TRANSFERT, RESEAUX, GRAPHES
Flots et coupes

Marie-Christine Costa

ENSTA Paris-Tech

TIPE 2014

Conférence auprès des professerus de classes préparatoires aux concours d'accès aux grandes écoles.

Initiation aux graphes et à la recherche opérationnelle.

Présentation mise à disposition pour utilisation éventuelle.

TRANSFERT, RESEAUX, GRAPHES

1. Graphes et réseaux de transport
2. Le problème du flot maximal
3. Un algorithme simple (Ford-Fulkerson)
4. Flot max et coupe min (dualité)
5. Des modèles mathématiques:programmes linéaires duaux (matrices TU)
6. Application: maximisation d’une fonction pseudobooléenne négative-positive
7. Pistes d'approfondissement
8. Bibliographie

1. Graphes et réseaux de transport

Qu'est-ce qu'un graphe ?

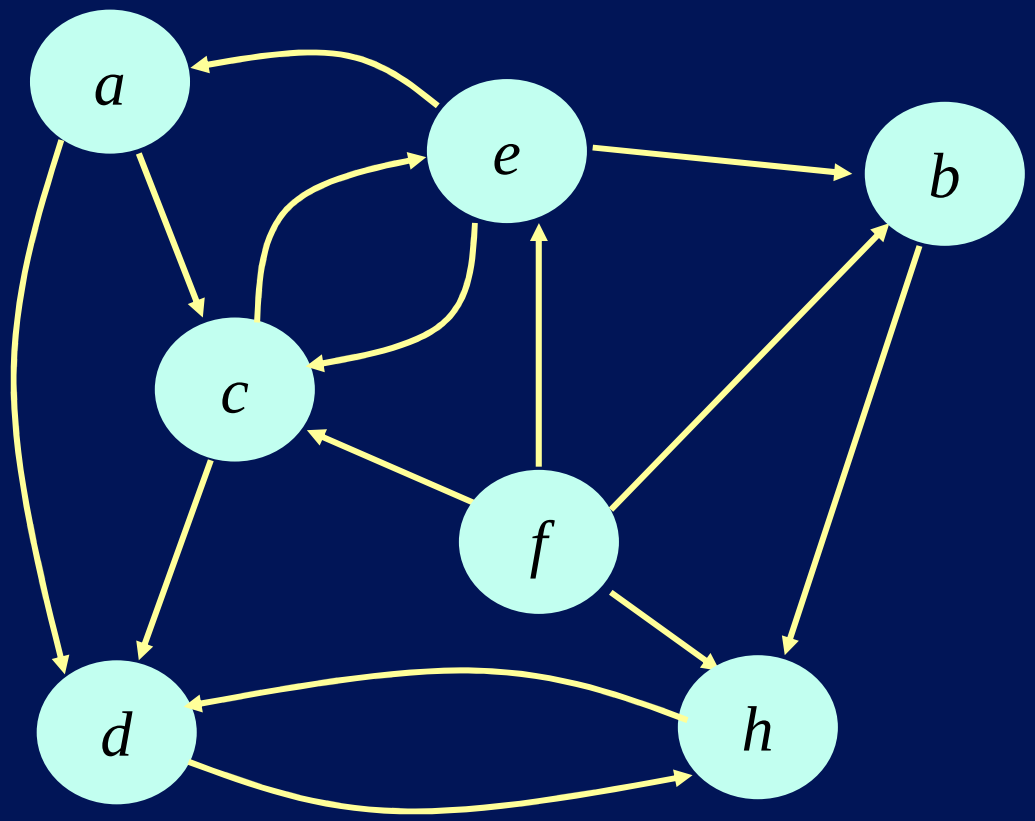
« Des points et des traits ou des flèches »

- Point de vue mathématique : une relation binaire
- Point de vue pratique : représentation abstraite d'un réseau (de télécommunication par exemple)

Les graphes, un outil magique, pour visualiser des échanges, pour la modélisation des systèmes réels, ou …pour jouer.

Les graphes sont utilisés dans des domaines très variés : économie, informatique, industrie, chimie, sociologie.

Un graphe orienté



$G=(X,U)=(X,\Gamma)$

X: {sommets}

(ou nœuds)

U: {arcs}

$\Gamma^+: X \rightarrow \mathcal{P}(X)$

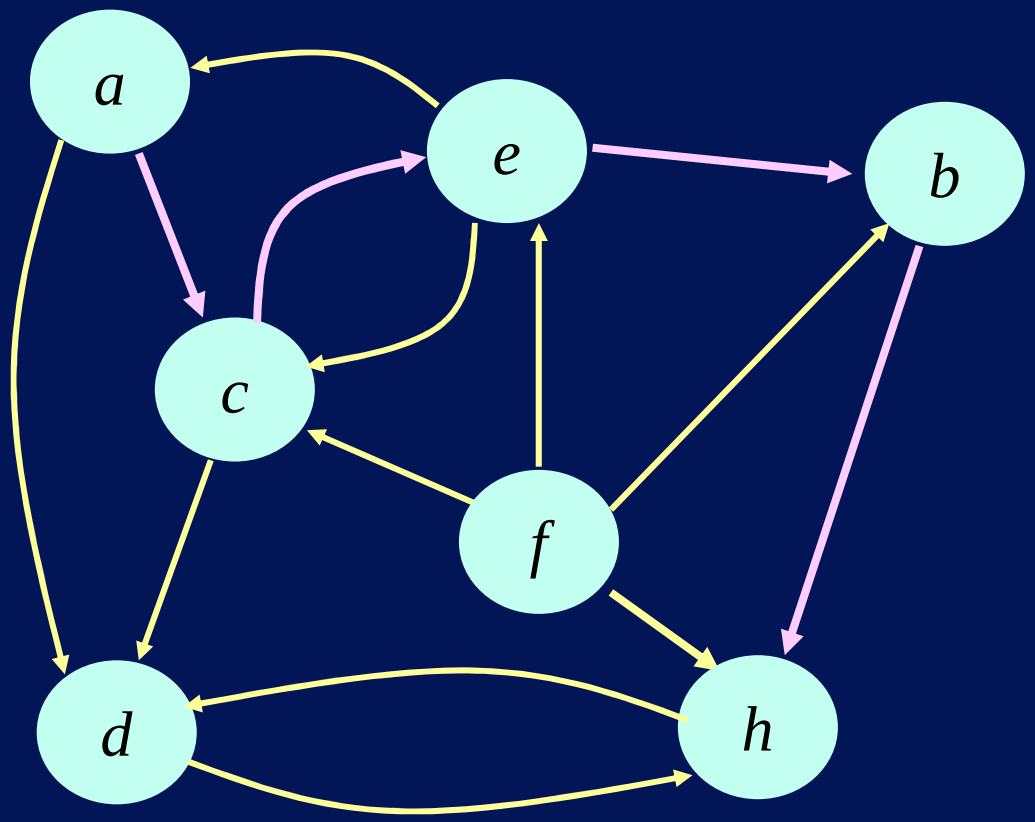
application multivoque « successeur »

$X=\{a,b,c,d,e,f,h\}$

$U=\{(a,c),(a,d),(b,h),\ldots,(f,e),(f,h)\}$

$\Gamma^+(e)=\{a,b,c\}$ $\Gamma^-(e)=\{c,f\}$ (Γ^- : prédécesseur)

Un graphe orienté

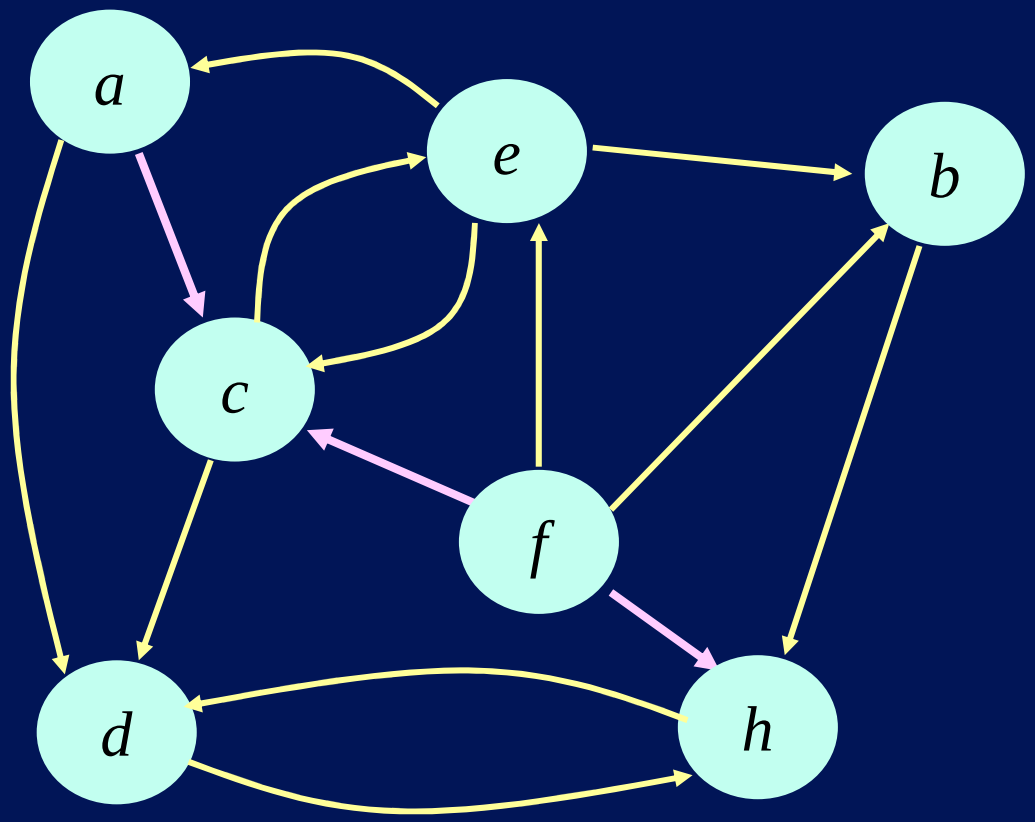


Chemin : suite d'arcs telle que l'extrémité terminale d'un arc coïncide avec l'extrémité initiale de l'arc suivant

Exemple

$\mu=((a,c),(c,e),(e,b),(b,h))=(a,c,e,b,h)$

Un graphe orienté



Chaîne : suite d'arcs telle que tout arc a une extrémité commune avec l'arc précédent (sauf éventuellement le premier) et l'autre avec l'arc suivant (sauf éventuellement le dernier)

Exemple

$\mu' = ((a,c),(f,c),(f,h)) = [a,c,f,h]$

Remarque: un chemin est une chaîne.

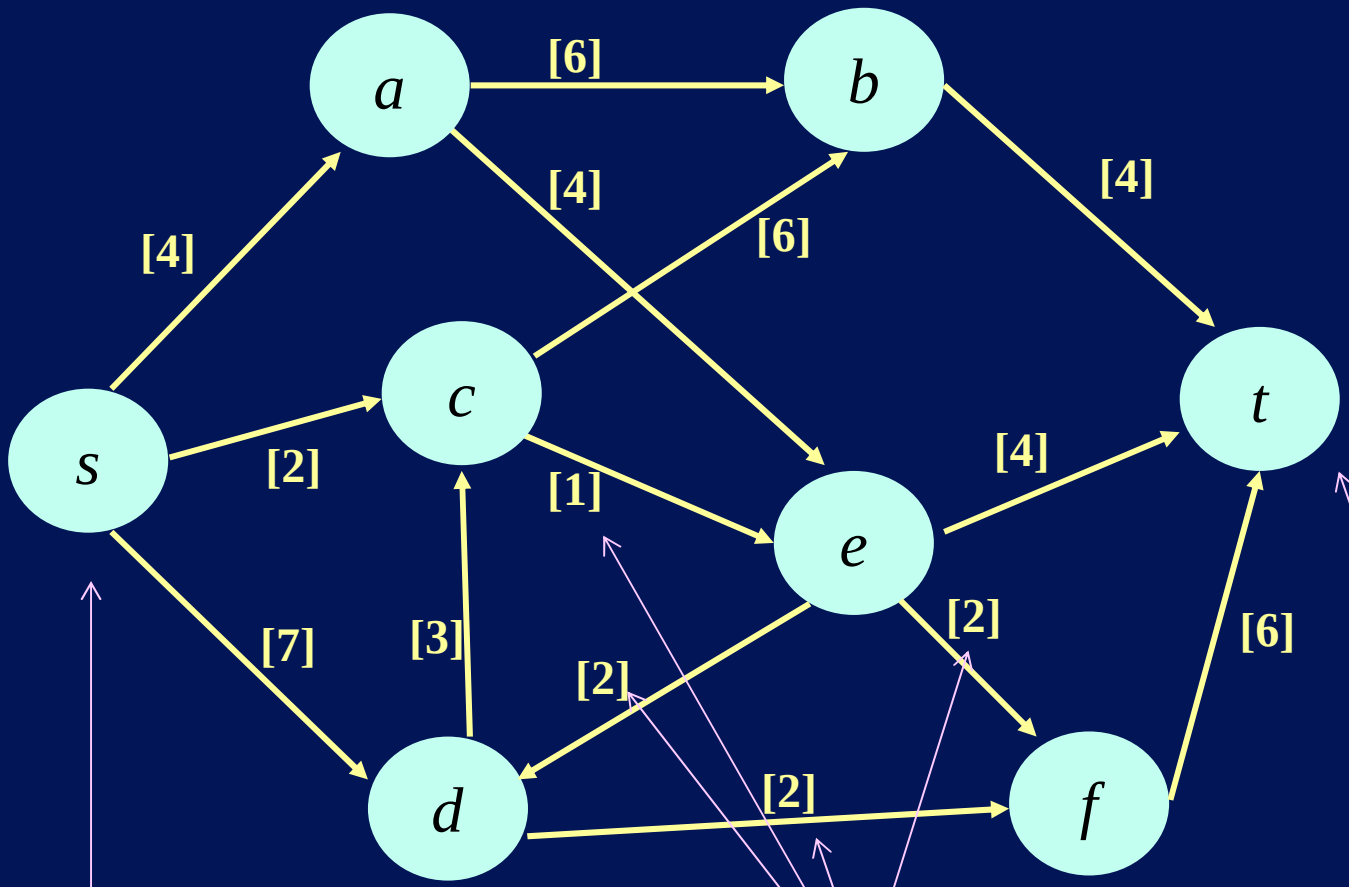
Réseau de transport

Graphe $R=(X,U,C)$ orienté dans lequel

1. Il existe une **source** s tel que $\Gamma^-(s)=\varnothing$
2. Il existe un **puits** t tel que $\Gamma^+(t)=\varnothing$
3. On associe à chaque arc $u=(i,j)$ de U une **capacité** $c_{ij} \geq 0$
4. Sur ce réseau, on trace un arc (fictif) de t à s (arc de retour)

Un réseau de transport

$R=(X,U,C)$



capacités

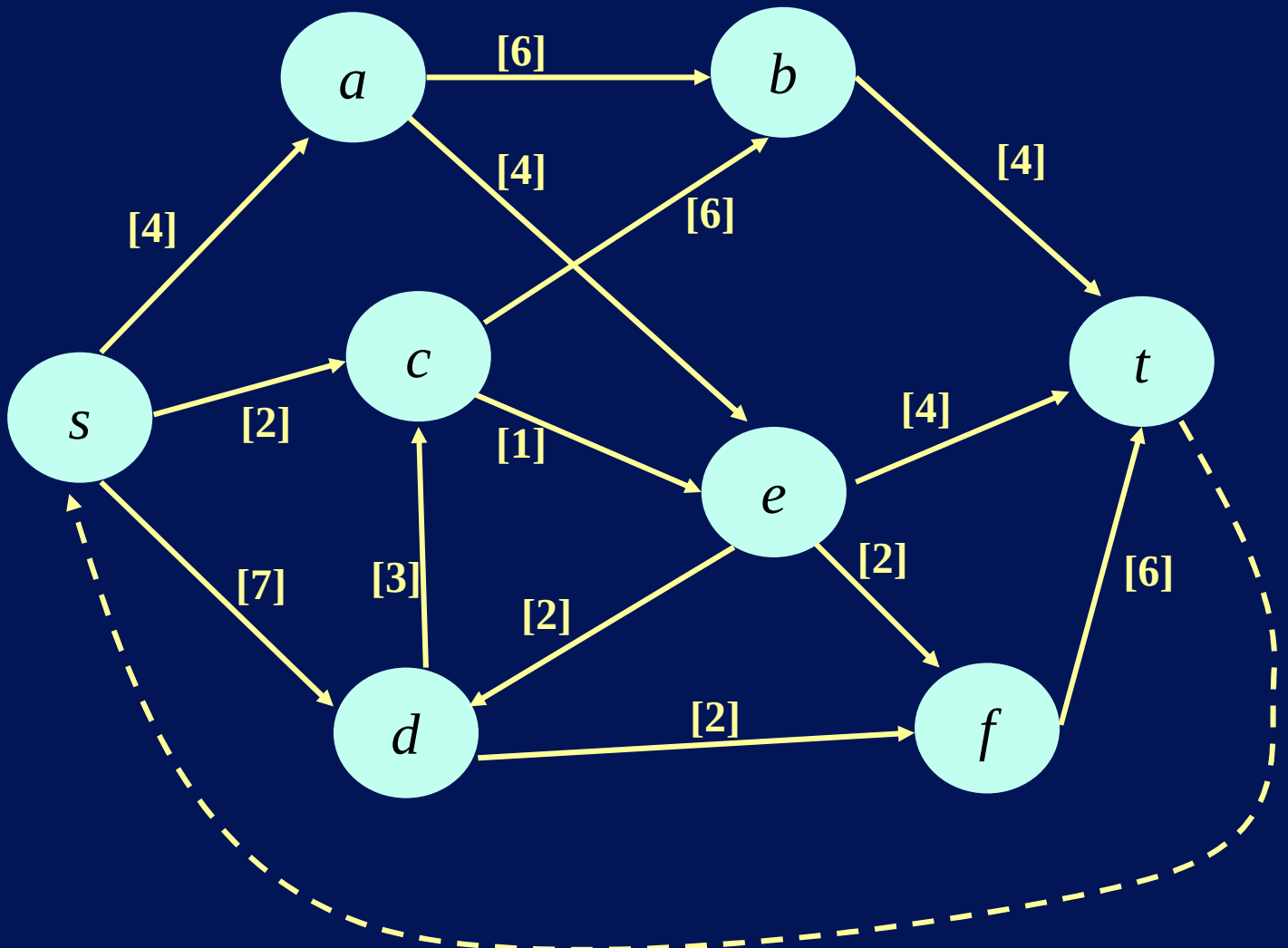
source

puits

capacités

Un réseau de transport

$R=(X,U,C)$

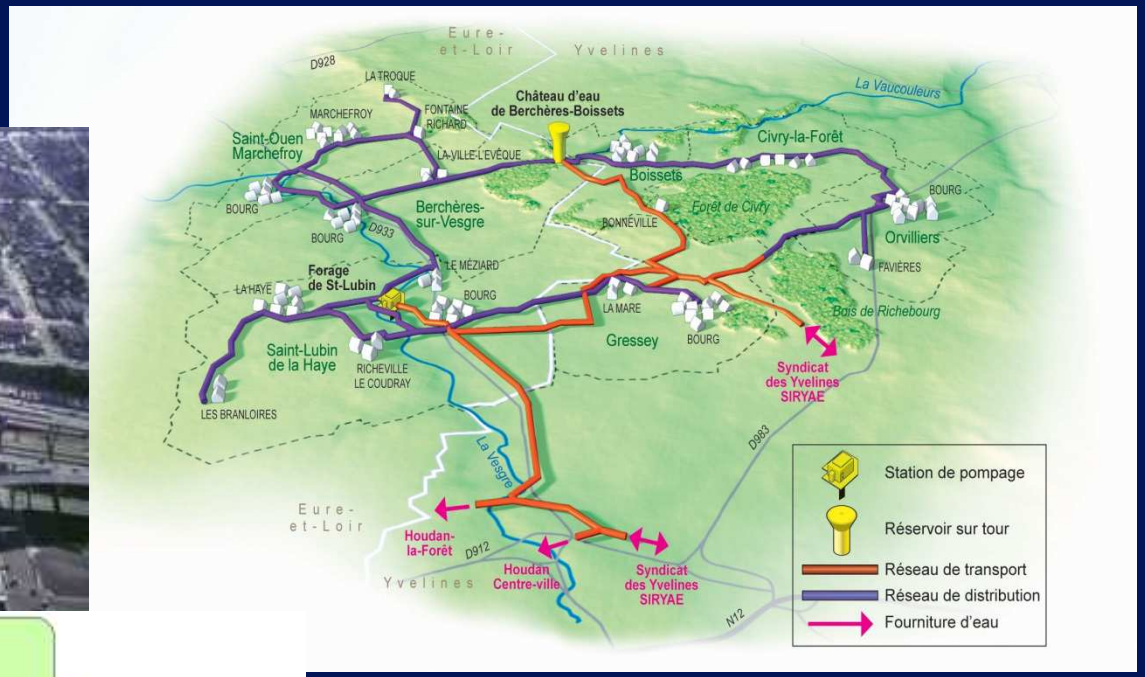


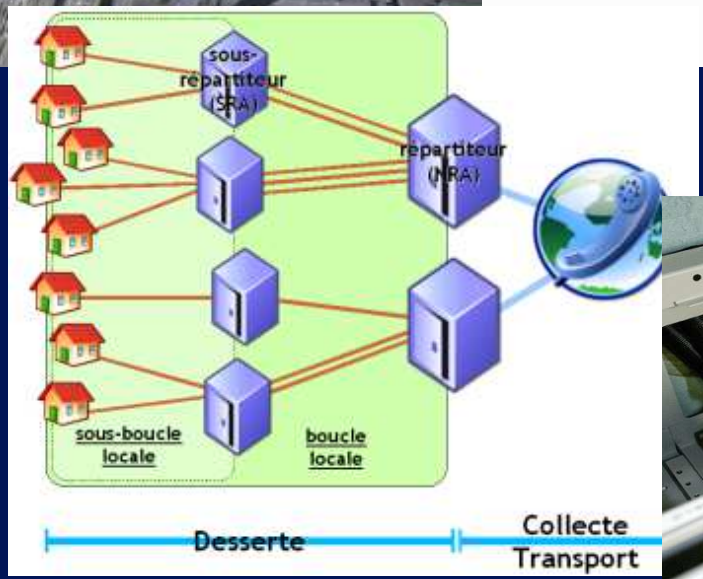
« Arc de retour » fictif (de très grande capacité)

