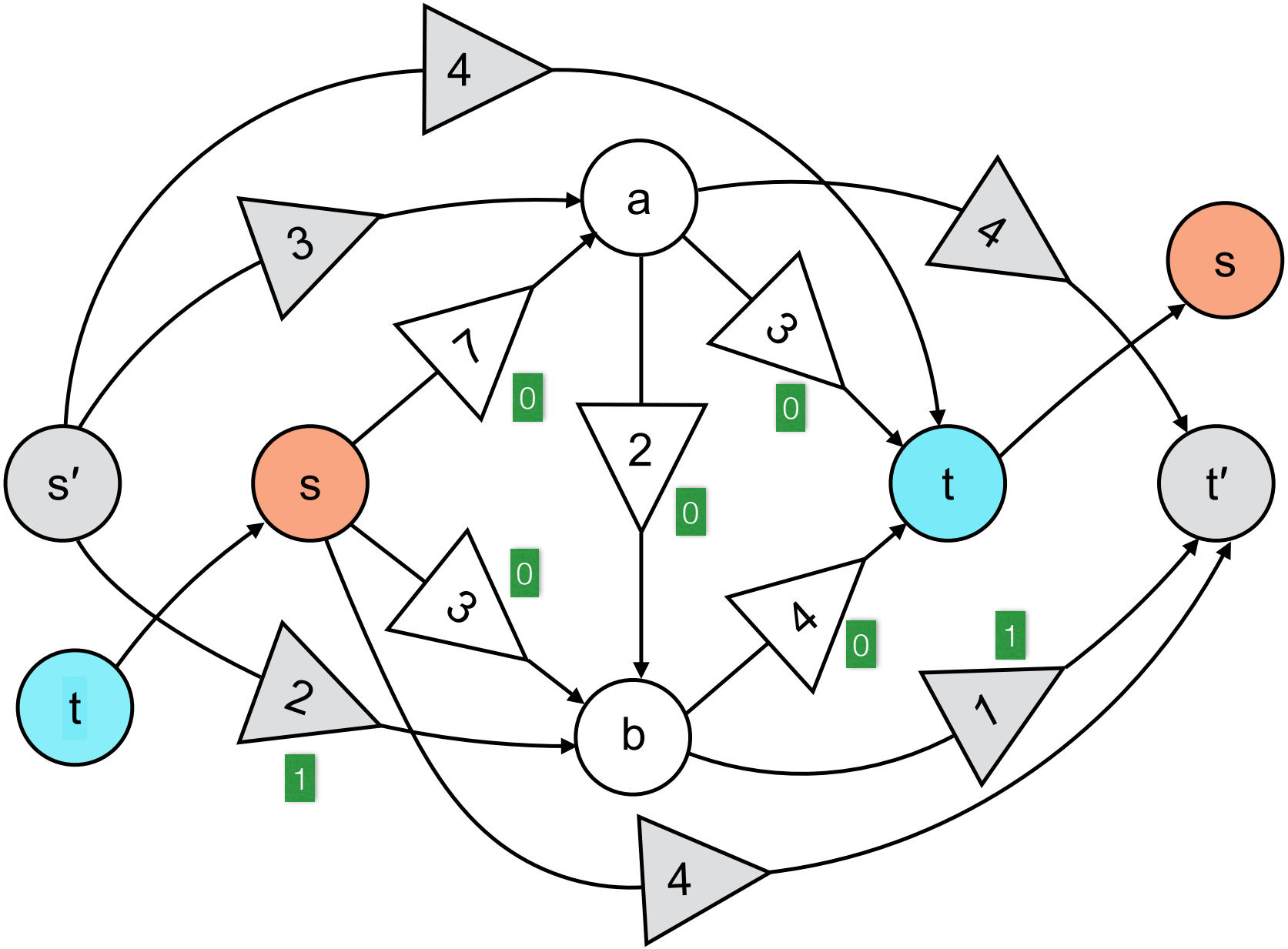
V. Flot maximal avec bornes inférieures et supérieures

b) Flot réalisable initial sur G' :



