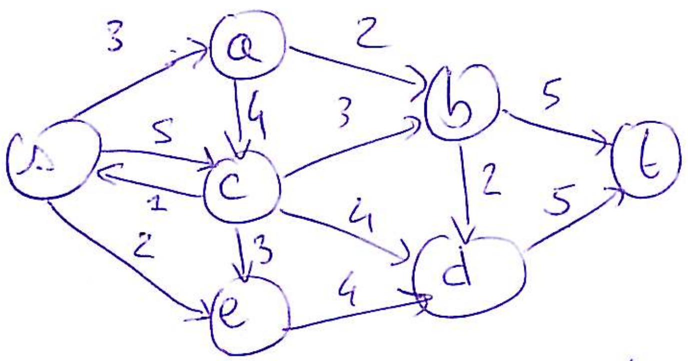
Problèmes de réseau modélisés avec des flot

Réseau = graphe orienté avec valuation positive
appelée "capacité" $c(e)$

ex. réseau électrique, distribution d'eau, Internet...

Deux sommets particuliers: s - source
t - puits

on veut transporter le plus de matière de s à t
en respectant les capacités du réseau.



- Problème du flot sans séparation

  a déjà été résolu on applique Floyd-Warshall avec

  le semi-anneau $\langle \mathbb{Z}^+ \cup \{\infty\}$ ou $\mathbb{R}^+ \cup \{\infty\}$, max, min, ∞, $0 \rangle$

- Problème du flot avec séparation

  Demande plus de travail → le sujet de cette partie.

Un flot f entre s et t est une application de ~~E~~ $V \times V$ dans $\mathbb{R}$ telle que

- $f(u,v) \leq \underbrace{c(u,v)}_{\text{capacité}}$ avec la convention que $c(u,v)=0$ si $(u,v) \notin E$.
- $f(u,v) = -f(v,u)$

(ceci implique en particulier que $f(u,v)=0$ si $(u,v) \notin E$ et $(v,u) \notin E$.)

- $\sum_{v \in V} f(u,v) = 0$ pour un u donné sauf s et t

(cela correspond à la loi des noeuds de Kirschhoff)

<u>La valeur d'un flot</u>, notée $|f|$ est

$|f| = \sum_{v \in V} f(s,v)$ (on peut montrer que $|f| = \sum_{u \in V} f(u,t)$)

<u>Demo</u>

$$\sum_{v \in V} f(s,v) = \sum_{v \in V} f(s,v) + \underbrace{\sum_{u \in V \setminus \{s,t\}} \sum_{v \in V} f(u,v)}_{=0}$$

$$= \sum_{u \in V \setminus \{t\}} \sum_{v \in V} f(u,v)$$

$$= \underbrace{\sum_{u \in V \setminus \{t\}} \sum_{v \in V \setminus \{t\}} f(u,v)}_{=0 \text{ par symetrie}} + \sum_{u \in V \setminus \{t\}} f(u,t)$$

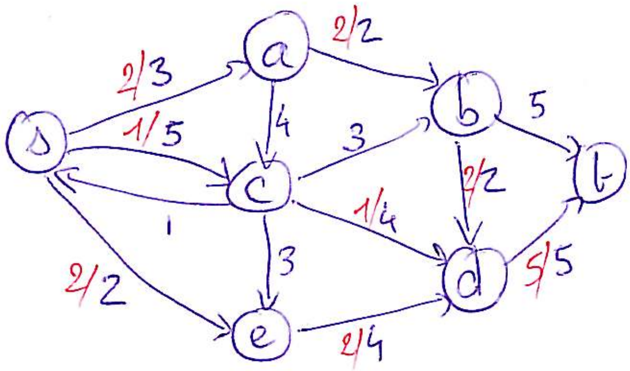
$$= \sum_{u \in V} f(u,t) \quad \text{car } f(t,t)=0$$

Intuitivement : tout ce qui quitte la source arrive au puits.
cette quantité est la valeur du flot.

Un exemple de flot

convention : $x \xrightarrow{a/b} y$ ~~désigne~~ représente $f(x,y)=a$
$c(x,y)=b$

on n'indiquera pas les flots ≤ 0



ici $|f| = 5$.