2. Le problème du flot maximal

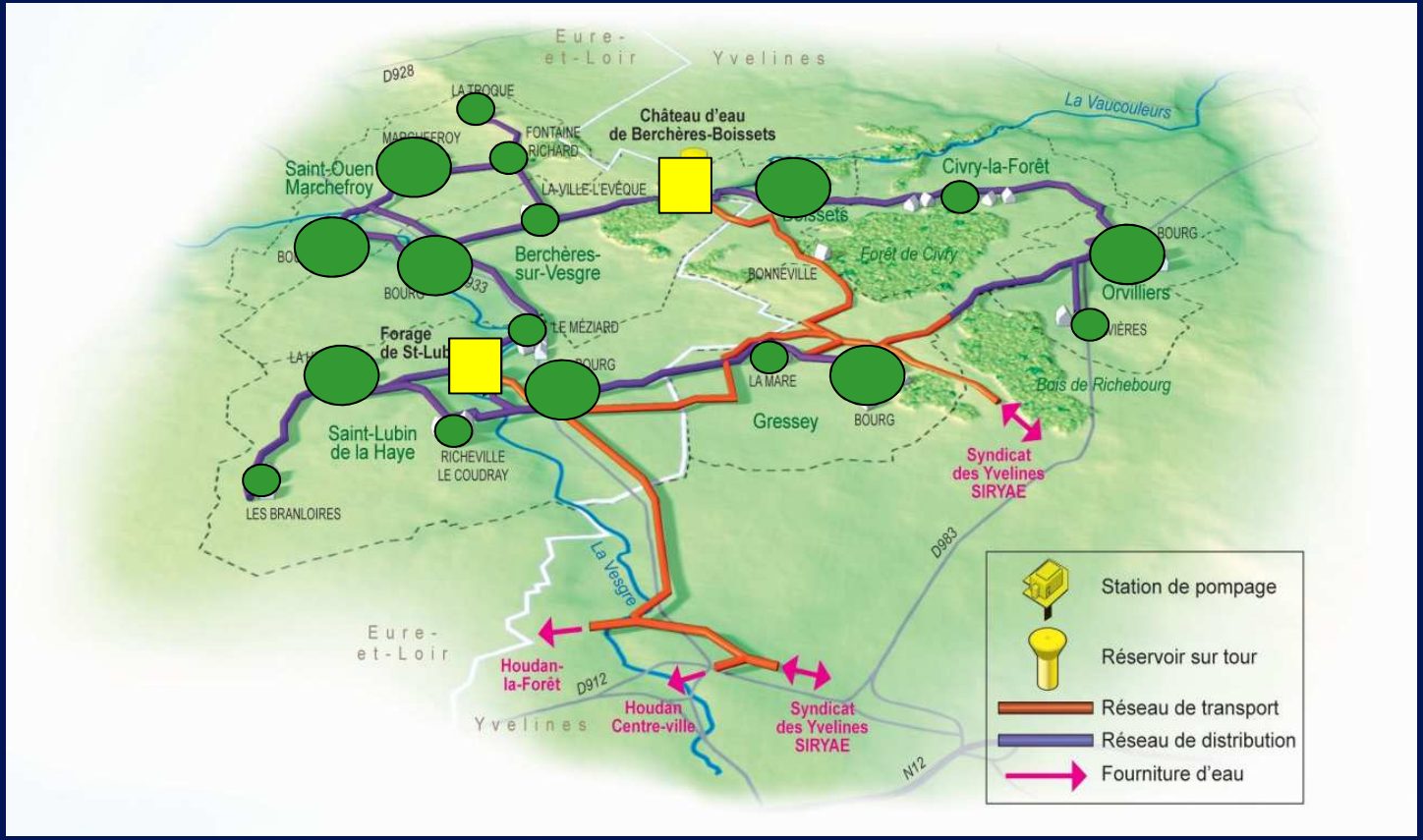
Comment transférer une quantité maximale de « matière » de *s* à *t* sans dépasser la capacité de chaque arc ?

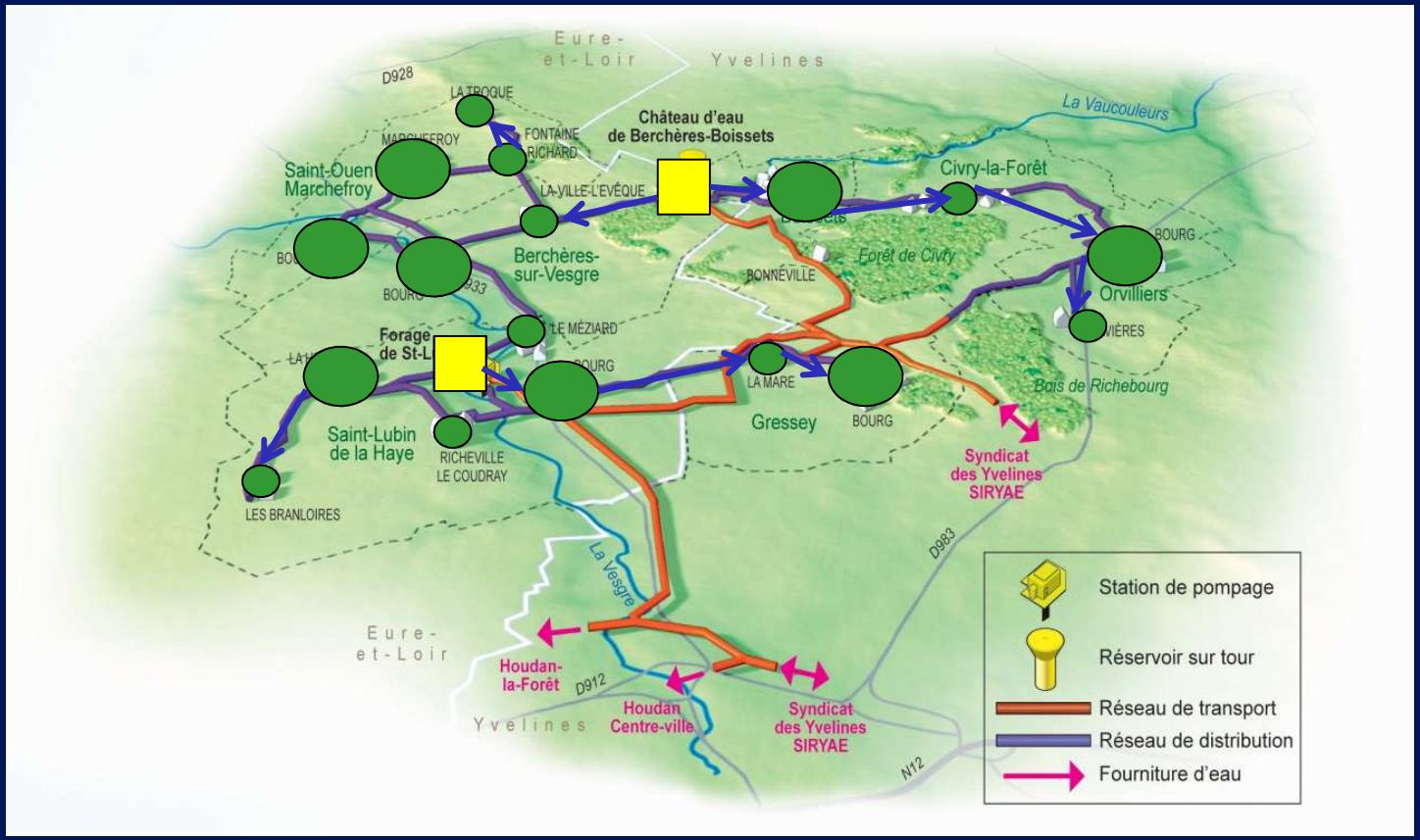
Le problème du flot maximal

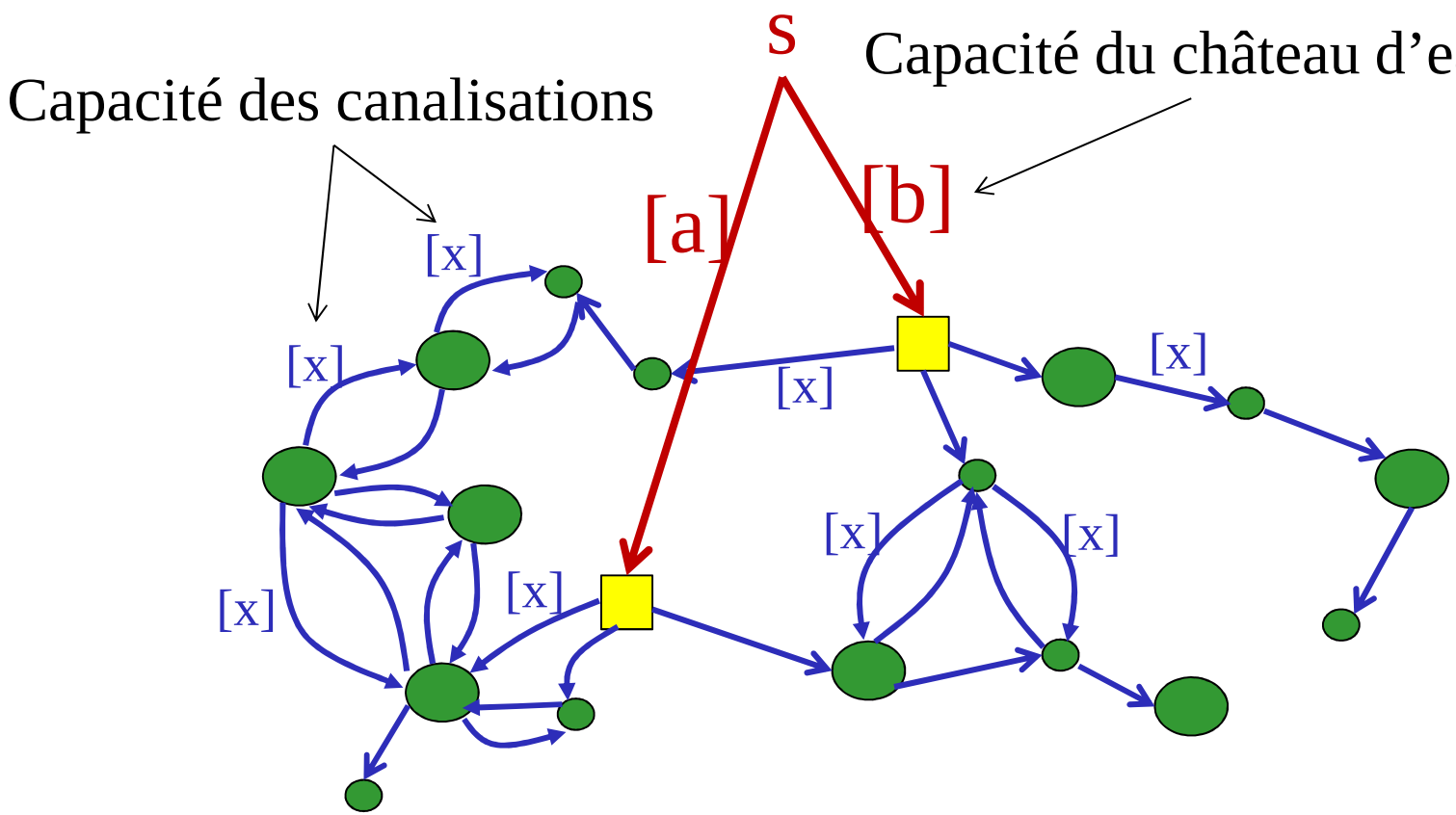


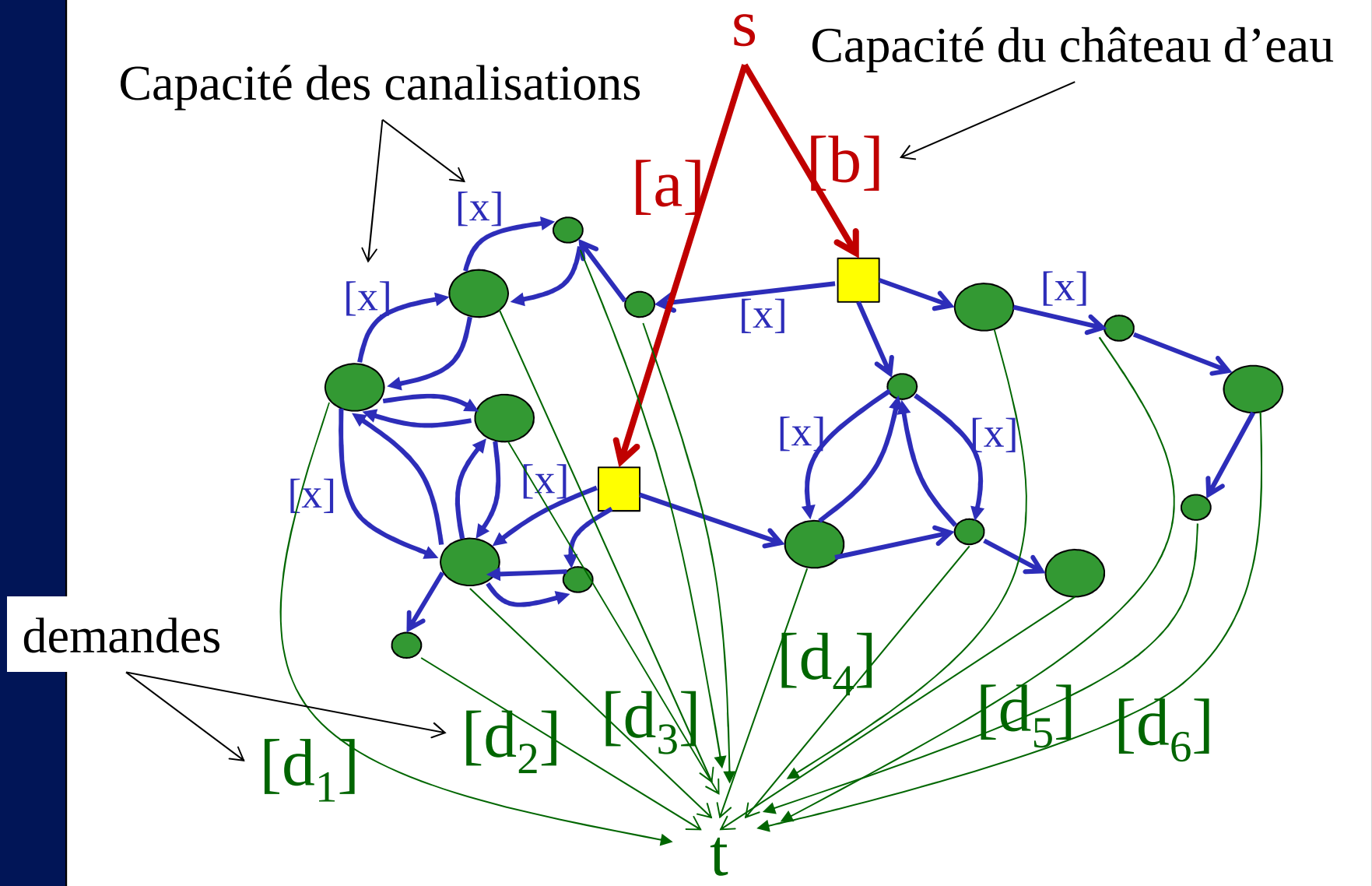


Un réseau de distribution de l'eau




Eure-et-Loir
Yvelines
D928
LA TROQUE
FONTAINE RICHARD
Château d'eau de Berchères-Boissets
La Vaucouleurs
Saint-Ouen Marchefroy
LA-VILLE-L'EVÊQUE
Civry-la-Forêt
Berchères-sur-Vesgre
Forêt de Civry
BONNEVILLE
BOURG
Orvilliers
LE MÉZIARD
Forage de St-L
LA MARE
Bois de Richebourg
Gressey
Saint-Lubin de la Haye
RICHEVILLE LE COUDRAY
LES BRANLOIRES
Syndicat des Yvelines SIRYAE
D983
La Vesgre
Houdan-la-Forêt
D912
Houdan Centre-ville
N12
Station de pompage
Réservoir sur tour
Réseau de transport
Réseau de distribution
Fourniture d'eau


s
Capacité du château d'eau
Capacité des canalisations
[a]
[b]
[x]
[x]
[x]
[x]
[x]
[x]
[x]
[x]