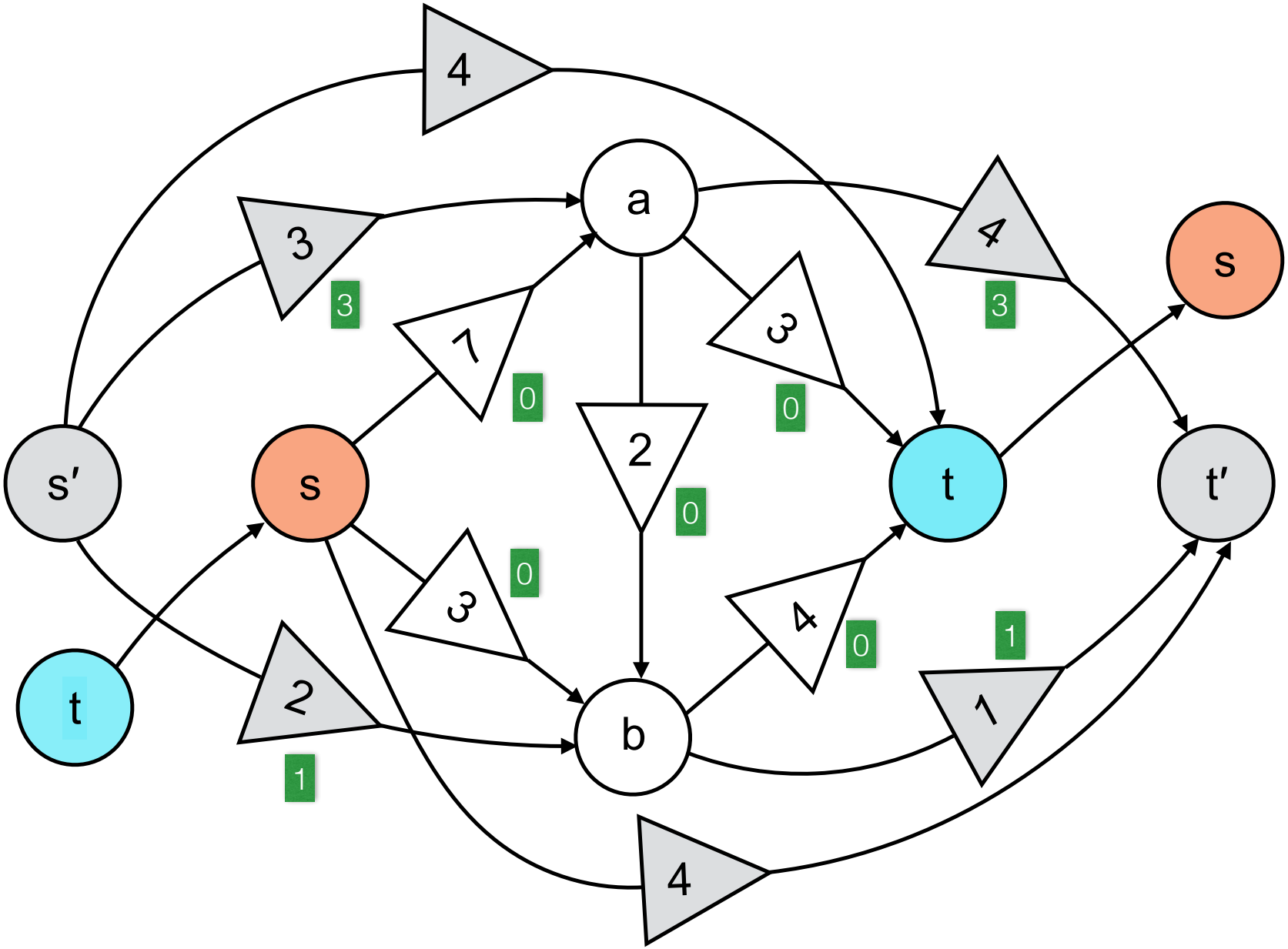
V. Flot maximal avec bornes inférieures et supérieures

b) Flot réalisable initial sur G' :