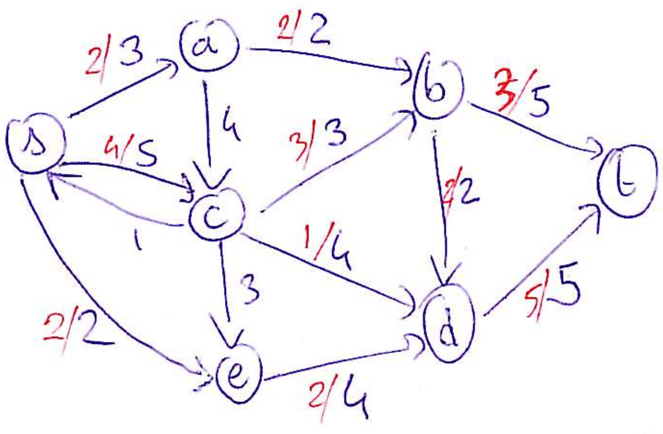
Problème du flot maximal

il s'agit de maximiser $|f|$.

Chemins améliorants

Une idée est de travailler itérativement sur le flot qu'on cherche à améliorer avec <u>un chemin améliorant</u>

Ici on trouve par exemple le chemin $s \to c \to b \to t$ qui peut améliorer le ~~chemin~~ flot de 3 :

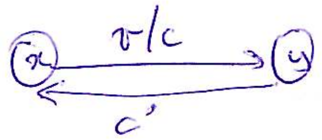


maintenant $|f| = 8$

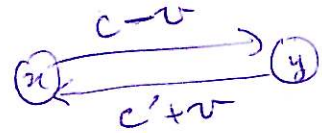
mais il n'existe plus de chemin améliorant.

Or on "voit" sur ce réseau qu'on devrait pouvoir obtenir $|f|=10$, si seulement on n'avait pas utilisé l'arc $b \to d$ ...
L'idée est d'imaginer un arc $d \xrightarrow{2} b$ entre d et b, qui permet en qq sorte de repousser ce qui s'y trouvait. Alors $s \to c \to d \to b \to t$ permet d'améliorer f de 1.

Graphe résiduel



est remplacé par

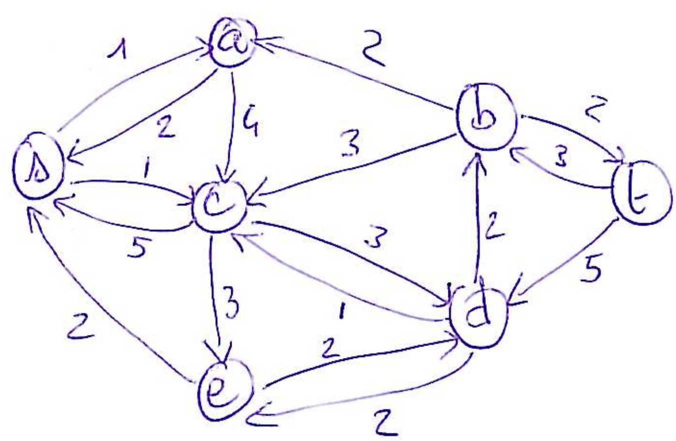


Autrement dit si $G=(V,E,c)$ on définit le graphe résiduel de G pour f par $G_f=(V,E',c_f)$