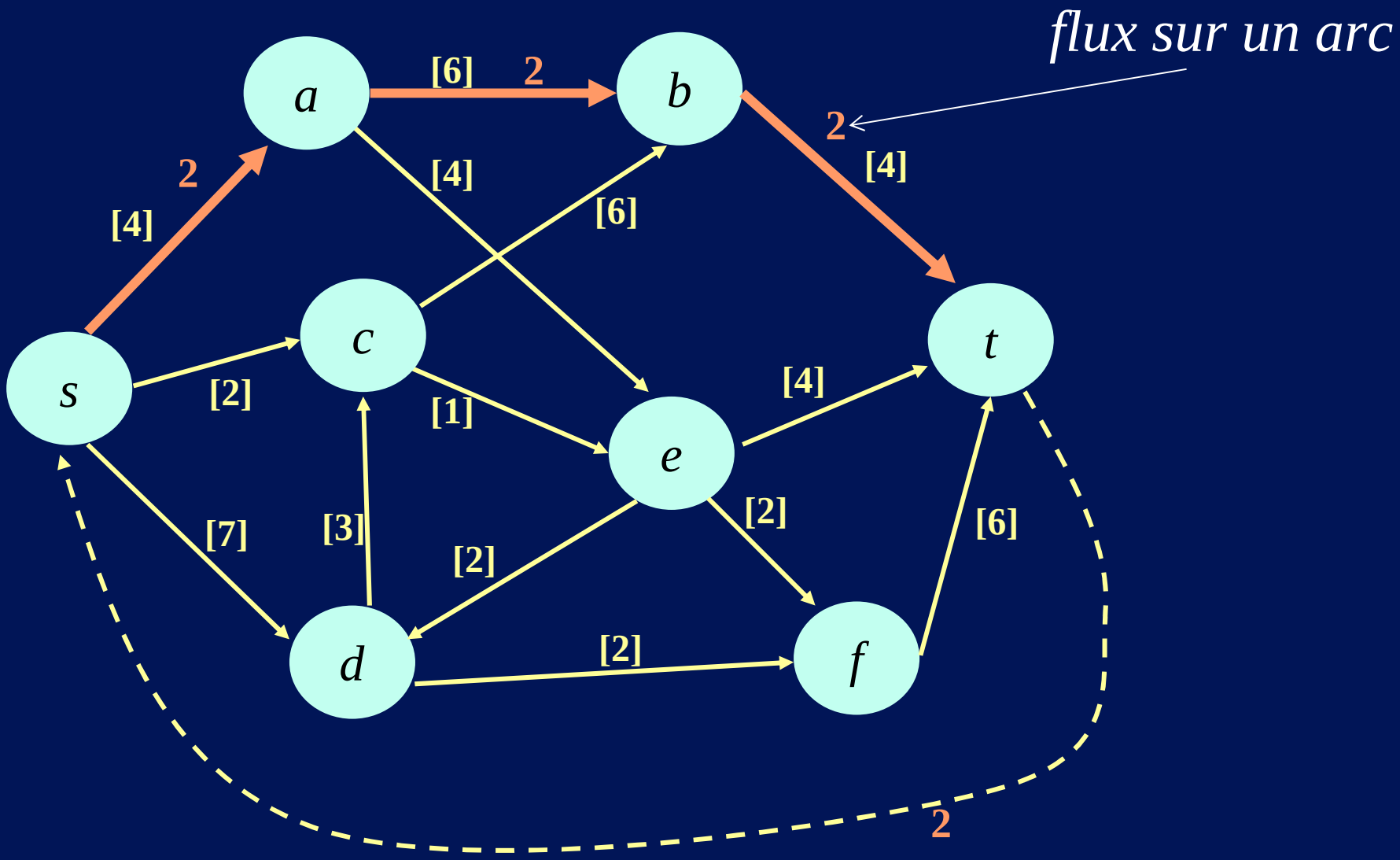

s
Capacité du château d'eau
Capacité des canalisations
[a]
[b]
[x]
[x]
[x]
[x]
[x]
[x]
[x]
[x]
demandes
[d1]
[d2]
[d3]
[d4]
[d5]
[d6]
t

Le problème du flot maximal

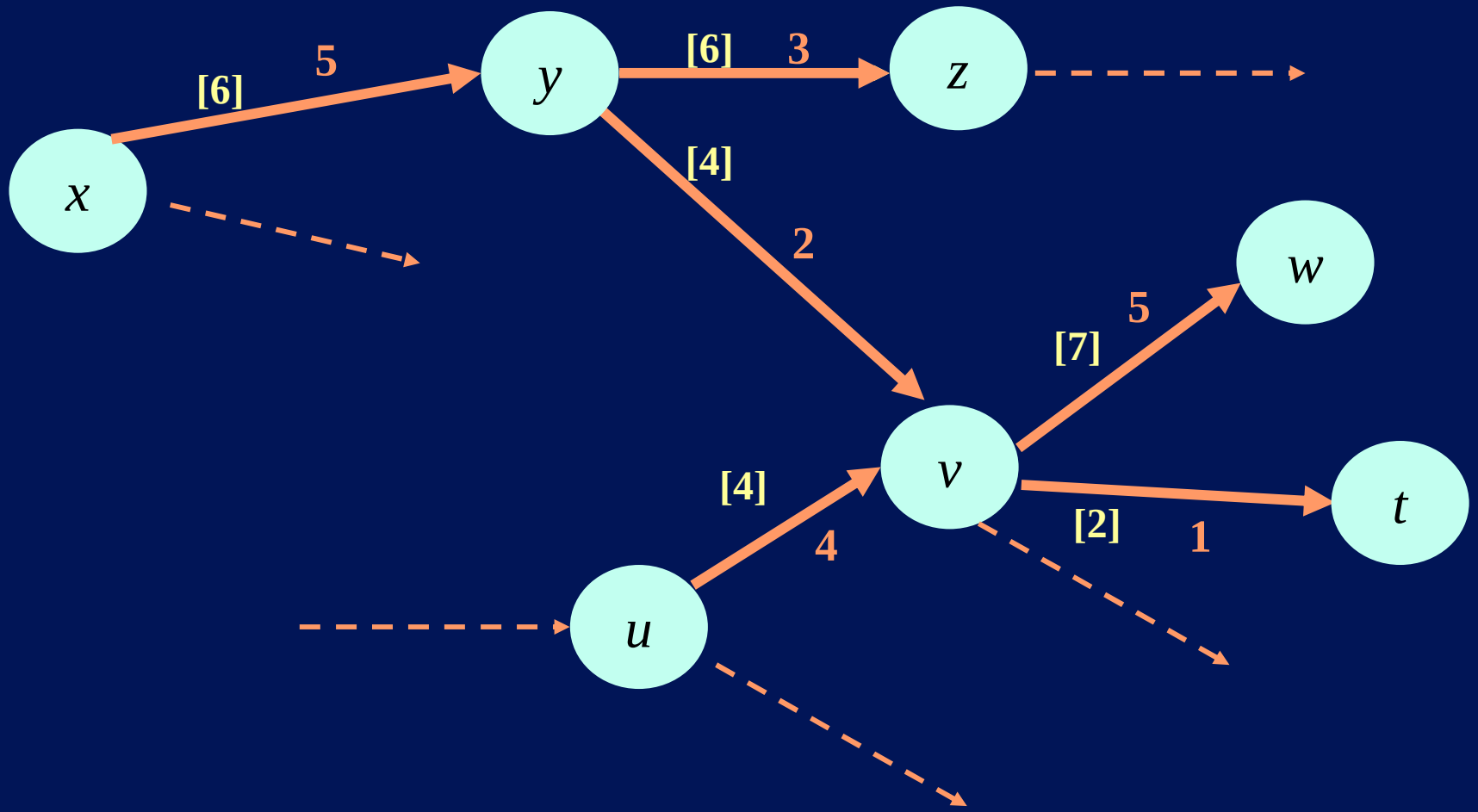
- Comment transférer une quantité maximale de « matière » de *s* à *t* sans dépasser la capacité de chaque arc ?

- Hypothèses:
  - On peut décomposer/recomposer la matière transférée en chaque nœud.
  - Il y a conservation de la matière en chaque nœud.

Un réseau de transport



Décomposition/recomposition des flux



φ_{ij} désigne le **flux** sur l'arc (i, j), qui correspond à la quantité de « matière » circulant sur l'arc.

Flot de s à t sur un réseau de transport

- Le vecteur $\varphi = (\varphi_{ij})_{(i,j)\in U\cup(t,s)}$ est appelé **flot** sur R.
- Les flux sur R doivent vérifier la **contrainte de capacité** : $0 \leq \varphi_{ij} \leq c_{ij}$
- Lorsque $\varphi_{ij} = c_{ij}$, l'arc (i, j) est dit **saturé**.
- Les flux sur R doivent également respecter la **loi de conservation** (≈loi de Kirchhoff en électricité 1845) :

$$\forall j \in X, \quad \sum_{i\in\Gamma^-} \varphi_{ij} = \sum_{i\in\Gamma^+} \varphi_{ji}$$

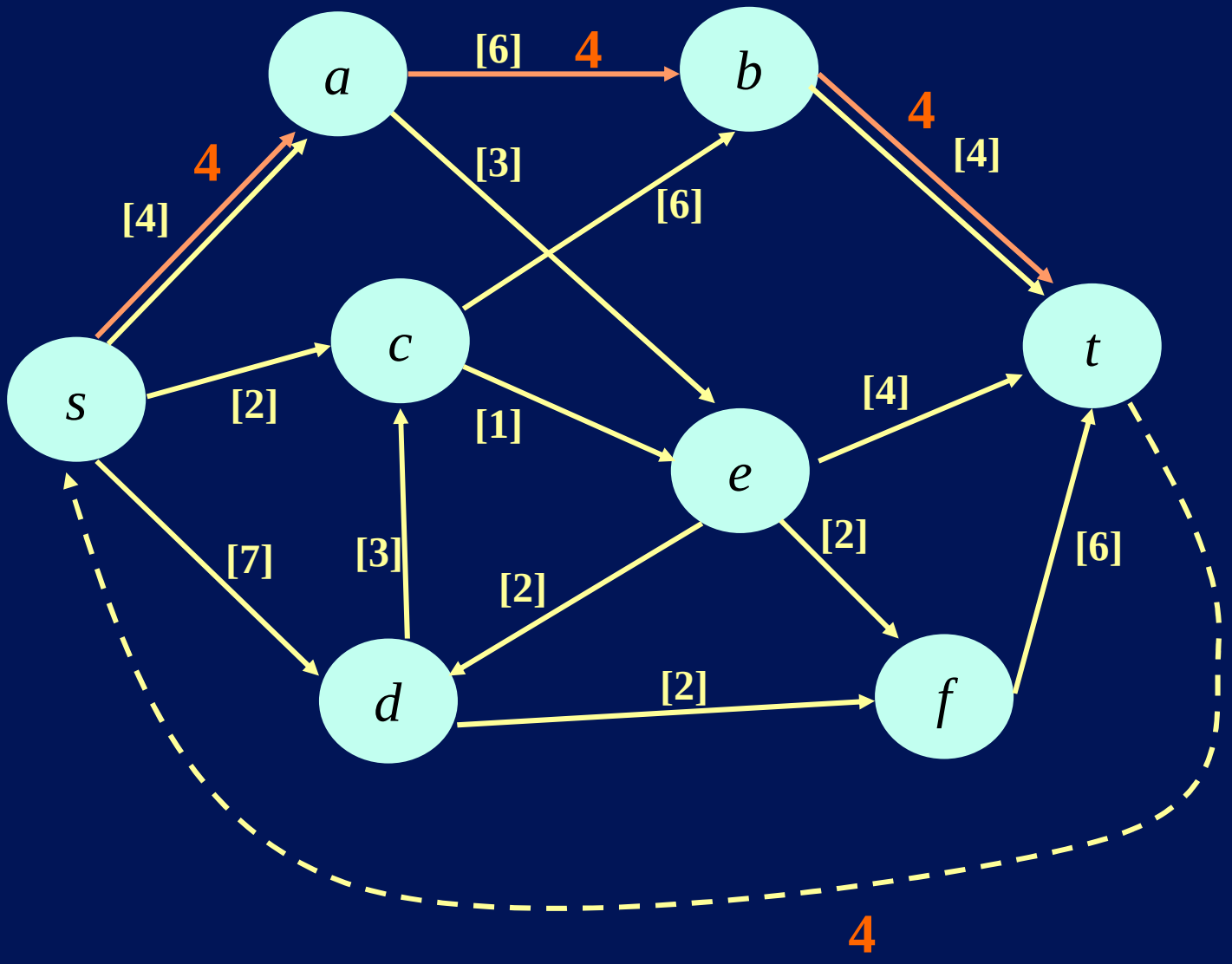
Flot sur un réseau de transport

- La loi de conservation est également vérifiée en *s* et *t* grâce à l'adjonction de l'arc de retour.

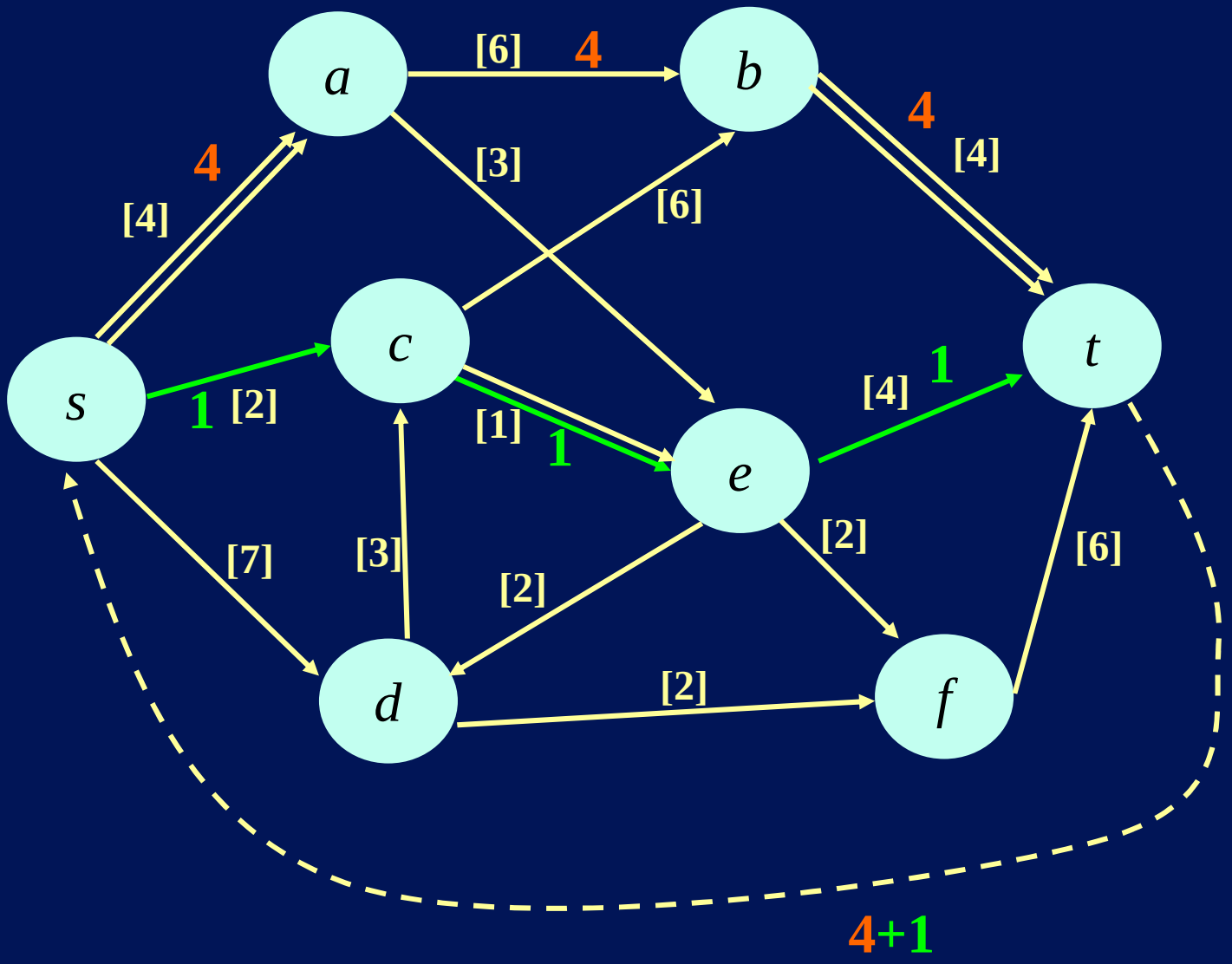
- Un flot φ est dit **réalisable** (ou **admissible** ou **compatible**) si et seulement si :
  - il respecte les contraintes de capacité sur chaque arc
  - Il respecte la loi de conservation en chaque sommet

- Un flot φ est dit **complet** si et seulement si tout chemin de *s* à *t* comporte au moins un arc saturé.

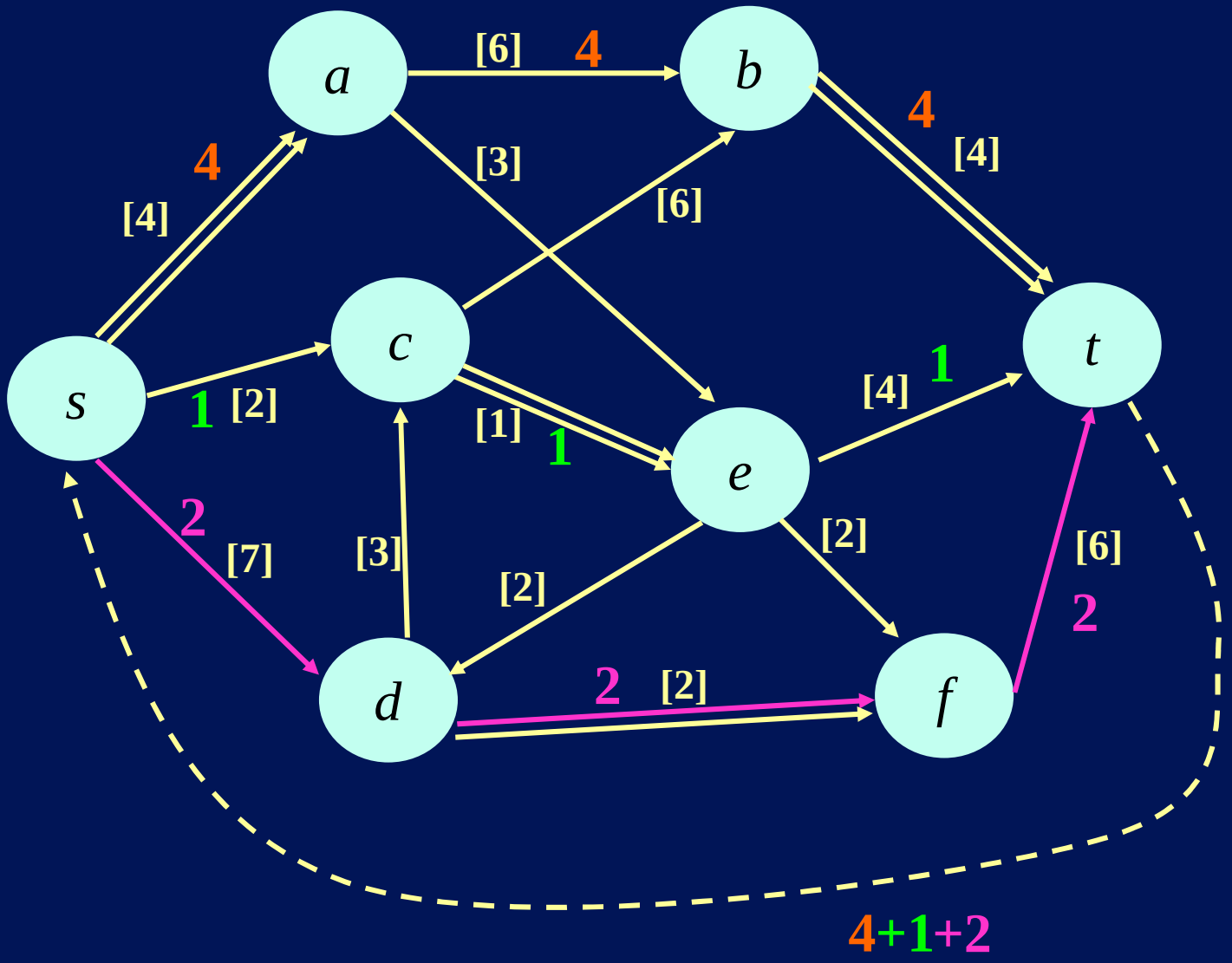
Un réseau de transport



Un réseau de transport



Un réseau de transport



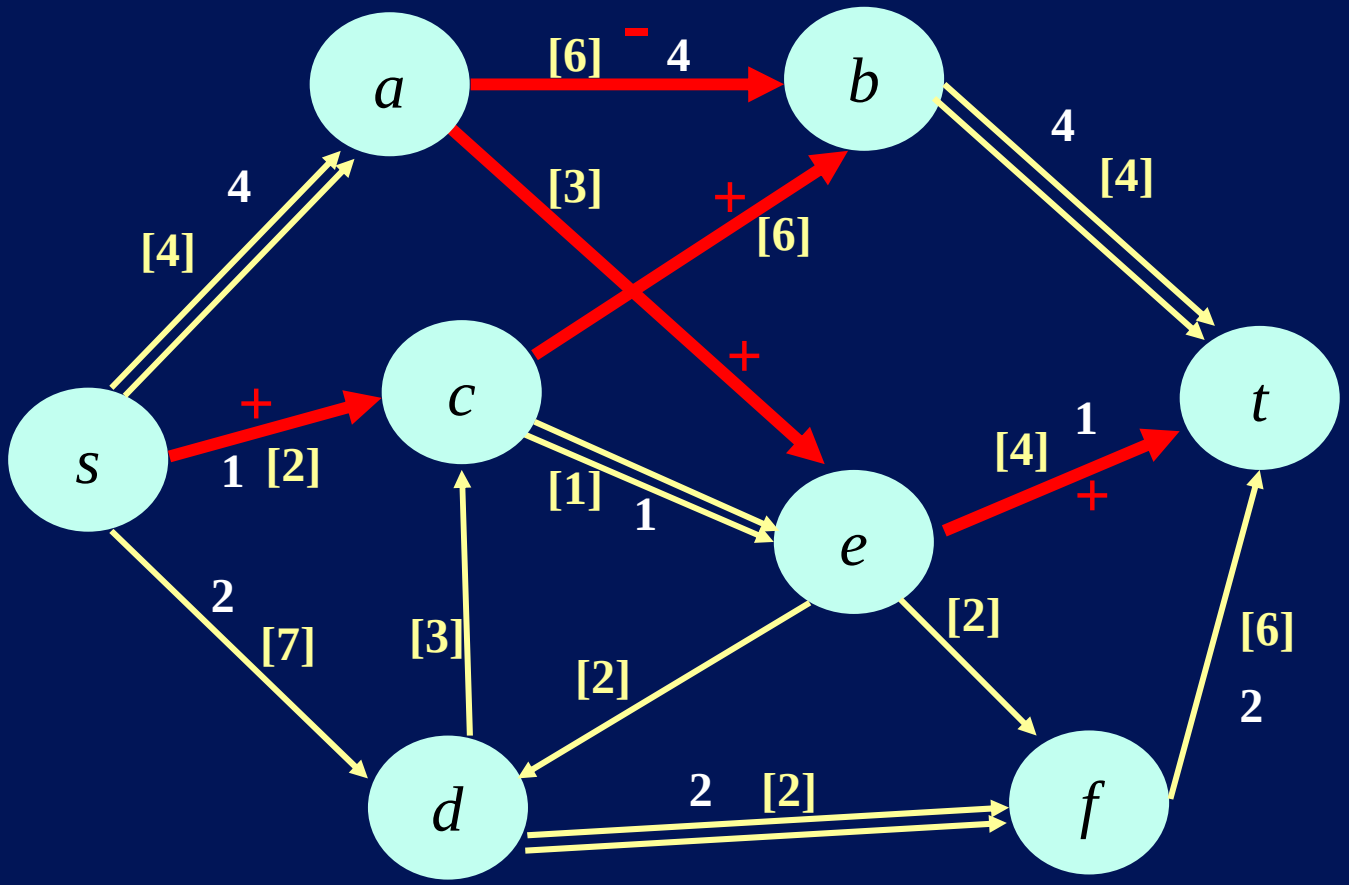
Le flot est complet et $v(\varphi)=7$

Chaîne améliorante

μ est une **chaîne améliorante** (ou augmentante) de s à t pour un flot φ admissible donné si :

