Théorie des Graphes
Flots max

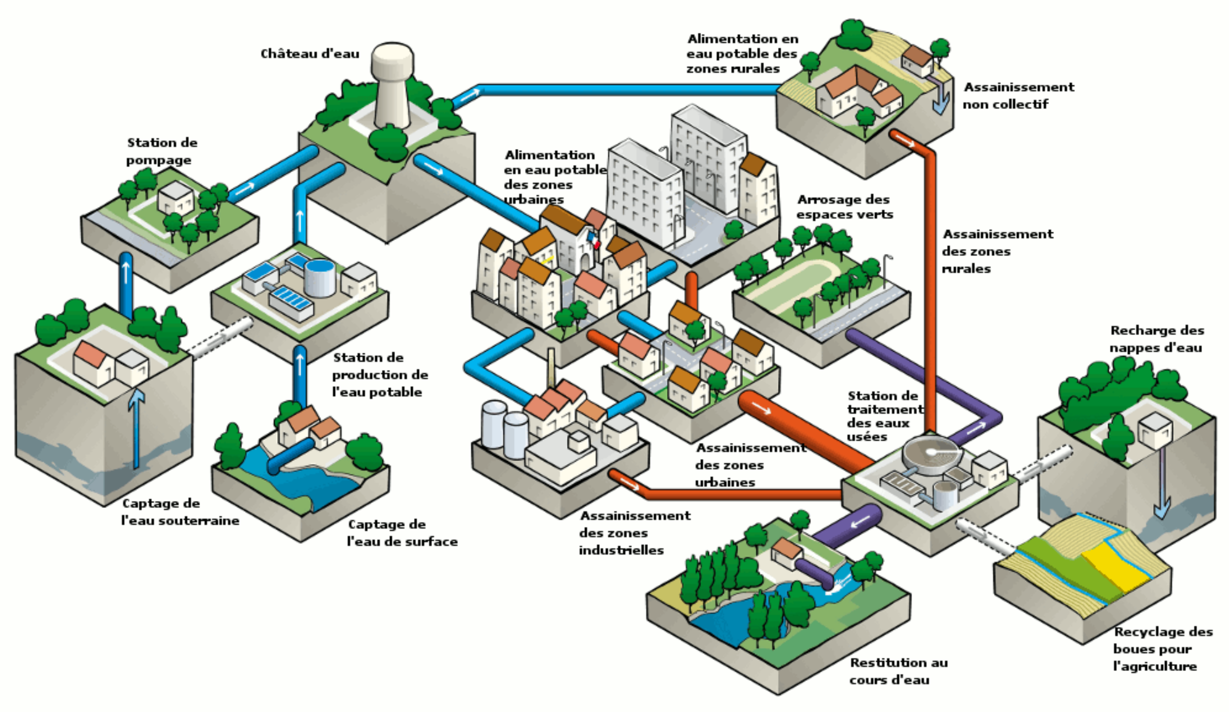
Fabrice Theoleyre

theoleyre@unistra.fr
http://www.theoleyre.eu

Maximisation de flot

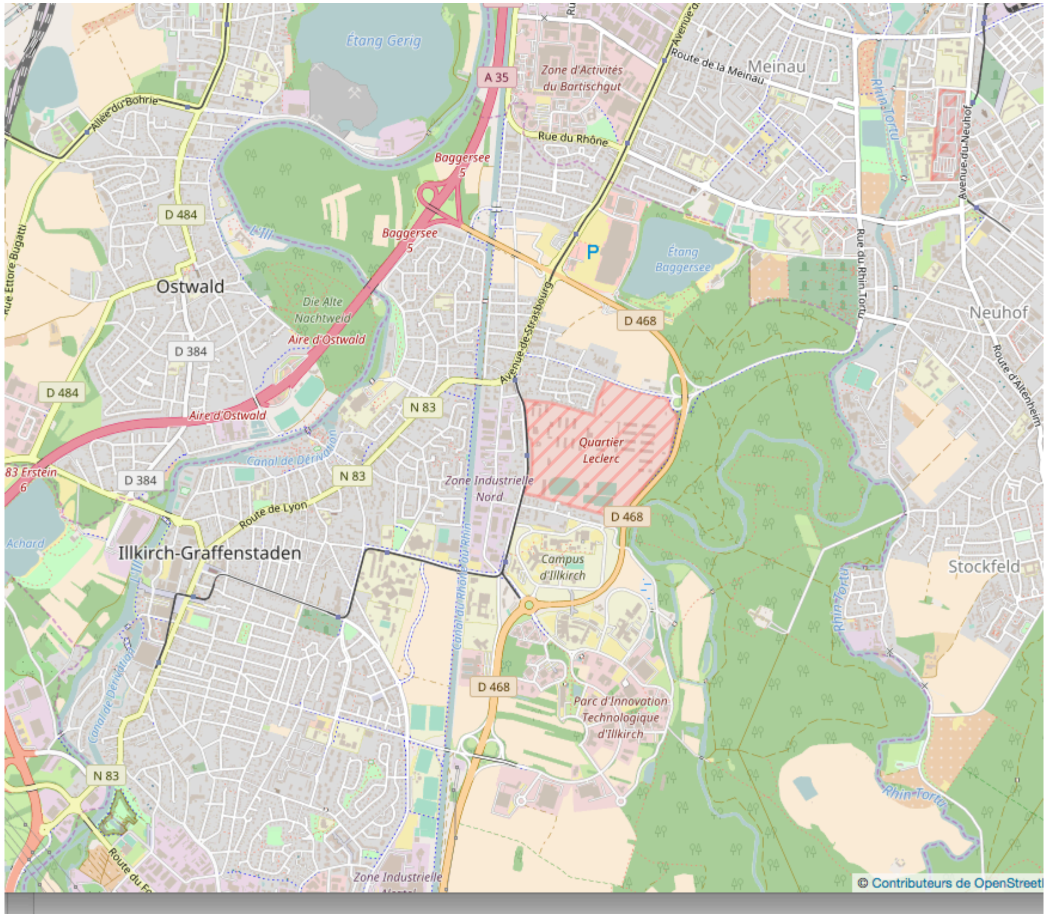
- Un problème clé en recherche opérationnelle
  - Réseau de distribution d’eau
    - ❖ Stations de pompage
    - ❖ Lieux de consommations
    - ❖ Canalisations (plus ou moins grosses)