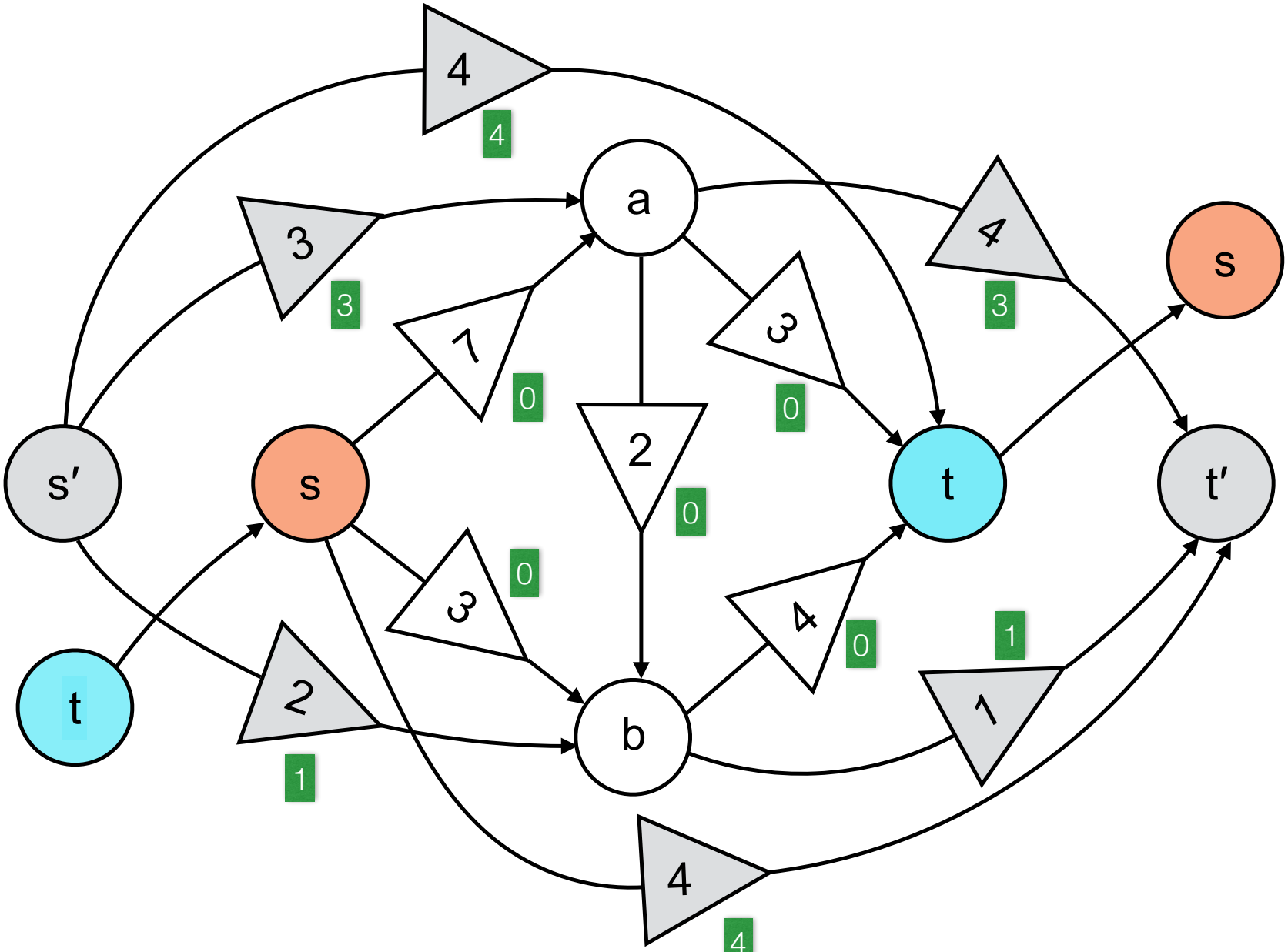
V. Flot maximal avec bornes inférieures et supérieures

b) Flot réalisable initial sur G' :