capacité

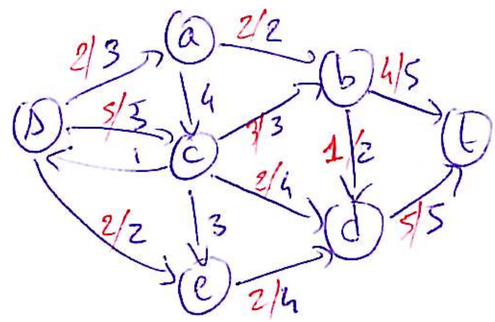
avec $c_f(u,v) = c(u,v) - f(u,v)$

et $E' = \{e \in V \times V \mid c_f(e) > 0\}$

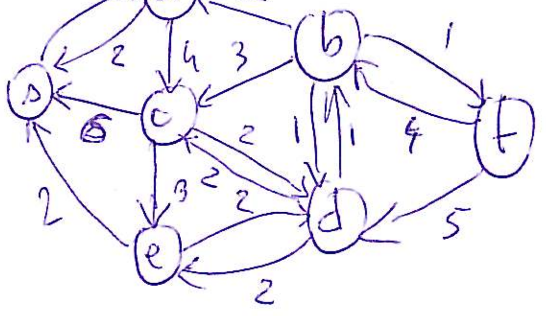


on trouve ici un chemin améliorant

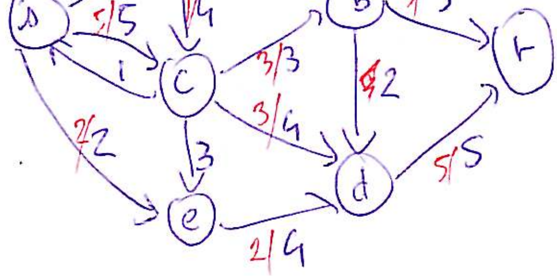
$s \xrightarrow{1} c \xrightarrow{3} d \xrightarrow{3} b \xrightarrow{2} t$



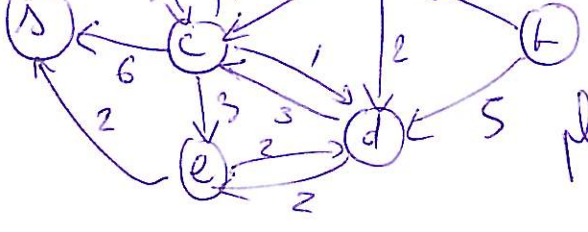
$|f| = 9$



ici $s \xrightarrow{1} a \xrightarrow{4} c \xrightarrow{2} d \xrightarrow{1} b \xrightarrow{1} t$



$|f| = 10$

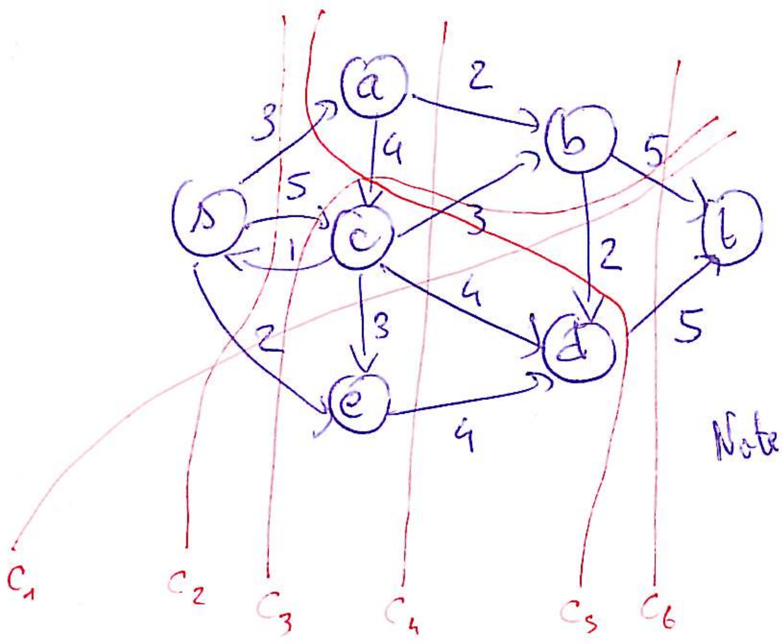


plus de chemin améliorant.

Coupe

- Une coupe $C = (S, T)$ est une partition de $V = S \uplus T$ telle que $s \in S$ et $t \in T$.
- La capacité d'une coupe $c(S,T) = \sum_{\substack{u \in S \\ v \in T}} c(u,v)$

Sur notre exemple



$c(C_1) = 16$
$c(C_2) = 10$
$c(C_3) = 17$
$c(C_4) = 13$
$c(C_5) = 11$
$c(C_6) = 10$
$\vdots$

Note : $c(C) \geq |f|$ quel que soit la coupe et le flot.

Théorème flot-maximale/coupe-minimal

- Si f est un flot de $G = (V, E, c)$ pour s, t alors les trois conditions suivantes sont équiv. :

1) f est un flot maximal
2) G_f (reseau résiduel) ne contient aucun chemin améliorant
3) $|f| = c(S,T)$ pour une coupe (S,T) de G.

Note - la coupe en question est forcément une coupe minimale

Méthode de Ford - Fulkerson pour calcul de flot maximal

Entrée : $G = (V, E, c, s, t)$
Sortie : un flot f de s à t tel que $|f|$ soit maximal.