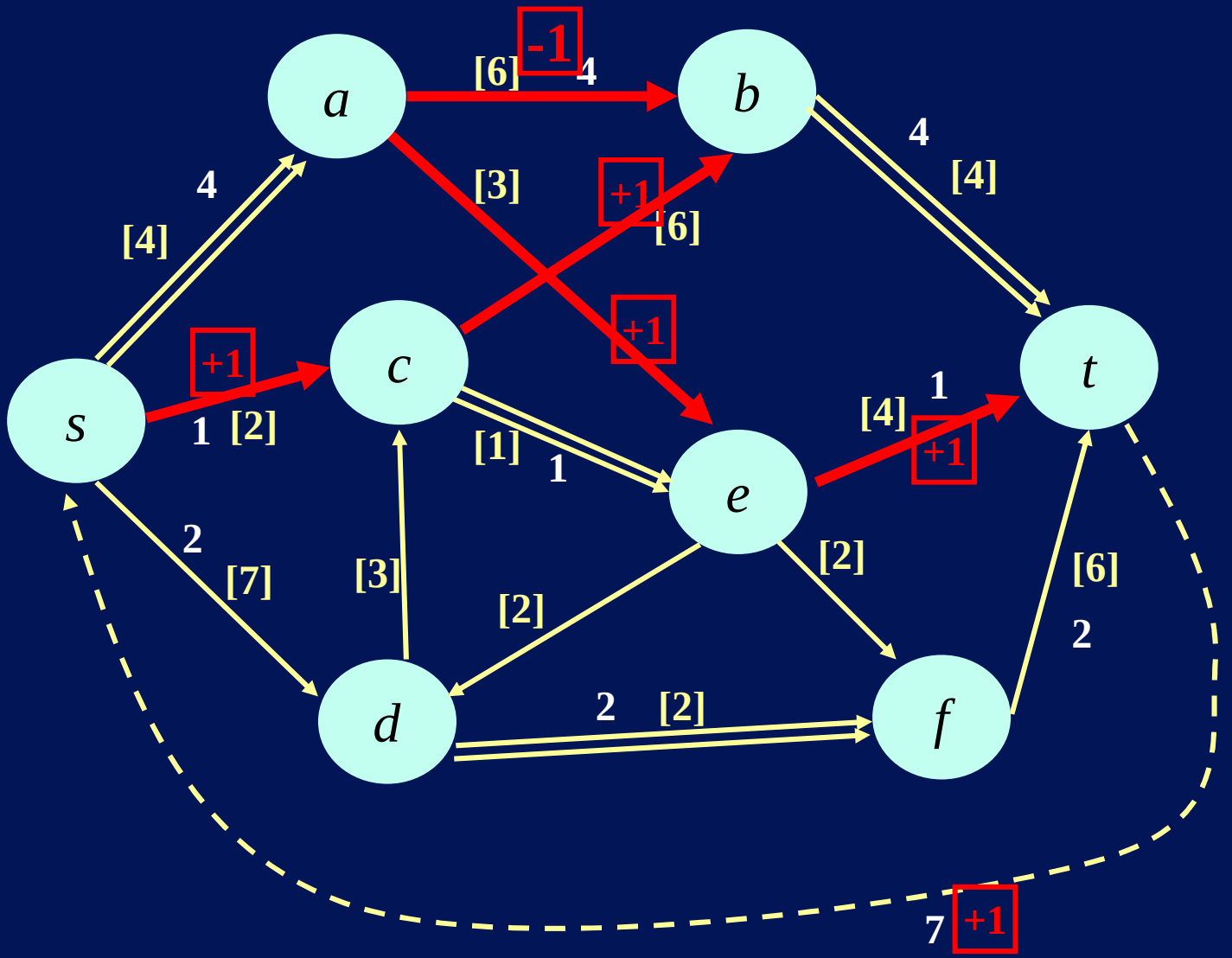
- $\varphi_{ij} < c_{ij}$ pour tout arc (i, j) de μ dans le "bon sens" (de s vers t)
- $\varphi_{ij} > 0$ pour tout arc (i, j) de μ dans le "mauvais sens" (de t vers s)

Chaîne améliorante



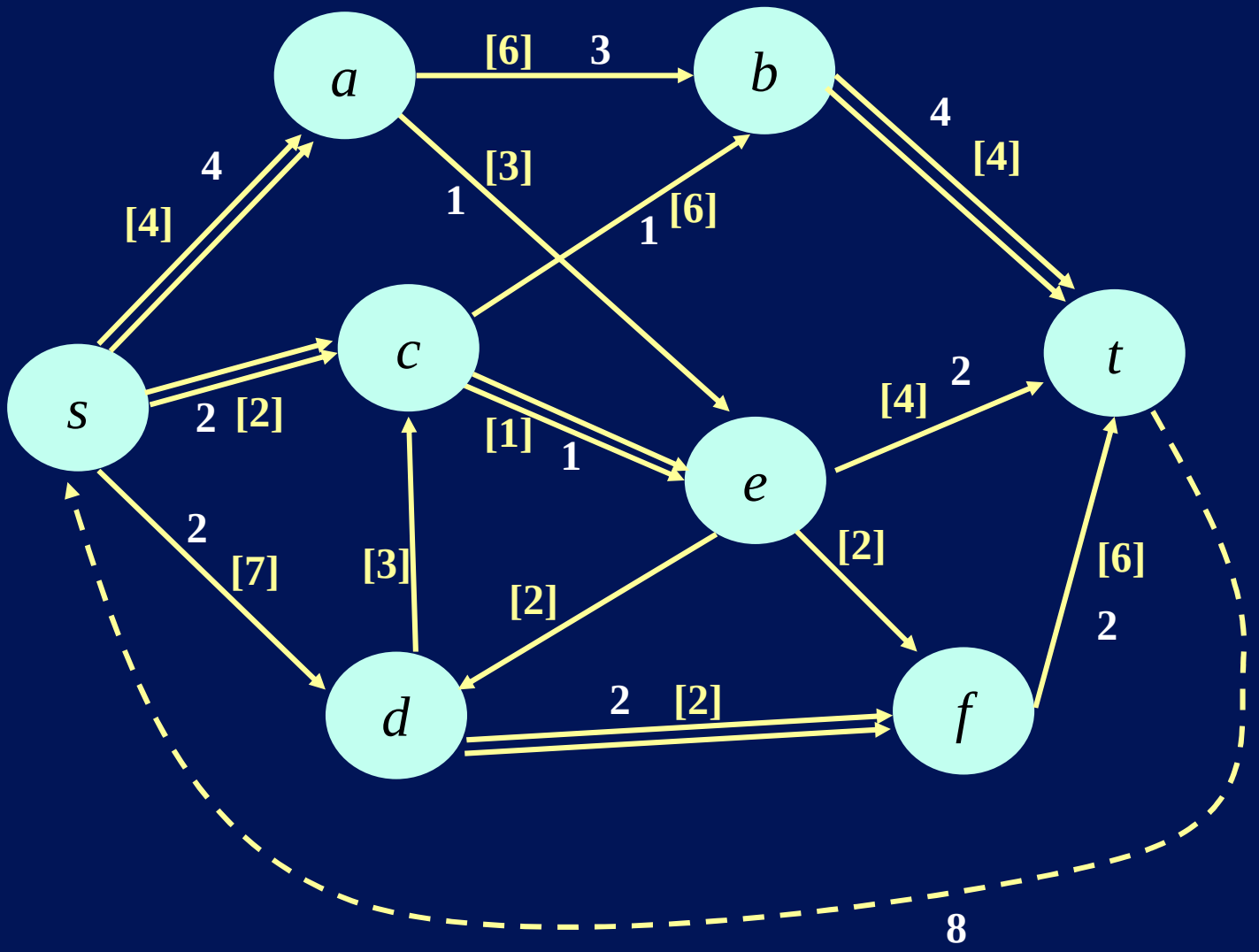
Amélioration de: min (2-1,6-0,4, 3,4-1)=1

Chaîne améliorante



$v(\varphi)=8$

Chaîne améliorante



$v(\varphi)=8$

Chaîne améliorante

- On note :
  - μ^+ = {arcs de μ dans le bon sens}
  - μ^- = {arcs de μ dans le mauvais sens}
- Augmentation de la valeur du flot de α :

$$\alpha = \min\left[\min_{(i,j)\in\mu^+}(c_{ij}-\varphi_{ij}),\ \min_{(i,j)\in\mu^-}\varphi_{ij}\right]$$

- Dans μ^+ : on augmente les flux de α
- Dans μ^- : on diminue les flux de α

3. Un algorithme simple (Ford-Fulkerson)

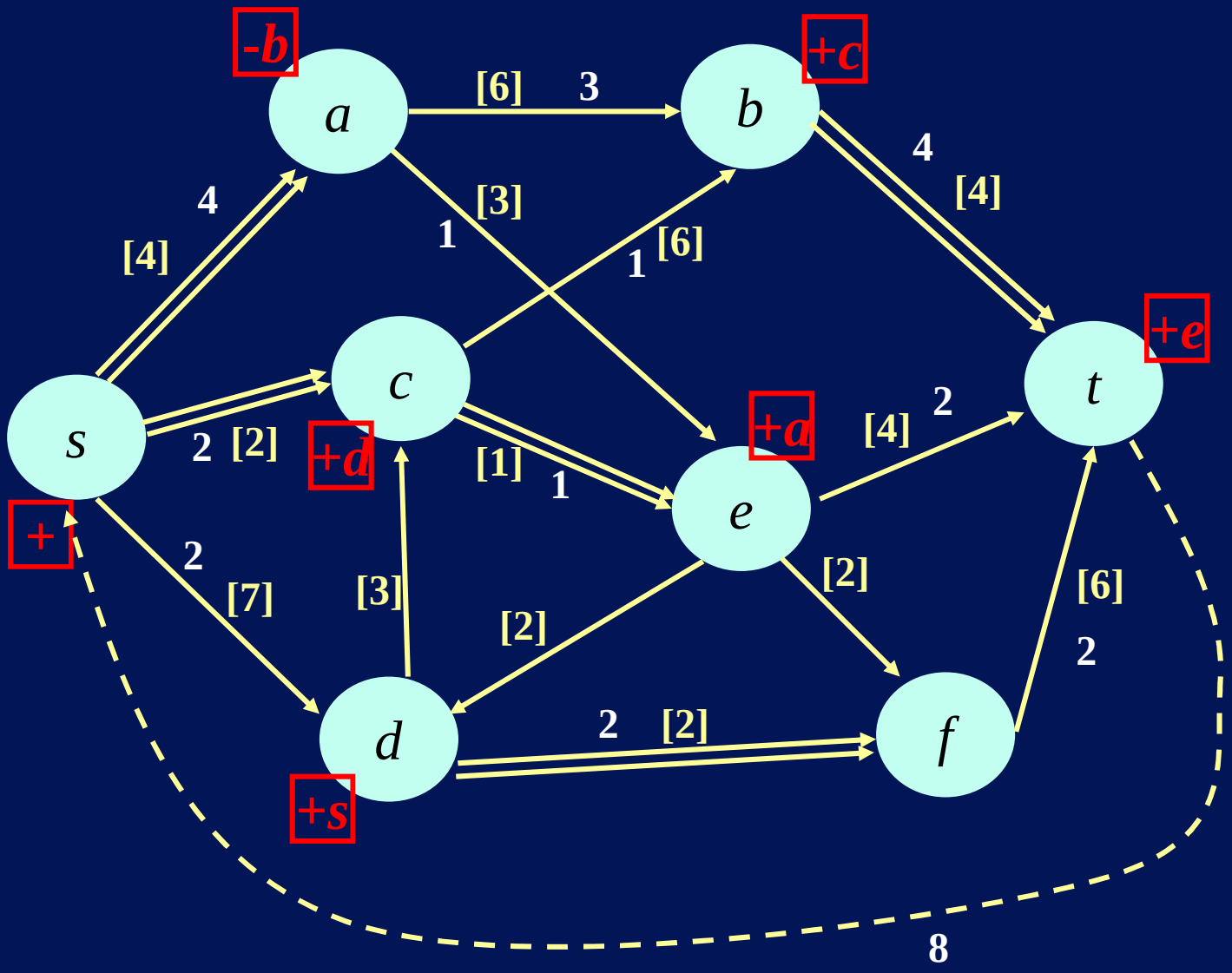
Idées de l'algorithme FF de recherche d'un flot maximal:

- Trouver un flot initial (complet)
- Tant que c'est possible:
  - chercher une chaîne améliorante
  - améliorer le flot le long de cette chaîne
- Il n'y a plus de chaîne améliorante: le flot ainsi obtenu est optimal . (*preuve plus loin*)

L’algorithme de Ford-Fulkerson: une procédure de marquage

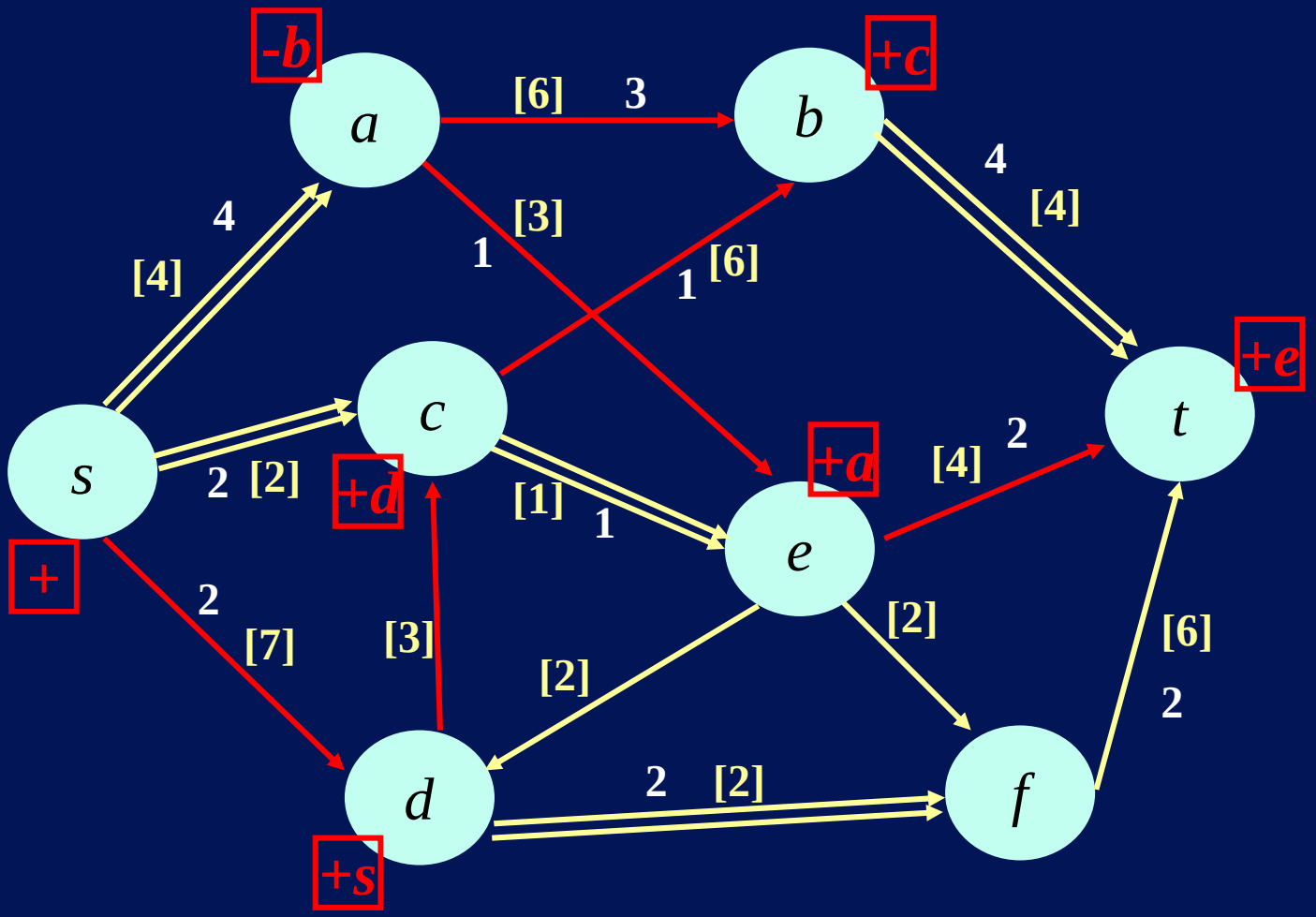
1. Établir, au jugé, un flot admissible, si possible complet
2. Marquer *s* d’un +, puis, au fur et à mesure, marquer :
   a) L’extrémité terminale *j* de tout arc (*i*, *j*) non saturé tel que *i* est marqué (on marque ‘+*i* ’ le sommet *j*), puis quand ces marquages sont achevés,
   b) L’extrémité initiale *i* de tout arc (*i*, *j*) de flux non nul tel que *j* est marqué (on marque ‘-*j* ’ le sommet *i*).
   c) Retour en a) jusqu’à ce que :
      i. Le marquage ne peut se poursuivre et on n’a pas pu marquer *t* : le flot maximal a alors été trouvé : STOP
      ii. Ou bien on a marqué *t* : choisir une chaîne améliorante, augmenter le flux le long de cette chaîne et retour en 2.