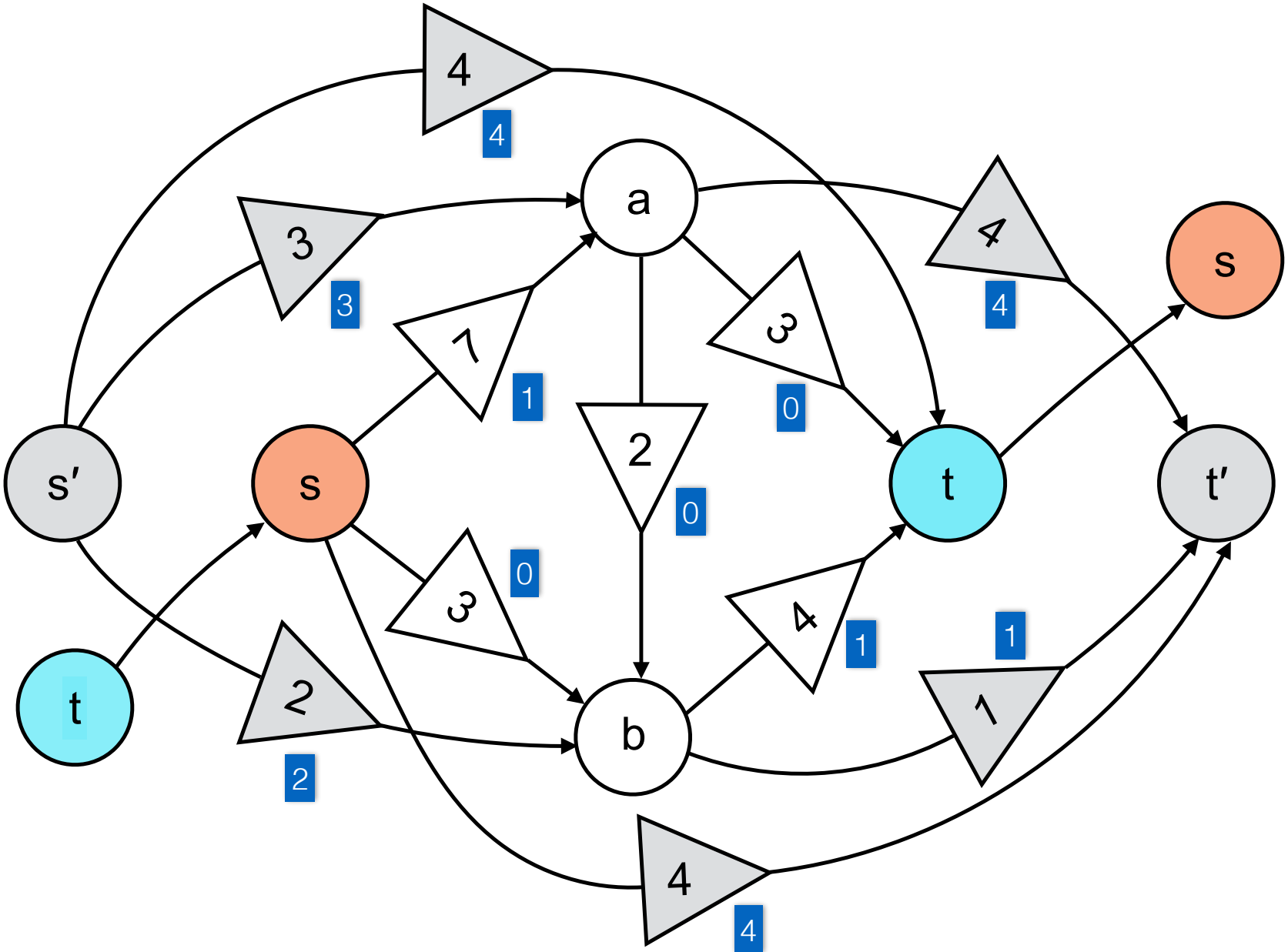
V. Flot maximal avec bornes inférieures et supérieures

b) Flot réalisable initial sur G' :



V. Flot maximal avec bornes inférieures et supérieures

c) Flot maximal sur G' (Ford-Fulkerson) :



V. Flot maximal avec bornes inférieures et supérieures

d) Flot réalisable sur G :