~~$f(u,v) \leftarrow 0$~~

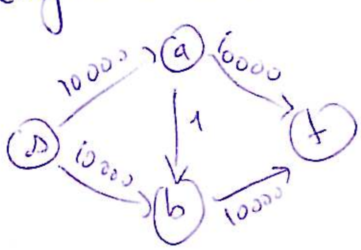
$\forall (u,v) \in V^2$, $f(u,v) \leftarrow 0$
tant qu'il existe dans G_f un chemin p de s à t
notons $c_f(p) = \min \{ c_f(u,v) \mid (u,v) \in p \}$ la capacité de p
pour chaque arc $(u,v) \in p$
$f(u,v) \leftarrow f(u,v) + c_f(p)$
$f(v,u) \leftarrow f(v,u) - c_f(p)$

Analyse

- trouver un chemin de s à t avec un parcours en profondeur se fait en $O(|E|)$ avec parcours en profondeur (par ex).
- si les capacités sont entières chaque chemin améliorant ~~peut~~ va augmenter $|f|$ au moins de 1.

L'algo tourne donc en $O(|E| \cdot |f^*|)$ si $|f^*|$ est la valeur du flot max.

ex de cas où ~~tous~~ les chemin améliorant ajoutent toujours 1 :



si on augmente avec les chemins

$s \to a \to b \to t$
$s \to b \to a \to t$
$s \to a \to b \to t$
$s \to b \to a \to t$
⋮
$\times 5000$

<u>Attention</u> si les capacités ne sont pas entières, l'algorithme ne termine pas forcément.

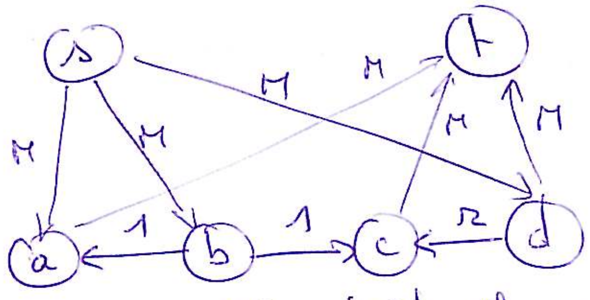
Exemple :

posons $M \geq 2$

$$r = \frac{\sqrt{5}-1}{2} \simeq 0{,}618$$

$$r^2 = 1 - r \simeq 0{,}382$$

($r = \frac{1}{\varphi}$ nb d'or)



chemin améliorant	flot envoyé	capacités résiduelles		
		1	1	r
s b c t	1	$1 = r^0$	0	r
s d c b a t	r	$1 - r = r^2$	r	0
s b c d t	r	r^2	0	r
s d c b a t	r^2	0	r^2	$r - r^2 = r^3$
s a b c	r^2	r^2	0	r^3

ces 4 chemins améliorant permettent de passer de capacités de la forme $(r^{2n}, 0, r^{2n+1})$ à des capacités de la forme $(r^{2(n+1)}, 0, r^{2(n+1)+1})$ en envoyant les flots : $r^{2n+1}, r^{2n+1}, r^{2(n+1)}, r^{2(n+1)}$.

Le flot total envoyé ~~est donc~~ converge vers

$$1 + 2\sum_{i=1}^{\infty} r^i = -1 + 2\sum_{i=0}^{\infty} r^i = -1 + 2\frac{1}{1-r}$$

avec $\frac{1}{1-r} = \frac{1-r+r}{1-r} = 1 + \frac{r}{1-r} = 1 + \frac{r}{r^2} = 1 + \frac{1}{r} = 2 + r$

donc $1 + 2\sum_{i=1}^{\infty} r^i = 3 + 2r \simeq 4.236$

Or le flot maximal ~~converge~~ est $2M+1$, il n'est pas atteint lorsque les chemins améliorants choisis sont ceux-ci.

L'algo de Ford-Fulkerson peut être amélioré pour

1) ne pas dépendre de $|f^*|$

2) terminer toujours

il suffit de chercher le chemin améliorant avec un parcours en largeur (pour prendre le plus court chemin (en comptant les arcs)). Cette modif est appelée algo d'<u>Edmonds-Karp</u>.

<u>Intuition derrière la complexité</u>

- la taille du plus court chemin de s à t dans G_f augmente (au sens large) à chaque itération
- chaque chemin peut être trouvé en $O(|E|)$
- à chaque itération l'un des $|E|$ arcs ~~devient~~ est saturé pour les itérations qui trouvent des chemins améliorant de la même taille, le m arc ne peut pas être à nouveau saturé. Par contre il peut l'être à nouveau plus tard pour d'autres longueurs de chemins.