Chaîne améliorante



$v(\varphi)=8$

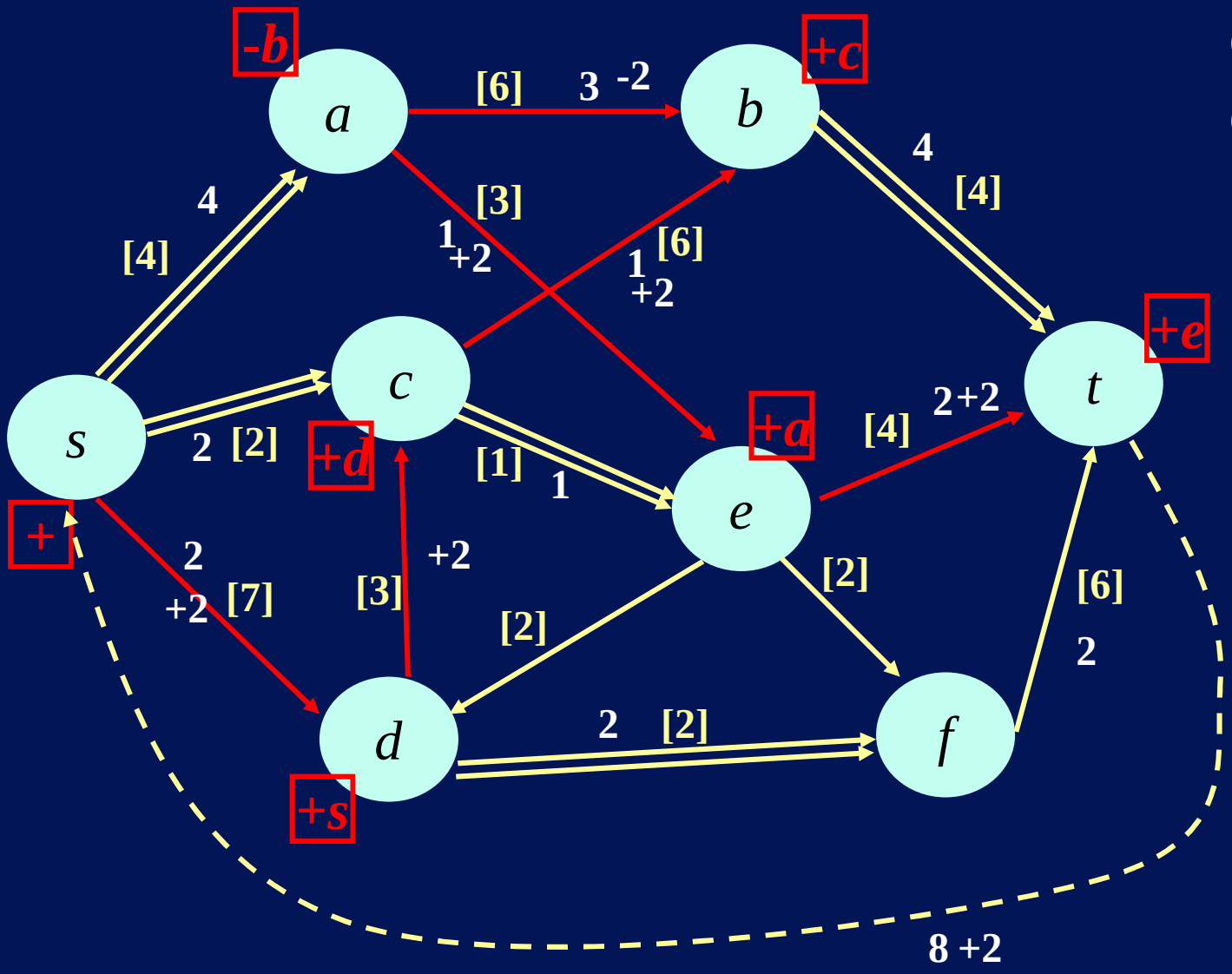
Chaîne améliorante



Amélioration de +2 le long de la chaîne

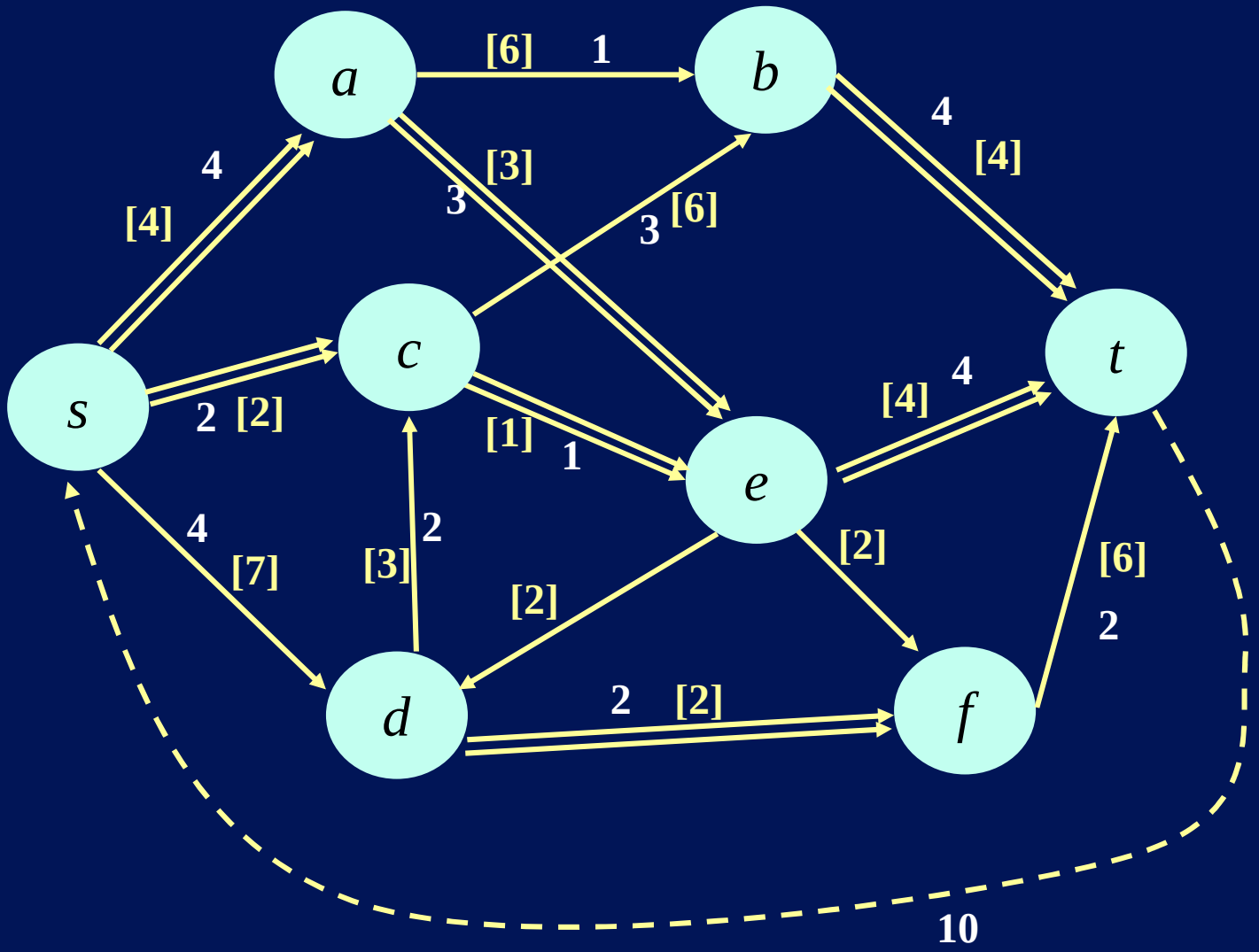
$$\mu \qquad s \xrightarrow[+]{5} d \xrightarrow[+]{3} c \xrightarrow[+]{5} b \xleftarrow[-]{3} a \xrightarrow[+]{2} e \xrightarrow[+]{2} t$$

Chaîne améliorante



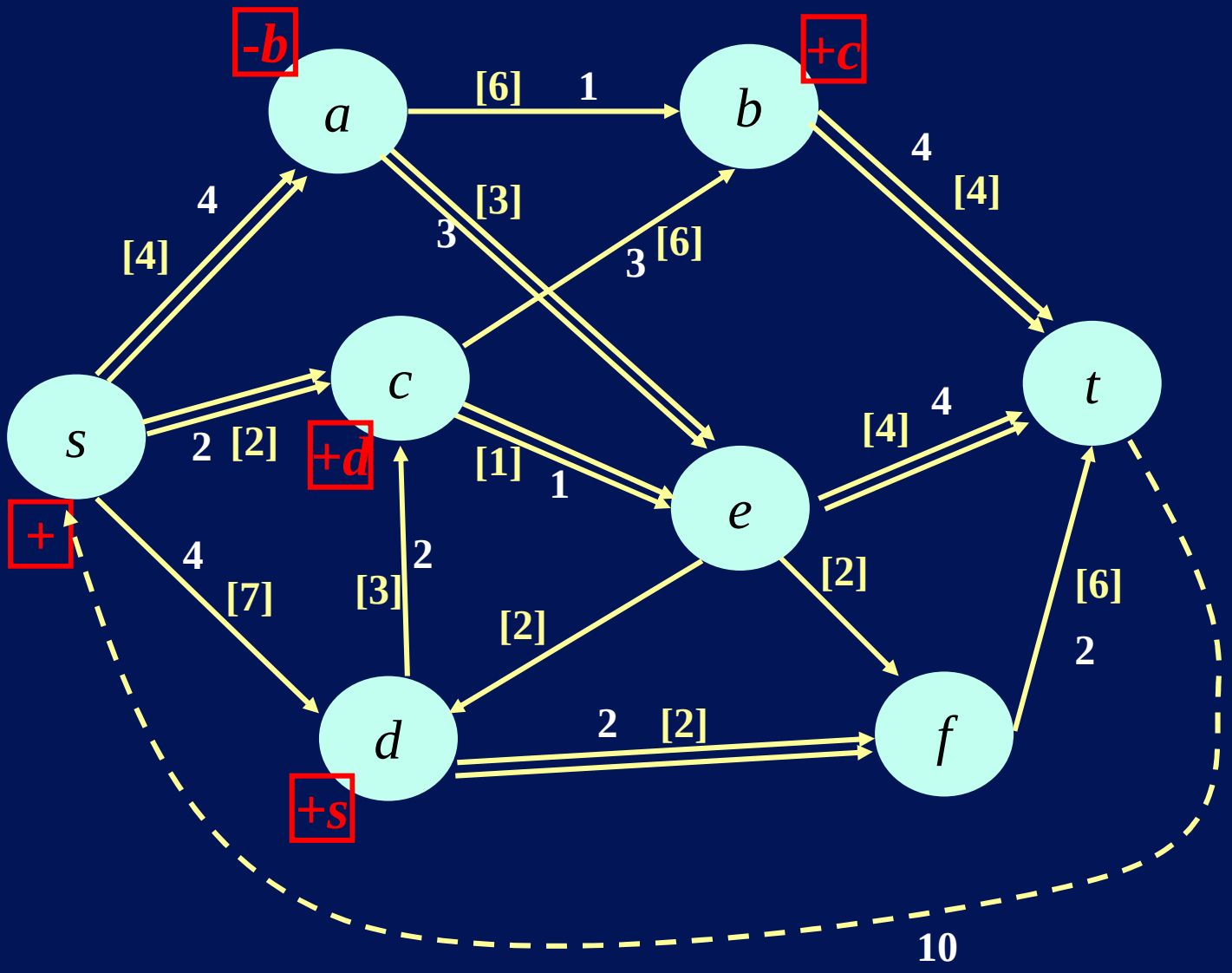
Amélioration de +2 le long de la chaîne

v(φ)=10



$v(\varphi)=10$

t non marqué: flot maximal



v(φ*)=10

4. Flot maximal et coupe minimale

Preuve de l'algorithme

Comment séparer la source *s* du puits *t* en supprimant un ensemble d'arcs de valeur totale minimale ?

« Séparer » veut dire qu'il n'existe plus de chemin de *s* à *t* après la suppression des arcs.

Remarque: les valeurs sur les arcs peuvent être des capacités, des coûts, des poids,…

Une **coupe** (S, T) est une partition de X en deux sous-ensembles S et T t.q. $s \in S$ et $t \in T$

On note :

$\omega^-(T)$ = {arcs entrant dans T} = $\{(i, j) \in U$ t.q. $i \in S$ et $j \in T\}$

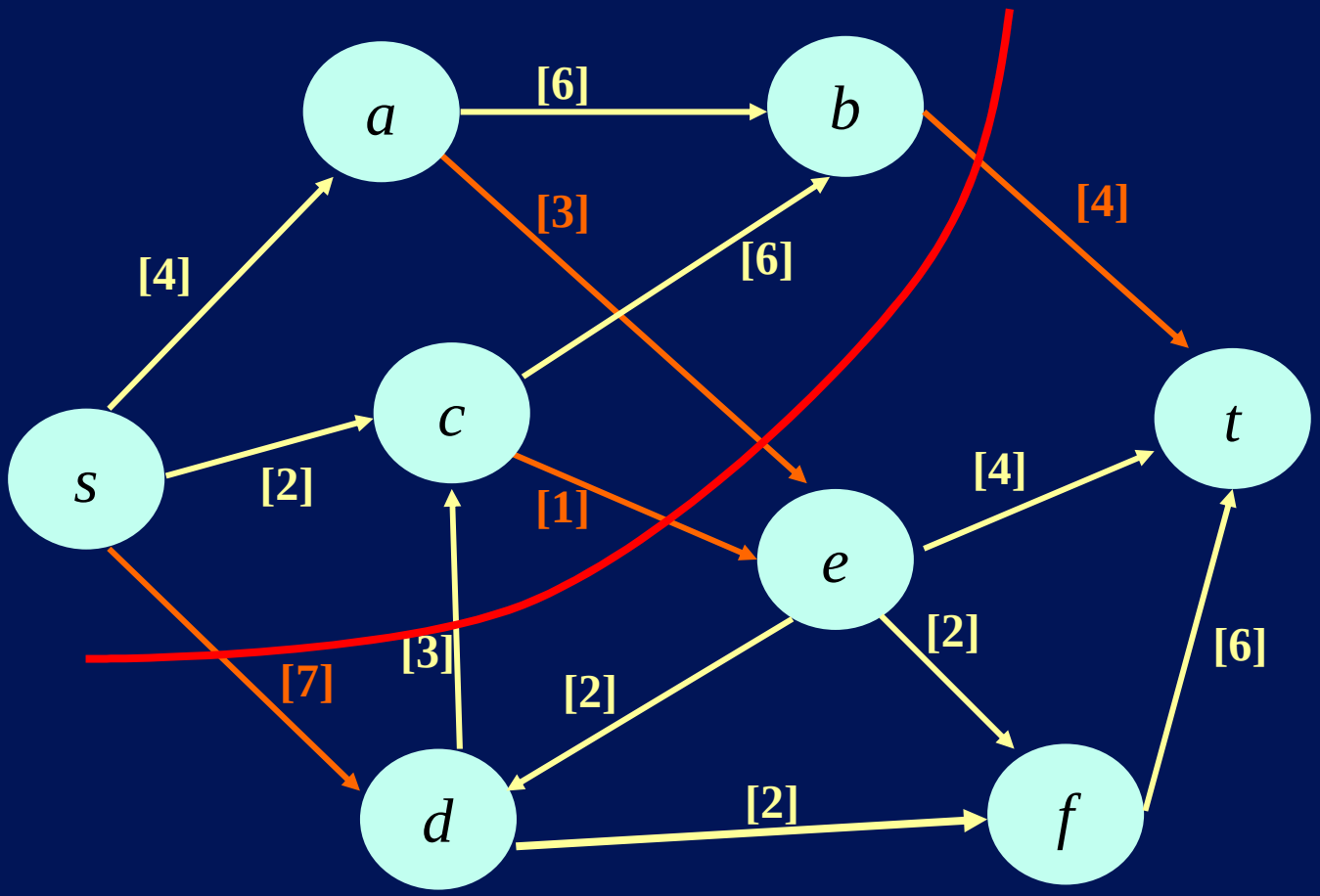
$\omega^+(T)$= {arcs sortant de T} = $\{(i, j) \in U$ t.q. $i \in T$ et $j \in S\}$

Par définition: $(t,s) \notin \omega^+(T)$

Soit (S, T) une coupe. La **capacité $c(S, T)$** de la coupe est définie par :

$$c(S,T) = \sum_{(i,j)\omega-(T)} c_{ij}$$

Coupe et réseau de transport



S={s,a,b,c}

T={t,d,e,f}

Exemple

Une coupe de valeur 15

$C=\omega^{-}(T)$={(b,t),(a,e),(c,e),(s,d)}

Flot / coupe

Propriété : Soit $R=(X,U)$ un réseau de transport.

$\forall\ \varphi$ flot admissible sur R, $\forall\ (S,T)$ coupe de R on a :

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

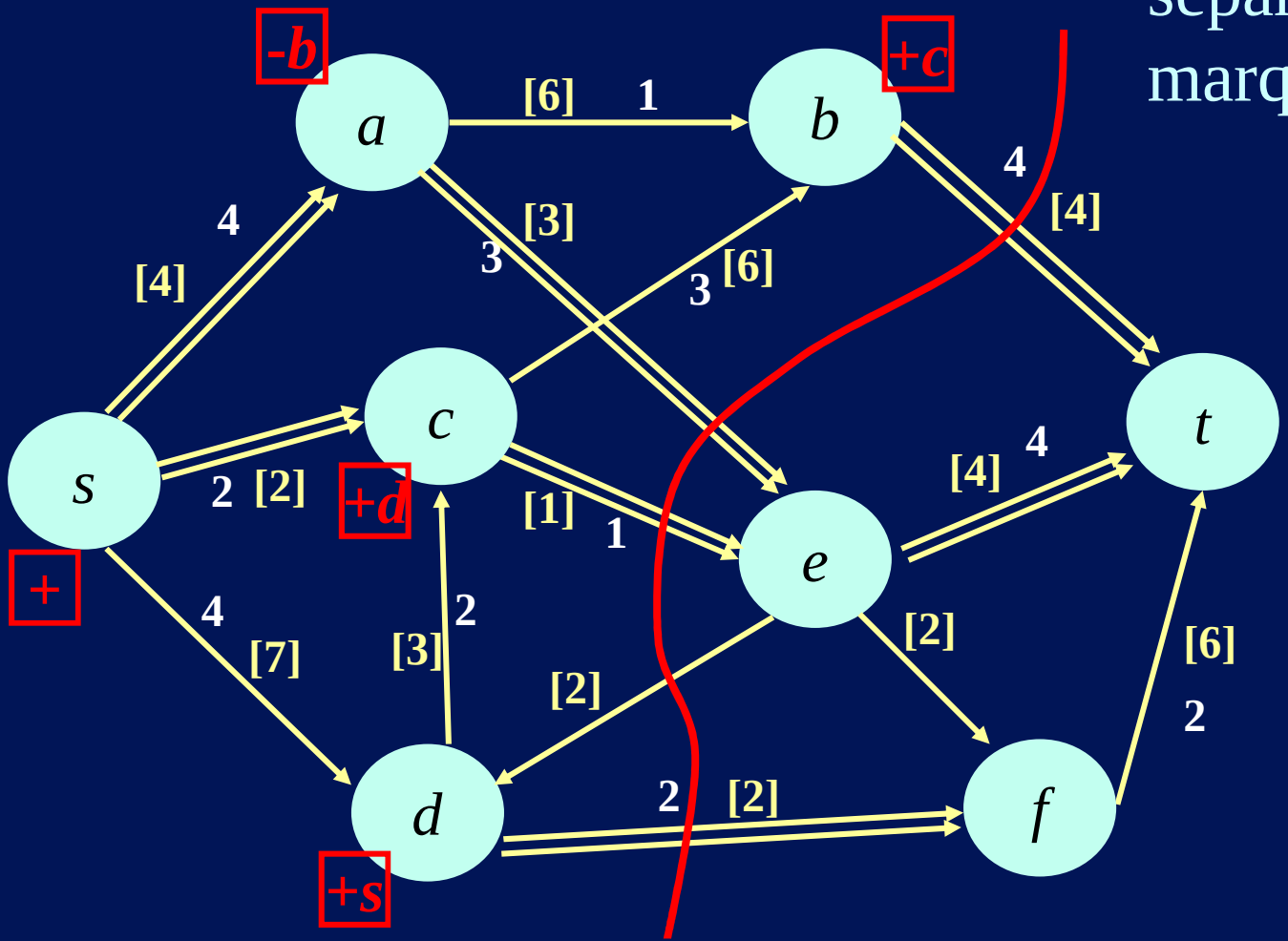
Preuve fondée sur la relation « flux ≤ capacité » et sur la loi de conservation des flux.

$\sum_{(i,j)\in \omega^-(T)} \varphi_{ij} = \sum_{(i,j)\in \omega^+(T)} \varphi_{ij} + \varphi_{ts}$ *et* $v(\varphi)=\varphi_{ts}$

$v(\varphi) = \sum_{(i,j)\in \omega^-(T)} \varphi_{ij} - \sum_{(i,j)\in \omega^+(T)} \varphi_{ij} \leq \sum_{(i,j)\in \omega^-(T)} c_{ij} = c(S,T)$

t non marqué: flot maximal

Coupe minimale : sépare les sommets marqués/non marqués



$S^*=\{s,a,b,c,d\}$

$T^*=\{t,e,f\}$

$C^*=\{(b,t),(a,e),(c,e),(d,f)\}$

$v(\varphi^*)=10=v(C^*)$

Théorème de Ford-Fulkerson

Théorème (Ford-Fulkerson, 1962) :

La valeur d'un flot maximal est égale à la capacité d'une coupe minimale.

Propriété CNS d'optimalité : un flot φ de s à t est maximal si et seulement s'il n'existe pas de chaîne améliorante de s à t.

Preuve du théorème et de l'algorithme de Ford-Fulkerson

Soit φ^* le flot obtenu par l'algorithme et soit S^* (resp. T^*) l'ensemble des sommets marqués (resp. non marqués) à la fin de l'algorithme. ($v(\varphi^*)=\varphi^*(t,s)$ et par définition: $(t,s)\notin \omega^+(T)$)

- Toute coupe (S,T) et tout flot φ vérifient: $v(\varphi) \leq c(S,T)$
- La loi de conservation des flux implique:

$$v(\varphi^*)= \sum_{(i,j)\in \omega^-(T)} \varphi^*_{ij} - \sum_{(i,j)\in \omega^+(T)} \varphi^*_{ij}$$

$$= \sum_{(i,j)\in \omega^-(T)} c^*_{ij} - 0 = c(S^*,T^*)$$

- Donc φ^* est un flot maximal et (S^*, T^*) est une coupe minimale.

Convergence de l'algorithme

Théorème des valeurs entières : Dans un réseau de transport à capacités entières, il existe un flot maximal dont tous les flux sont entiers.

Convergence de l'algorithme : Sous l'hypothèse des capacités entières, l'algorithme de Ford-Fulkerson converge en un nombre fini d'itérations :

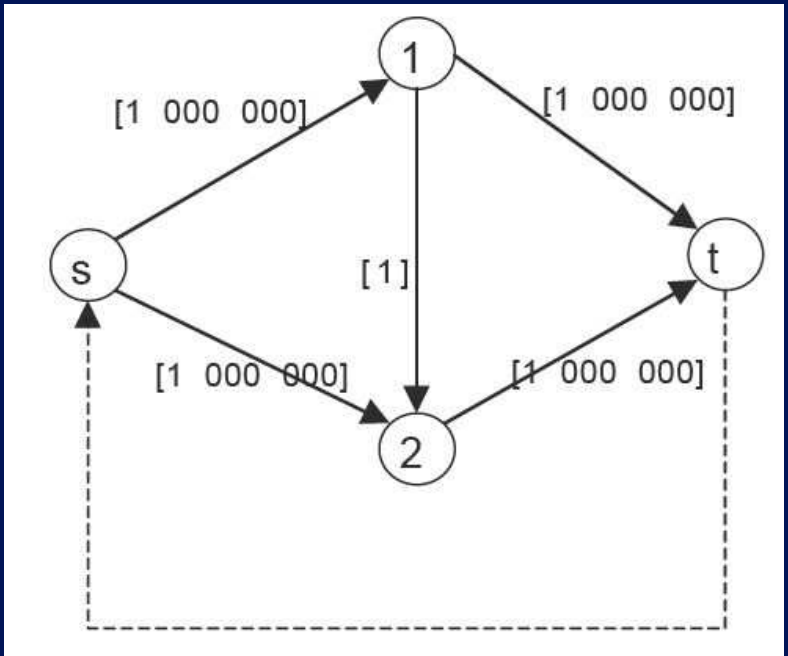
- La valeur du flot max est bornée par la capacité d'une coupe quelconque
- À chaque itération de l'algorithme, on augmente le flot d'une valeur $\alpha > 0$ et entière

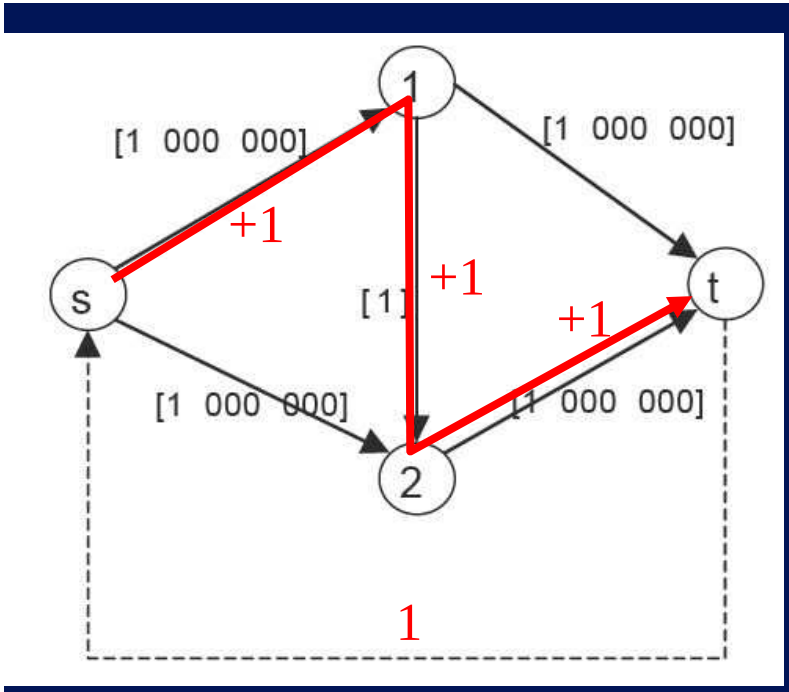
Complexité de l’algorithme (nombre d’itérations)

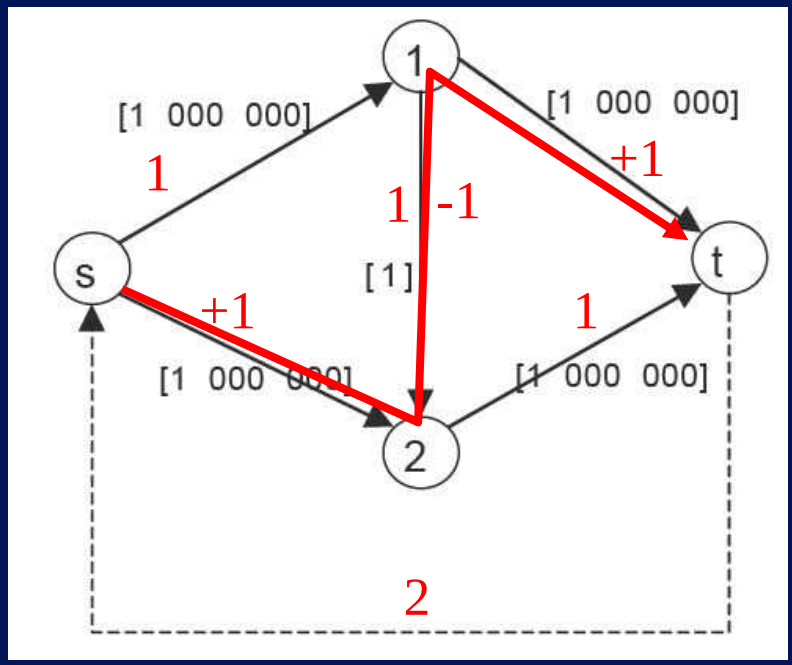
Si l’on ne prend pas garde au choix de la chaîne améliorante, on peut avoir un nombre d’itérations égal à la valeur du flot maximal.

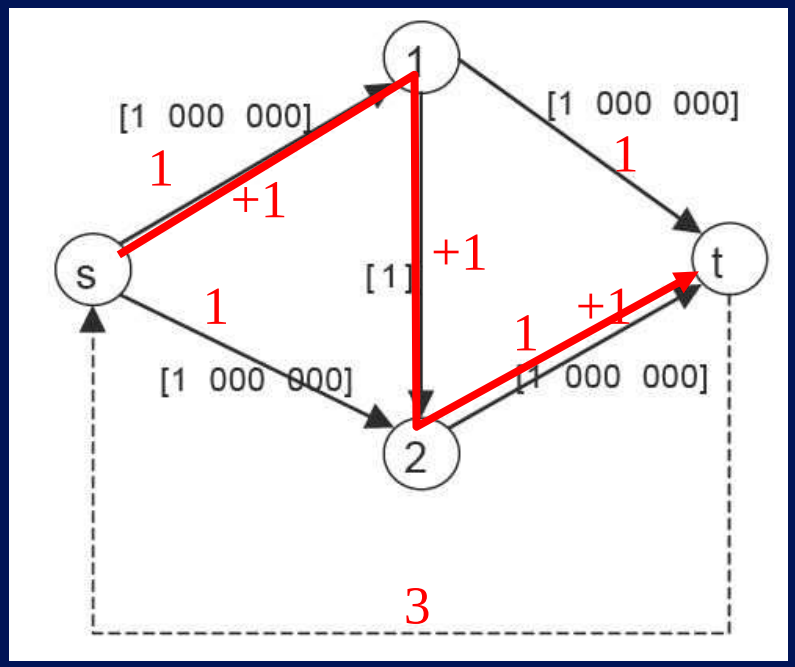
Complexité de l'algorithme (nombre d'itérations)

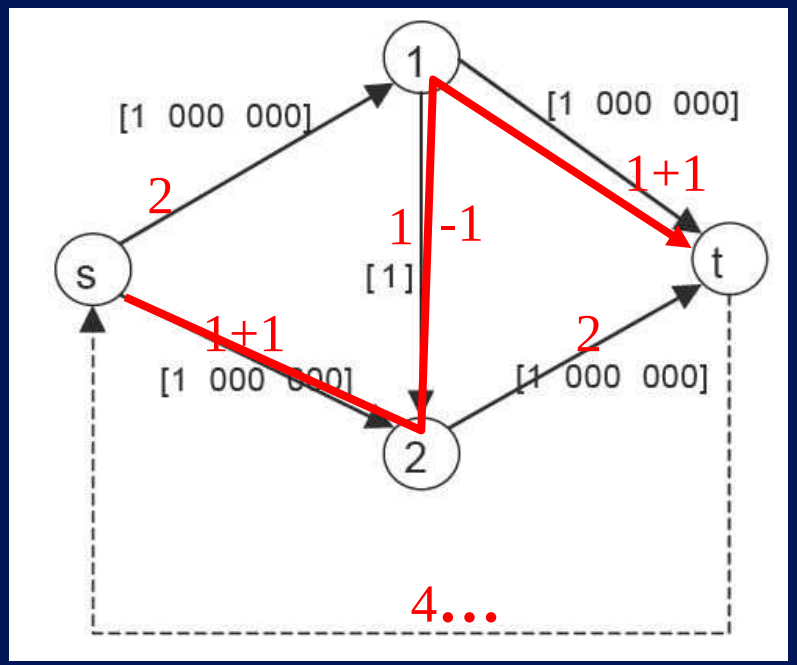
Si l'on ne prend pas garde au choix de la chaîne améliorante, on peut avoir un nombre d'itérations égal à la valeur du flot maximal.




1
[1 000 000]
[1 000 000]
+1
+1
[1]
+1
s
t
[1 000 000]
[1 000 000]
2
1


1
[1 000 000]
[1 000 000]
1
1
-1
+1
[1]
s
t
+1
1
[1 000 000]
[1 000 000]
2
2


1
[1 000 000]
[1 000 000]
1
+1
1
+1
[1]
s
t
1
1
+1
[1 000 000]
[1 000 000]
2
3


1
[1 000 000]
[1 000 000]
2
1+1
1
-1
[1]
s
t
1+1
2
[1 000 000]
[1 000 000]
2
4...

Complexité de l'algorithme (nombre d'«opérations »)

Théorème : Si chaque augmentation du flot est faite suivant une chaîne améliorante de longueur minimale, alors le flot maximal est obtenu après moins de $mn/2$ itérations, et au plus m essais de marquage à chaque itération, soit de l'ordre de $m^2n/2$ « opérations ».

(n=nombre de sommets ; m=nombre d'arcs)

Remarque : il existe des algorithmes plus efficaces.

5. Des modèles mathématiques

« Programmation linéaire »

$$\max \quad \varphi_{ts}$$

$$\varphi_{ij} \leq c_{ij} \qquad \forall (i,j) \in U \qquad (1)$$

$$\sum_{i \in \Gamma^-(j)} \varphi_{ij} - \sum_{i \in \Gamma^+(j)} \varphi_{ji} = 0 \qquad \forall j \in X \qquad (2)$$

$$\varphi_{ij} \geq 0 \qquad \forall (i,j) \in U$$

(1) : contraintes de capacités

(2) : Contraintes de conservation des flux

Programme linéaire :

$$(P) \quad \begin{array}{ll} \text{Max} & Ax \\ & Bx \leq C \\ & x \geq 0 \end{array}$$

A: vecteur n C: vecteur m B matrice m x n

L'ensemble des points admissibles (tels que $Bx \leq C$) est défini par un polyèdre. (on peut avoir aussi = ou $\geq$)

<u>Propriété</u>

Si (P) admet une solution finie, alors le programme admet un optimum en un point extrême du polyèdre.

Programme linéaire : (P)

$$\begin{aligned} \text{Max } & Ax \\ & Bx \leq C \\ & x \geq 0 \end{aligned}$$

L'optimum d'un programme linéaire en variables continues peut être obtenu en temps polynomial (fonction de m et n).

On peut résoudre un PL avec Excel et il existe des logiciels très performants, libres ou commercialisés mais gratuits pour les académiques, qui traitent des problèmes ayant des dizaines de milliers de variables et contraintes en quelques minutes.

Si les variables sont entières, les problèmes sont nettement plus difficiles à résoudre.

Flot maximal

Un modèle mathématique « Programme linéaire »

$$\max \quad \varphi_{ts}$$

$$\varphi_{ij} \leq c_{ij} \qquad \forall (i,j) \in U \qquad (1)$$

$$\sum_{i \in \Gamma^-(j)} \varphi_{ij} - \sum_{i \in \Gamma^+(j)} \varphi_{ji} = 0 \qquad \forall j \in X \qquad (2)$$

$$\varphi_{ij} \geq 0_{\,1} \qquad \forall (i,j) \in U$$

Ajout de variables d'écart fictives: transformation des inégalités en égalités.

Un modèle mathématique « Programme linéaire »

$$\max \quad \varphi_{ts}$$

$$\varphi_{ij} \leq c_{ij} \qquad \forall (i,j) \in U \qquad (1)$$

$$\sum_{i \in \Gamma^-(j)} \varphi_{ij} - \sum_{i \in \Gamma^+(j)} \varphi_{ji} = 0 \qquad \forall j \in X \qquad (2)$$

$$\varphi_{ij} \geq 0 \qquad \forall (i,j) \in U$$

$$\max \quad \varphi_{ts}$$

$$\varphi_{ij} + e_{ij} = c_{ij} \qquad \forall (i,j) \in U \qquad (1)$$

$$\sum_{i \in \Gamma^-(j)} \varphi_{ij} - \sum_{i \in \Gamma^+(j)} \varphi_{ji} = 0 \qquad \forall j \in X \qquad (2)$$

$$\varphi_{ij} \geq 0 \quad e_{ij} \geq 0 \qquad \forall (i,j) \in U$$

Programme linéaire : (P)

$$\begin{aligned} \text{Max } & Ax \\ & B \leq C \\ & x \geq 0 \end{aligned}$$

Propriété

Si (P) admet une solution finie et C est entier et B est totalement unimodulaire (TU) alors

(P) admet une solution optimale entière.

Définition: une matrice M est TU si tout déterminant d'une sous-matrice carrée de M vaut 0, 1 ou -1.

Programme de flot max:

$$\max \quad \varphi_{ts}$$

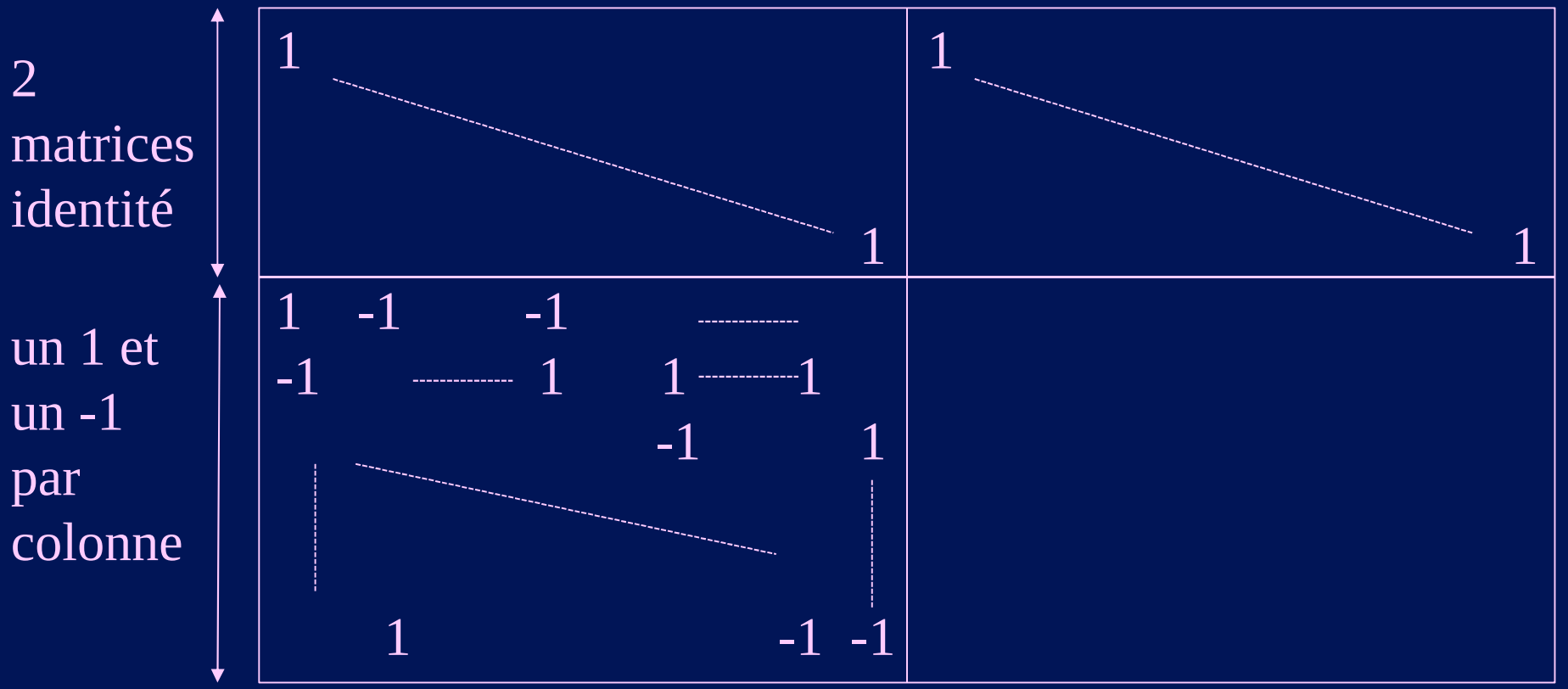
$$\varphi_{ij} + e_{ij} = c_{ij} \qquad \forall (i,j) \in U \qquad (1)$$

$$\sum_{i \in \Gamma^-(j)} \varphi_{ij} - \sum_{i \in \Gamma^+(j)} \varphi_{ji} = 0 \qquad \forall j \in X \qquad (2)$$

$$\varphi_{ij} \geq 0 \qquad e_{ij} \geq 0 \qquad \forall (i,j) \in U$$

Toute matrice de flot est totalement unimodulaire.