→ il y a donc au plus $O(|E|.|V|)$ itérations.

⇒ complexité $O(|E|^2.|V|)$

Algo de Dinic

=Dinitz en vrai

À partir du graphe résiduel $G_f = (V, E_f, c_f, s, t)$ on peut construire un graphe en couches (qui contient tous les ~~plus~~ chemins de longueur la plus courte).

$$G_L = (V, E_L, c_f', s, t) \text{ où}$$

$$E_L = \{ (u,v) \in E \mid dist(s,v) = dist(s,u) + 1 \}$$

$$c_f' = \begin{cases} c_f(u,v) & \text{si } (u,v) \in E_L \\ 0 & \text{sinon.} \end{cases}$$

un flot de blocage est un flot f tel que le graphe $G' = (V, E_L')$ avec $E_L' = \{ (u,v) \mid f(u,v) < c_f'(u,v) \}$ ne contient pas de chemin de s à t.

Algo pour $G = (V, E, c, s, t)$

1. $\forall (u,v) \in V^2$, $f(u,v) = 0$
2. ~~construire~~ répéter
   ~~a. construire G_f~~
   a. construire G_L à partir de G_f
   b. si $dist(s,t) = \infty$ retourner f.
   c. trouver un flot de blocage f' dans G_L
   d. $\forall (u,v)$ $f(u,v) \leftarrow f(u,v) += f'(u,v)$.

Analyse

- à chaque itération la distance de s à t dans G_L augmente au moins d'1.
  $\rightarrow$ il y a donc $O(|V|)$ itérations.

- G_L se construit en $O(|E|)$
  - le calcul d'un flot bloquant peut se faire avec des parcours en profondeur successifs (dans G_L qui est acyclique) :
    - chaque DFS a une profondeur d'au plus $O(|V|)$
    - chaque DFS supprime au moins un arc (saturé) donc le nombre de DFS est au plus $O(|E|)$

  $\Rightarrow$ un flot bloquant se calcule en $O(|V|.|E|)$.

complexité totale : $O(|V|) \times (O(|E|) + O(|V|.|E|))$

$= O(|V|^2.|E|)$.

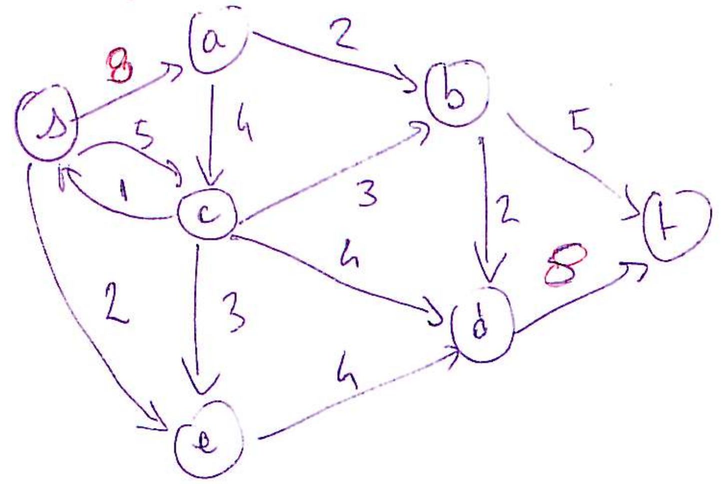
Cas particuliers

- capacités unitaires : calcul de flot bloquant en $O(|E|)$ au lieu de $O(|V|.|E|)$ car chaque arc est supprimé/saturé après avoir participé à un chemin.

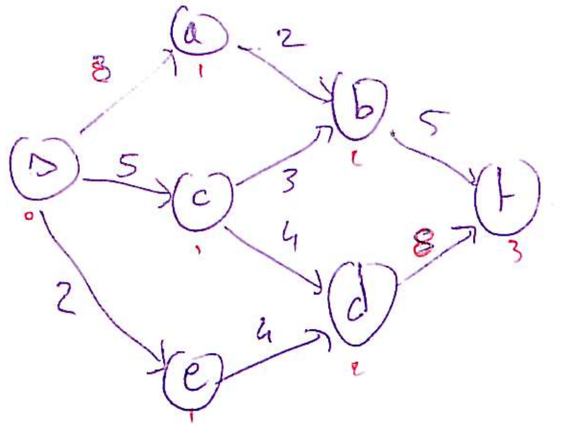
  $\Rightarrow O(|V|.|E|)$ pour l'algo complet.

exemple Dinic

$G = G_f$.