Conséquence: un logiciel de PL fournit une solution entière (si le problème admet une solution finie).



$$\max \quad \varphi_{ts}$$

$$\varphi_{ij} + e_{ij} = c_{ij} \qquad \forall (i,j) \in U \qquad (1)$$

$$\sum_{i \in \Gamma^-(j)} \varphi_{ij} - \sum_{i \in \Gamma^+(j)} \varphi_{ji} = 0 \qquad \forall j \in X \qquad (2)$$

$$\varphi_{ij} \geq 0 \quad e_{ij} \geq 0 \qquad \forall (i,j) \in U$$

B: 0, 1 ou -1

B est T.U.

Dualité flot/coupe

Deux nouveaux programmes mathématiques

Données: $G=(X,E)$

c_e, $e \in E$ capacité de l'arc e $(e=(i,j))$

Chemins de s à t numérotés de 1 à M (M peut être très grand)

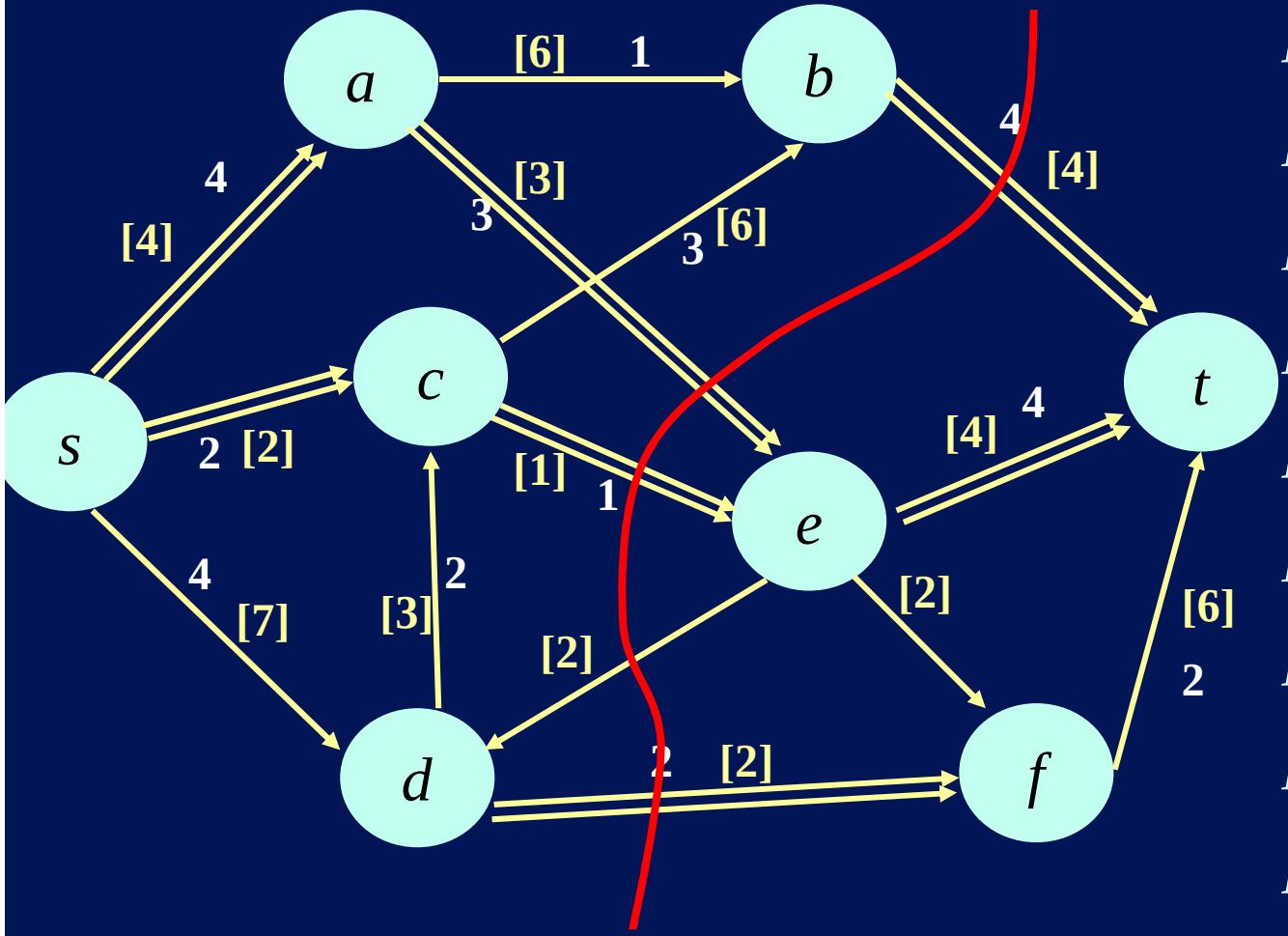
p_i , $i=1,\ldots,M$ chemin de s à t

Variables:

f_i , $i=1,\ldots,M$ $f_i \geq 0$ flot circulant sur le chemin p_i

y_e, $e \in E$ $y_e=1$ si l'arc e est coupé, $y_e=0$ sinon

Flot total: $v(\varphi) = \sum_{i=1,\ldots,M} f_i$



$y_{bt}=y_{ae}=y_{ce}=y_{df}=1$

$y_{ij}=0$ pour les autres arcs.

v(φ)=10

$p_1=(s,a,b,t)$ $f_1=1$

$p_2=(s,a,e,t)$ $f_2=3$

$p_3=(s,a,e,d,f,t)$ $f_3=0$

$p_4=(s,a,e,f,t)$ $f_4=0$

$p_5=(s,c,b,t)$ $f_5=2$

$p_6=(s,c,e,t)$ $f_6=0$

$p_7=(s,d,c,b,t)$ $f_7=1$

$p_8=(s,d,f,t)$ $f_8=2$

$p_9=(s,d,c,e,t)$ $f_3=1$

$p_{10}=(s,a,e,d,c,b,t)$ =

$p_{11}=(s,c,e,f,t)$ …

$f_{10}=f_{11}=\ldots=0$

Dualité flot/coupe

Deux nouveaux programmes mathématiques

- Flot

$$Max \quad \sum_{i=1}^{M} f_i$$

$$\sum_{\substack{i \text{ t.q.} \\ e \in p_i}} f_i \leq c_e \quad \forall e \quad (1_0)$$

$$f_i \geq 0 \quad \forall i = 1,.., M$$

Contraintes de capacités sur chaque arc

(flot sur l'arc = somme des flots sur les chemins qui passent par cet arc)

Remarque: si c_e=1, on cherche des chemins disjoints.

Dualité flot/coupe

Deux nouveaux programmes mathématiques

- Coupe

$$Min \sum_{e \in E} c_e \; y_e$$

$$\sum_{e \in p_i} y_e \geq 1 \quad \forall i \qquad (2_0)$$

$$y_e \in \{0,1\} \quad \forall e \in E$$

Contraintes de coupe

(Il faut au moins un arc coupé sur chaque chemin p_i de s à t, donc au moins un y_e qui vaut 1 sur chaque chemin p_i).

Dualité flot/coupe

Deux nouveaux programmes mathématiques

- Flot

$$Max \sum_{i=1}^{M} f_i$$

$$\sum_{i \text{ t.q. } e \in p_i} f_i \leq c_e \quad \forall e \quad (1_0)$$

$$f_i \geq 0 \quad \forall i = 1,..,M$$

- Coupe

$$Min \sum_{e \in E} c_e \; y_e$$

$$\sum_{e \in p_i} y_e \geq 1 \quad \forall i \quad (2_0)$$

$$y_e \in \{0,1\} \quad \forall e \in E$$

Deux programmes mathématiques « duaux »

matrice des coef. B $\quad$ ${}^tB$

- Flot

$$Max \sum_{i=1}^{M} f_i$$

$$\sum_{i \text{ t.q. } e \in p_i} f_i \leq c_e \quad \forall e \qquad (1_0)$$

$$f_i \geq 0 \quad \forall i = 1,.., M$$

- Coupe

$$Min \sum_{e \in E} c_e \; y_e$$

$$\sum_{e \in p_i} y_e \geq 1 \quad \forall i \qquad (2_0)$$

$$0 \leq y_e \leq 1 \quad \forall e \in E$$

Deux programmes mathématiques « duaux »

- Flot

$$Max \sum_{i=1}^{M} f_i$$

$$\sum_{i \text{ t.q. } e \in p_i} f_i \leq c_e \quad \forall e \quad (1_0)$$

$$f_i \geq 0 \quad \forall i = 1,.., M$$

- Coupe

$$Min \sum_{e \in E} c_e \; y_e$$

$$\sum_{e \in p_i} y_e \geq 1 \quad \forall i \quad (2_0)$$

$$0 \leq y_e \leq 1 \quad \forall e \in E$$

Deux PL

« duaux »

B TU $\Leftrightarrow$ ${}^tB$ TU

B TU $\Leftarrow$ ${}^tB$ Matrice de « chaîne »: TU

- Flot

$$Max \sum_{i=1}^{M} f_i$$

$$\sum_{\substack{i \text{ t.q.} \\ e \in p_i}} f_i \leq c_e \quad \forall e \quad (1_0)$$

$$f_i \geq 0 \quad \forall i = 1,.., M$$

- Coupe

$$Min \sum_{e \in E} c_e \; y_e$$

$$\sum_{e \in p_i} y_e \geq 1 \quad \forall i \quad (2_0)$$

$$0 \leq y_e \leq 1 \quad \forall e \in E$$

Même valeur optimale (entière) !!

6. Maximisation d'une fonction pseudobooléenne négative-positive

Méthode montrée sur un exemple mais généralisable à toute fonction de ce type (et même plus, aux fonctions sur-modulaires). $x_j \in \{0,1\}$ $x_j = 1 - \bar{x}_j$ $f: \{0,1\}^n \rightarrow Z$

Maximiser

$$f = -2x_1-3x_2-2x_3-3x_4+2x_1x_2+3x_1x_3+6x_3x_4$$

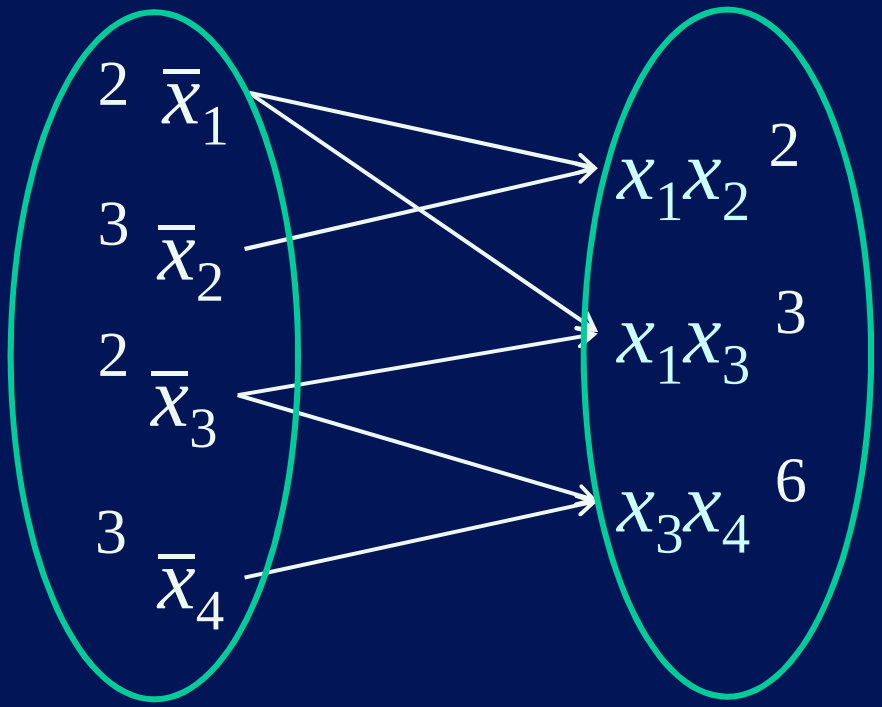
$$f = -10+2\,\bar{x}_1+3\,\bar{x}_2+2\,\bar{x}_3+3\,\bar{x}_4+2x_1x_2+3x_1x_3+6x_3x_4$$

Maximisation d'une fonction négative-positive

Graphe de conflit

$x_i = 0$ ou $1 \quad \forall i$

$$\max 2\,\bar{x}_1+3\,\bar{x}_2+2\,\bar{x}_3+3\,\bar{x}_4+2x_1x_2+3x_1x_3+6x_3x_4$$



Recherche d'un ensemble stable de poids maximal dans un graphe biparti

Stable: sous-ensemble de sommets non adjacents 2 à 2

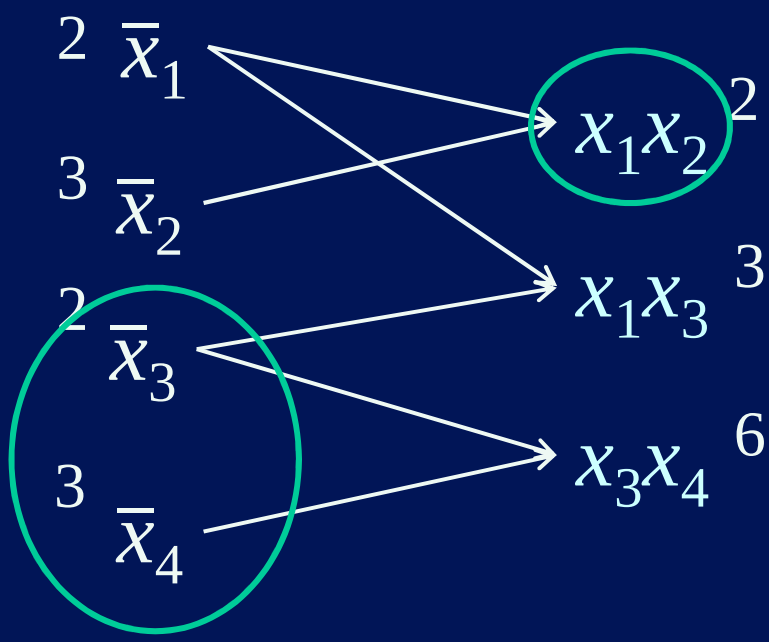
Partition des sommets en 2 stables

Maximisation d'une fonction négative-positive

Graphe de conflit

$x_i = 0$ ou $1 \quad \forall i$

$$\max 2\,\bar{x}_1+3\,\bar{x}_2+2\,\bar{x}_3+3\,\bar{x}_4+2x_1x_2+3x_1x_3+6x_3x_4$$



Recherche d'un ensemble stable de poids maximal dans un graphe biparti

Stable: sous-ensemble de sommets non adjacents 2 à 2

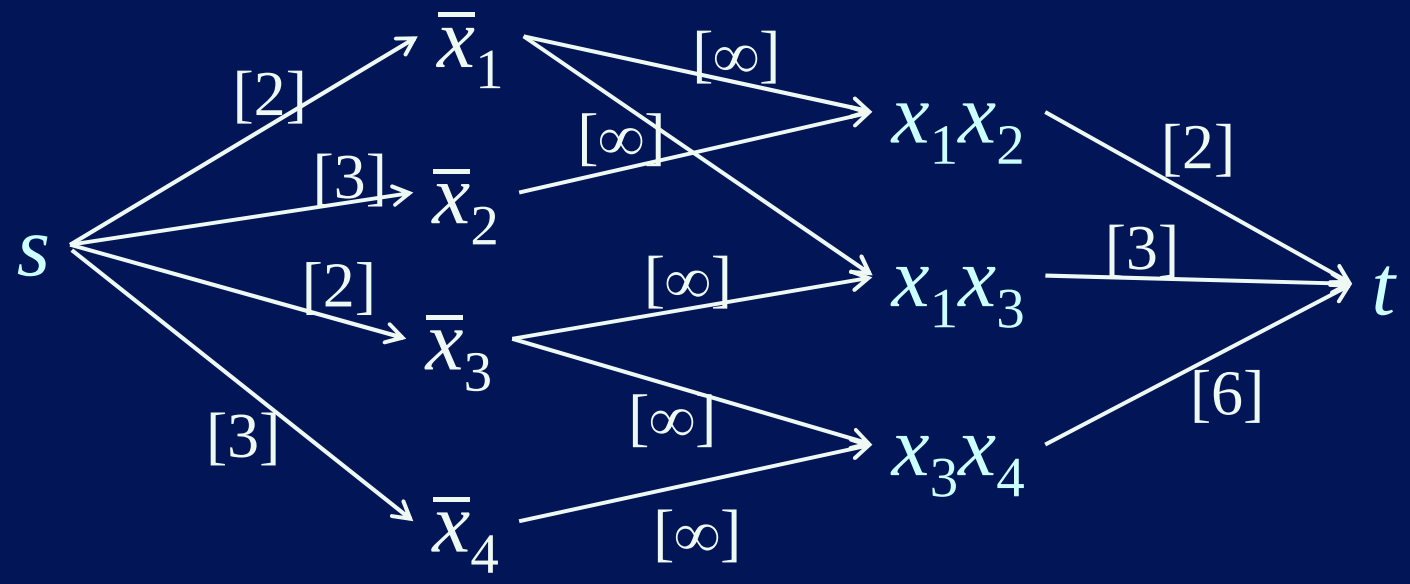
Exemple: $\bar{x}_3$, $\bar{x}_4$, x_1x_2 $\quad f=7$

Maximisation d'une fonction négative-positive

$$\max 2\,\bar{x}_1+3\,\bar{x}_2+2\,\bar{x}_3+3\,\bar{x}_4+2x_1x_2+3x_1x_3+6x_3x_4$$