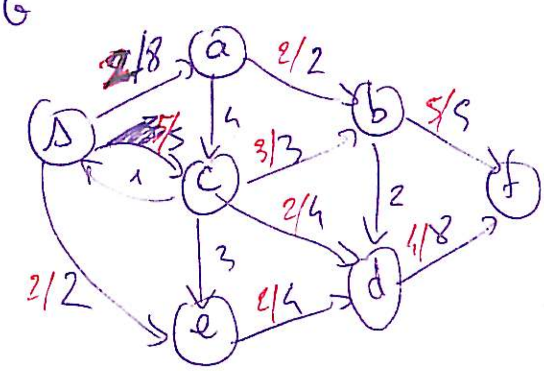
G_L graphe en couches



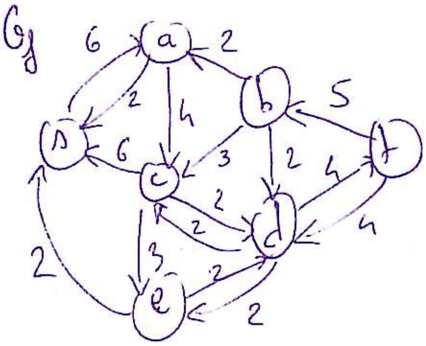
$s \to a \to b \to t$ flot 2
$s \to c \to b \to t$ flot 3
$s \to c \to d \to t$ flot 2
$s \to e \to d \to t$ flot 2