Recherche d'un ensemble stable de poids maximal G biparti

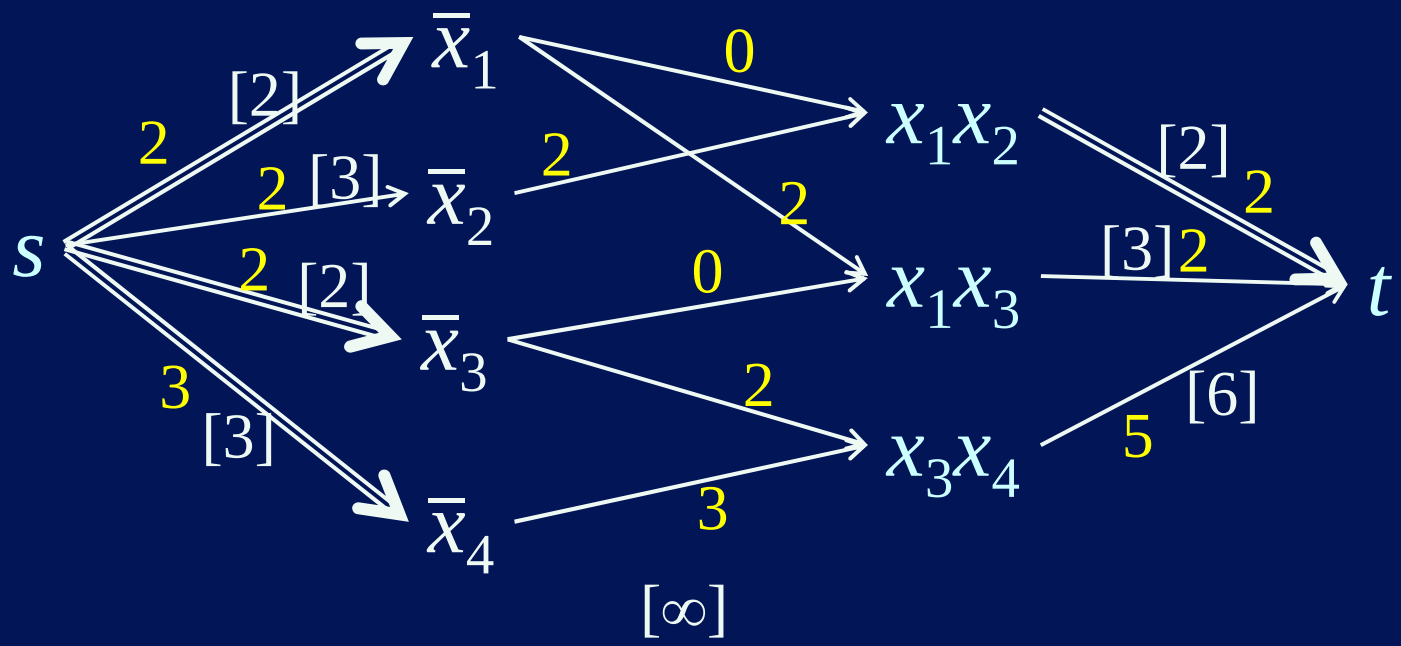
flot maximal



Maximisation d'une fonction négative-positive

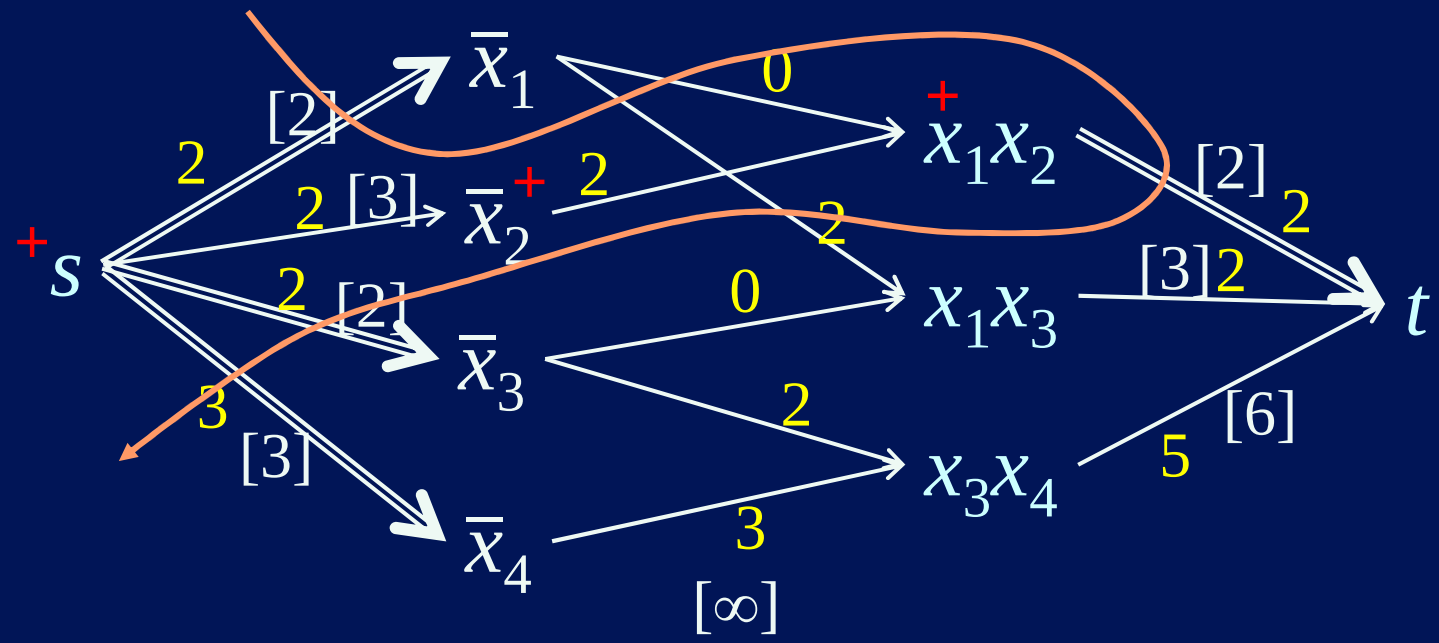
$$\max\ 2\,\bar{x}_1+3\,\bar{x}_2+2\,\bar{x}_3+3\,\bar{x}_4+2x_1x_2+3x_1x_3+6x_3x_4$$

flot maximal / coupe minimale



Maximisation d'une fonction négative-positive

$$\max\ 2\,\bar{x}_1+3\,\bar{x}_2+2\,\bar{x}_3+3\,\bar{x}_4+2x_1x_2+3x_1x_3+6x_3x_4$$



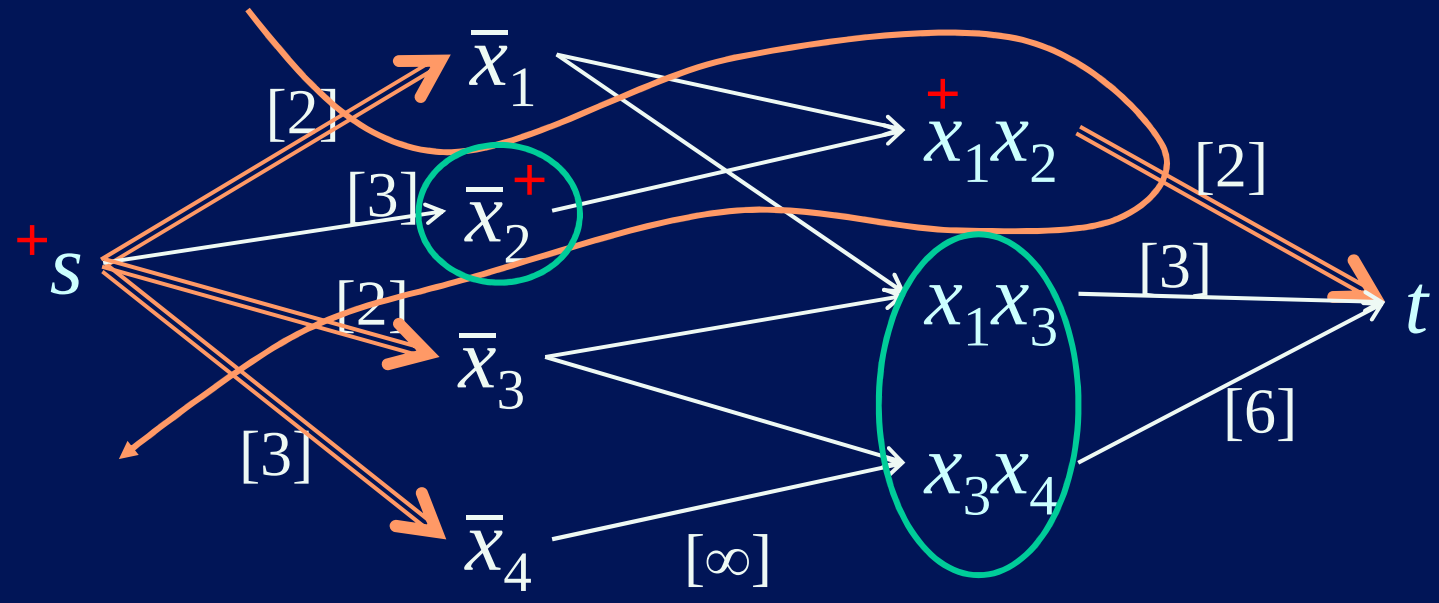
poids du stable $\{\bar{x}_2, x_1x_3, x_3x_4\}$ =

somme de tous les poids (arcs incidents *s* ou *t)* – valeur de la coupe

et coupe min ⇒ stable max

Maximisation d'une fonction négative-positive

$$\max\ 2\,\bar{x}_1+3\,\bar{x}_2+2\,\bar{x}_3+3\,\bar{x}_4+2x_1x_2+3x_1x_3+6x_3x_4$$



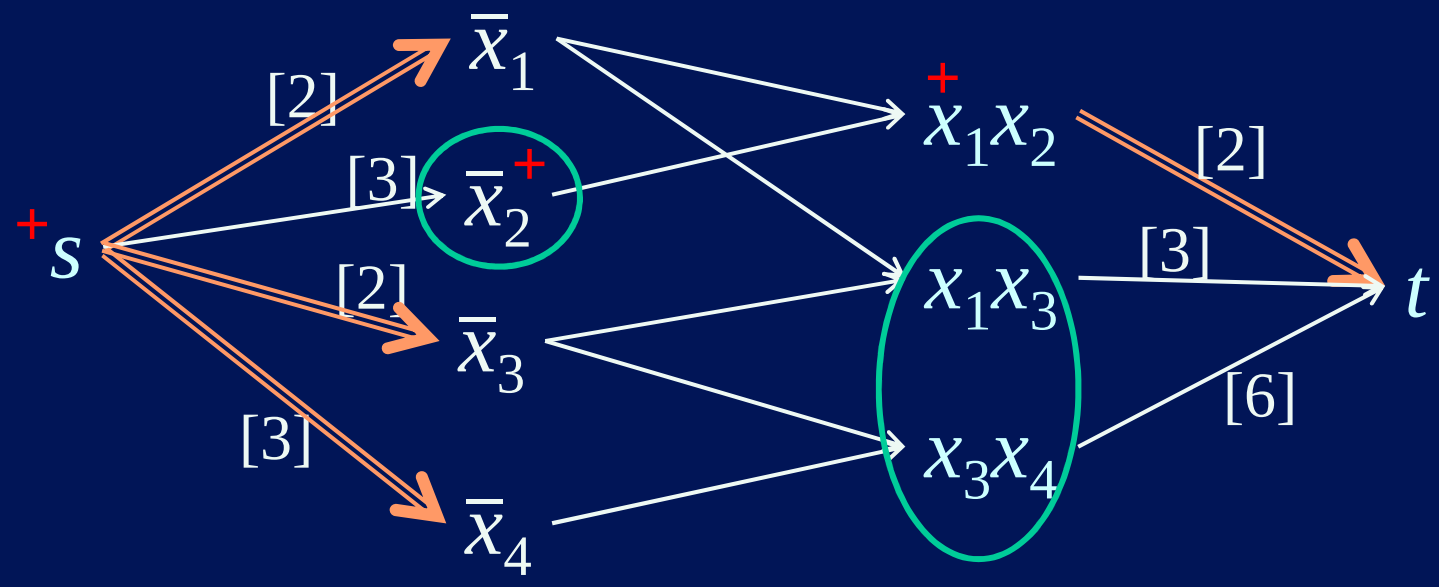
valeur de la coupe =

somme de tous les poids – poids du stable $\{\bar{x}_2, x_1x_3, x_3x_4\}$

et coupe min $\Rightarrow$ stable max

Maximisation d'une fonction négative-positive

$$\max 2\,\bar{x}_1+3\,\bar{x}_2+2\,\bar{x}_3+3\,\bar{x}_4+2x_1x_2+3x_1x_3+6x_3x_4$$

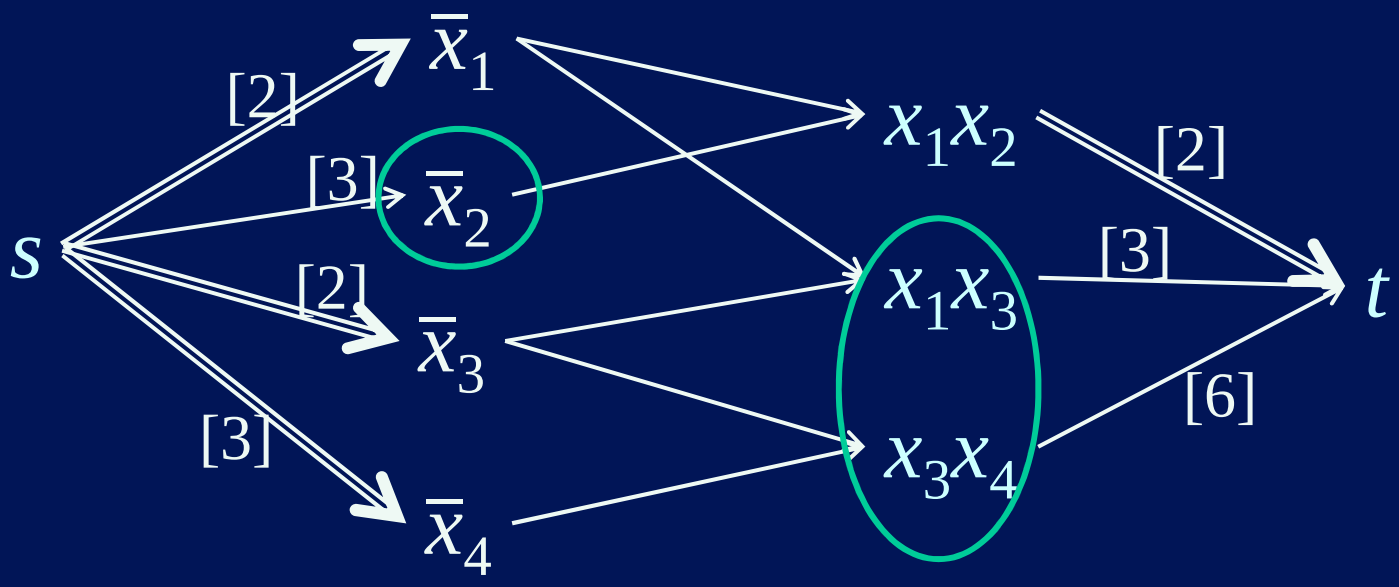


stable max $\{\bar{x}_2, x_1x_3, x_3x_4\}$ de poids 3+3+6=12

Solution: $f = 12$ avec $x_2 = 0$, $x_1 = x_3 = x_4 = 1$

Maximisation d'une fonction négative-positive

$$\max 2\,\bar{x}_1+3\,\bar{x}_2+2\,\bar{x}_3+3\,\bar{x}_4+2x_1x_2+3x_1x_3+6x_3x_4$$



stable max $\{\bar{x}_2, x_1x_3, x_3x_4\}$ de poids 3+3+6=12

Solution: $f=12$ avec $x_2 = 0$, $x_1 = x_3 = x_4 = 1$

Cas général de maximisation d'une fonction pseudobooléenne négative-positive

Une fonction pseudobooléenne est une fonction réelle de variables binaires qui peut s'écire:

$$f(x_1,x_2,\ldots,x_n)= \Sigma_{k=1,\ldots,p}\ b_k\ \ \Pi_{j\in N^k}\ x_j + K$$

$\forall k$ b_k réel, N_k indices des variables du $k^{\text{ième}}$ monome, K cste

Maximiser f: problème difficile en général

problème de flot si « négative-positive »

Maximisation d'une fonction négative-positive

Définition

Une fonction pseudobooléenne négative-positive s'écrit:

$$f(x_1,\ldots,x_n)= -\Sigma_{j=1,\ldots,n}\, c_j\, x_j+\Sigma_{i=1,\ldots,m}\, b_i \prod_{j\in N_i} x_j$$

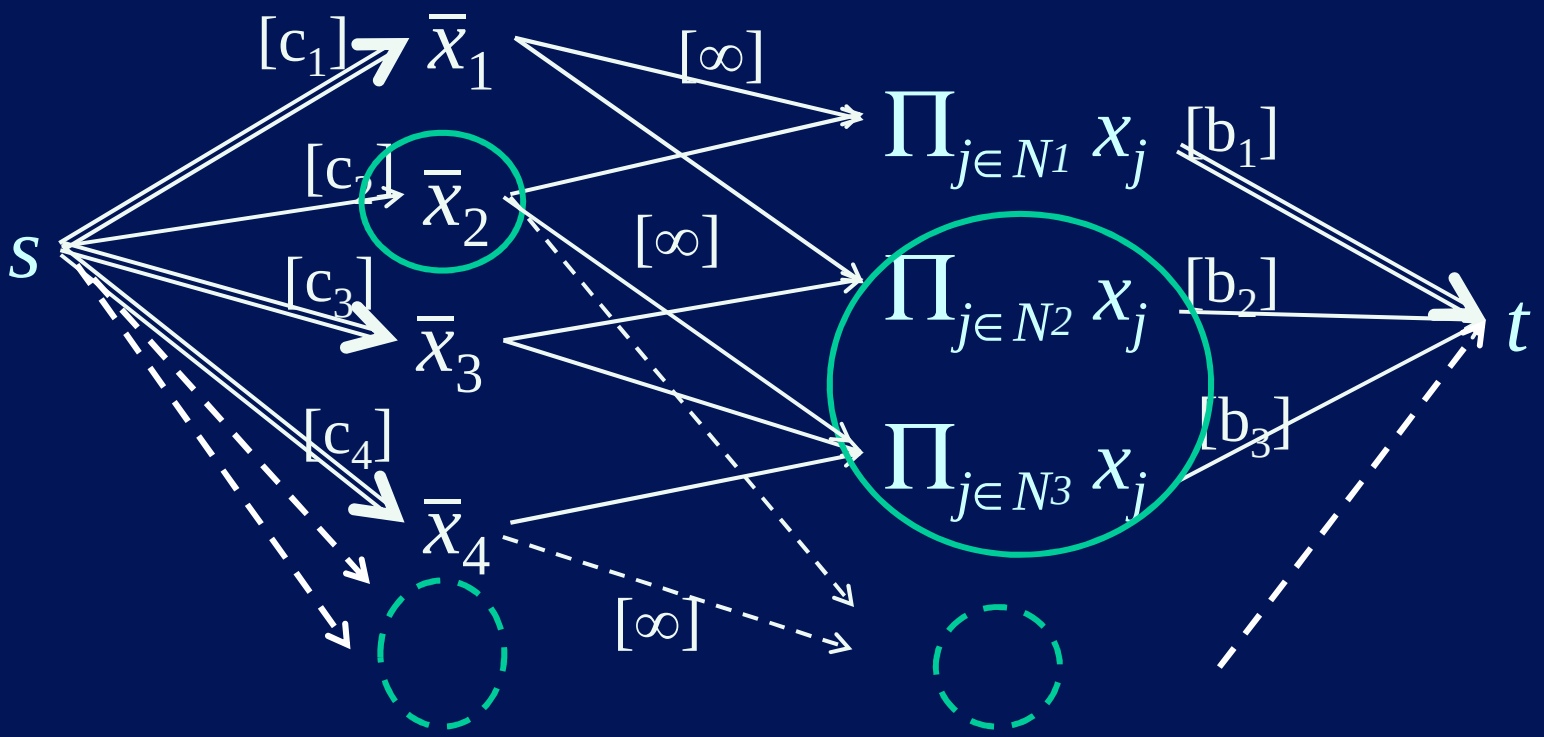
$c_j \geq 0 \;\forall\; j=1,\ldots,n$, $b_i \geq 0 \;\forall i=1,\ldots,m$

$$f = -\Sigma_{j=1,\ldots,n}\, c_j + \Sigma_{j=1,\ldots,n}\, c_j\, \bar{x}_j + \Sigma_{i=1,\ldots,m}\, b_i \prod_{j\in N_i} x_j$$

où $\bar{x}_j = 1 - x_j$

Maximisation d'une fonction négative-positive

$$\max \quad f = \sum_{j=1,\ldots,n} c_j \bar{x}_j + \sum_{i=1,\ldots,m} b_i \prod_{j \in N_i} x_j$$



Solution: stable de poids max obtenu en sélectionnant les sommets adjacents aux arcs non saturés incidents à *s* et *t*.

7. Pistes d'approfondissement

- Flot maximal de coût minimal (problème "facile")
- Flot avec multiplicateurs
- Multiflots et multicoupes (problèmes "difficiles")
- Matrice totalement unimodulaire et programmation linéaire en nombres entiers
- Programmation linéaire et dualité
- Recherche d'applications (réseaux de transport, de télécommunications...)

8. Petite bibliographie

- Recherche opérationnelle pour ingénieurs (Volume 1). D. de Werra, T. M. Liebling, J.-F. Hêche. 2003. Presses polytechniques universitaires romandes.
- Précis de recherche opérationnelle. R. Faure, B. Lemaire, C. Picouleau. 2009. Dunod.
- Optimisation combinatoire: Théorie et algorithmes. B. H. Korte, J. Vygen. 2010. Springer.
- Méthodes de planification en transport. Y. Nobert, R. Ouellet, R. Parent. 2005. Les presses de l'université de Montréal.
- Programmation linéaire avec Excel. C. Prins, M. Sevaux. 2011. Eyrolles.
- Optimisation discrète: De la modélisation à la résolution par des logiciels de programmation mathématique. A. Billionnet. 2007. Dunod.
- Graphes et algorithmes. M. Gondran et M. Minoux. 2009. Lavoisier.
- Une application : Optimizing the deployment of a multilevel optical FTTH network. M. Chardy, M.-C. Costa, A. Faye, M. Trampont. EJOR European Journal of Operational research, vol. 222(3), pp. 430--440, 2012.
- Un survey sur les multiflots: M.-C. Costa, L. Létocart , F. Roupin. Minimal multicut and maximal integer multiflow: a survey, EJOR European J. on Operations Research, vol. 162(1), pp. 55-69, 2005.
- et pour vous amuser: L'agrapheur: intrigues policières à saveur mathématique, Alain Hertz, 2010. Presses internationales Polytechnique.