9

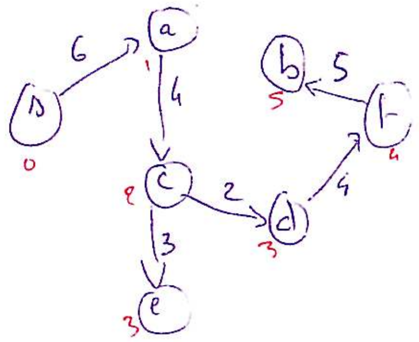
G



G_f

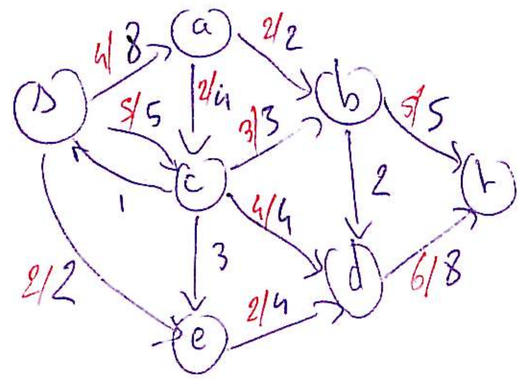


G_L

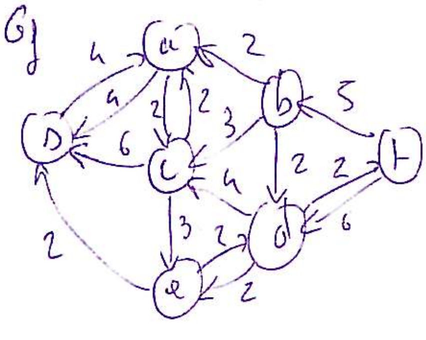


$s \to a \to c \to d \to t$ 2

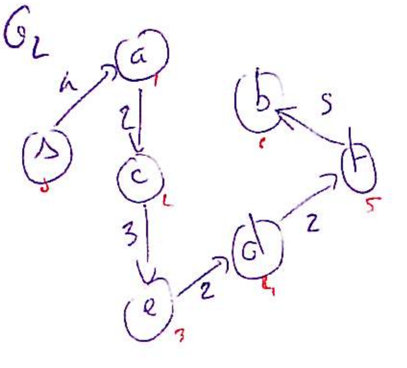
G



G_f

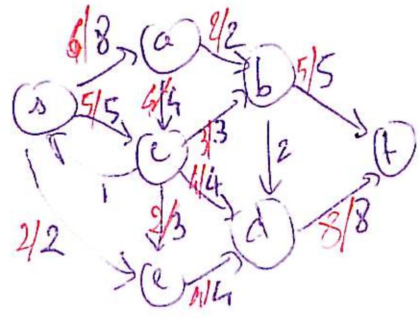


G_L

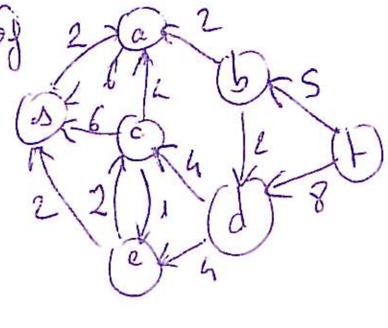


$s \to a \to c \to e \to d \to t$ 2

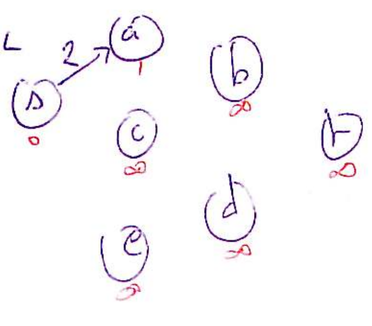
G



G_f



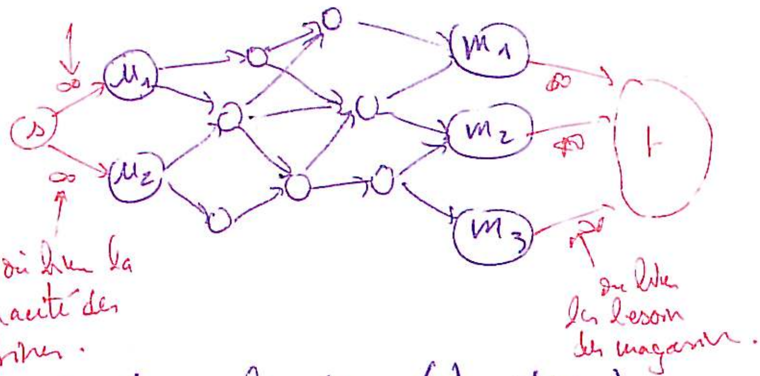
G_L



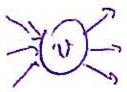
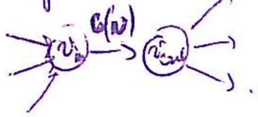
Application...

flots multi-sources/puits

p.ex. 2 usines, 3 magasins

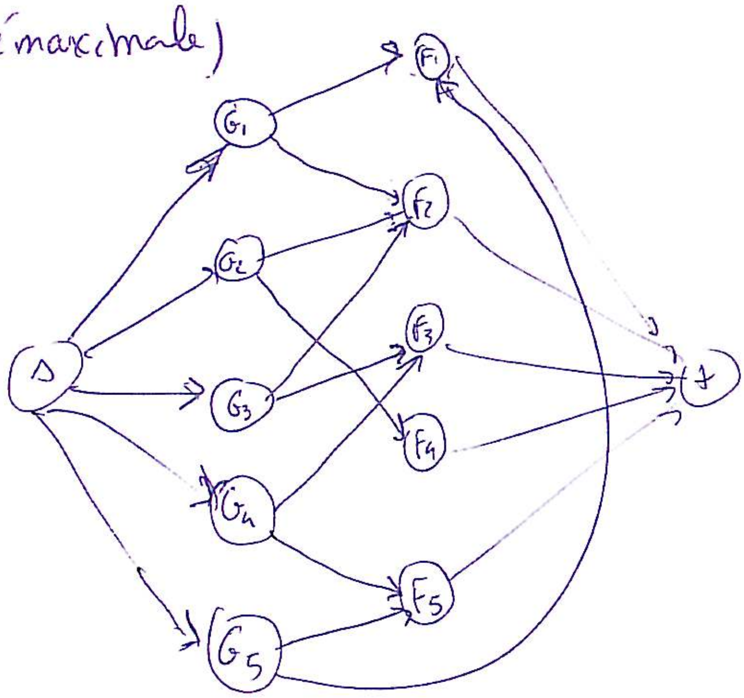


si un noeud a une contrainte (pas plus de $c(v)$ matière)

alors  est éclaté en 

matching

(de cardinalité maximale)



nombre de chemins ~~disjoints~~ arc-disjoints de s à t

→ ~~max~~ flot max sur un reseau de capacité unique.

Nombre de chemin sommets-disjoint

→ idem, mais en coupant les sommets.