Question : tout le monde aura-t-il bien de l’eau à la pression suffisante ?



Réseaux véhiculaires

- Une agglomération
  - Des rues / routes / autoroutes
    - ❖ Avec une vitesse et une capacité données
  - Un ensemble de véhicules



Question : comment dois je guider les véhicules pour éviter l'apparition de bouchons ?
Où se trouvent les points de mon réseau à améliorer ?

UNIVERSITÉ DE STRASBOURG

Internet

- Comment dois je router mes paquets dans le réseau ?
  - Un ensemble de paquets
    - ❖ De taille donnée
  - Des liens de communication
    - ❖ Radio, filaires (bande passante)

Réseaux de Transport & Flots

- Réseau de transport
  - il s'agit d'un quintuplet (S,A,s,p,c)
    - G=(S,A) est un graphe orienté
    - s,p $\in$ S respectivement l'unique source et l'unique puits de G
    - c : A → $\mathbb{R}^+ \cup \{+\infty\}$ soit un application définie sur les arcs de A nommée <u>capacité</u>
      - Pour tout arc a, c(a) représente le débit maximal pouvant transiter par a (autrement dit sa capacité).
  - Plusieurs sources?
    - Super-puits et super-source

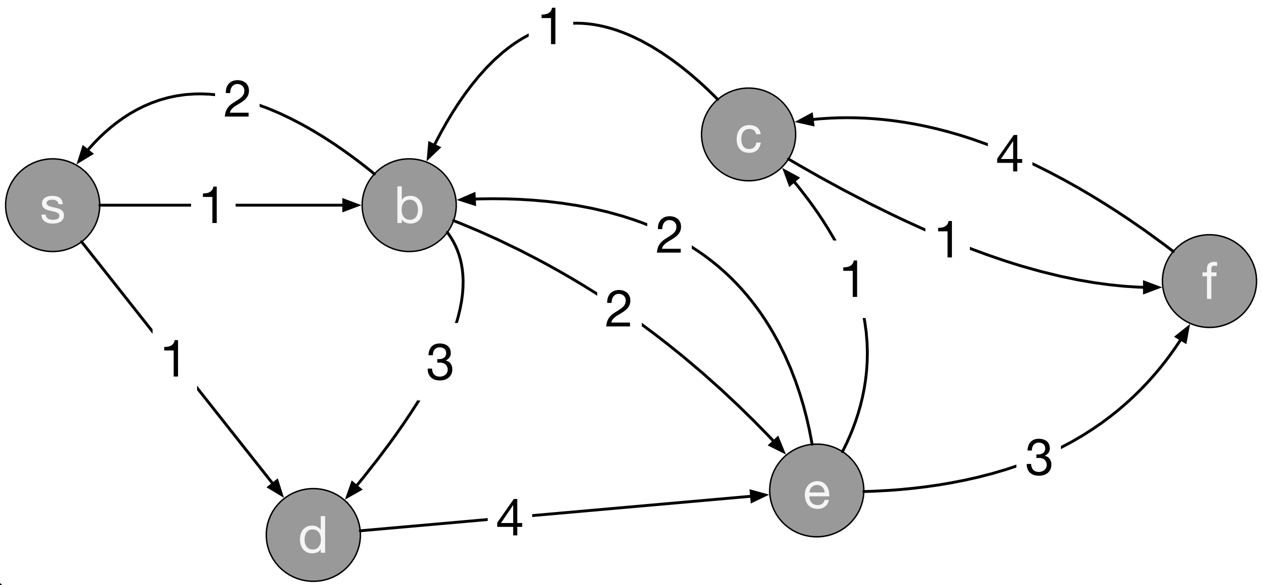
- Définition d'un flot dans un graphe
  - A chaque arc u, on associe un un nombre réel positif (ou nul) **f(u)**, dénoté ***flux***
    - $\forall u \in A, f(u) \in \mathbb{R}^+$
    - f : A → $\mathbb{R}^+$
  - Si on considère les sommets
    - $f^+(x) = \sum_{y \in \Gamma(x)} f((x,y))$
      - débit sortant
    - $f^-(x) = \sum_{y \in \Gamma(x)} f((y,x))$
      - débit entrant

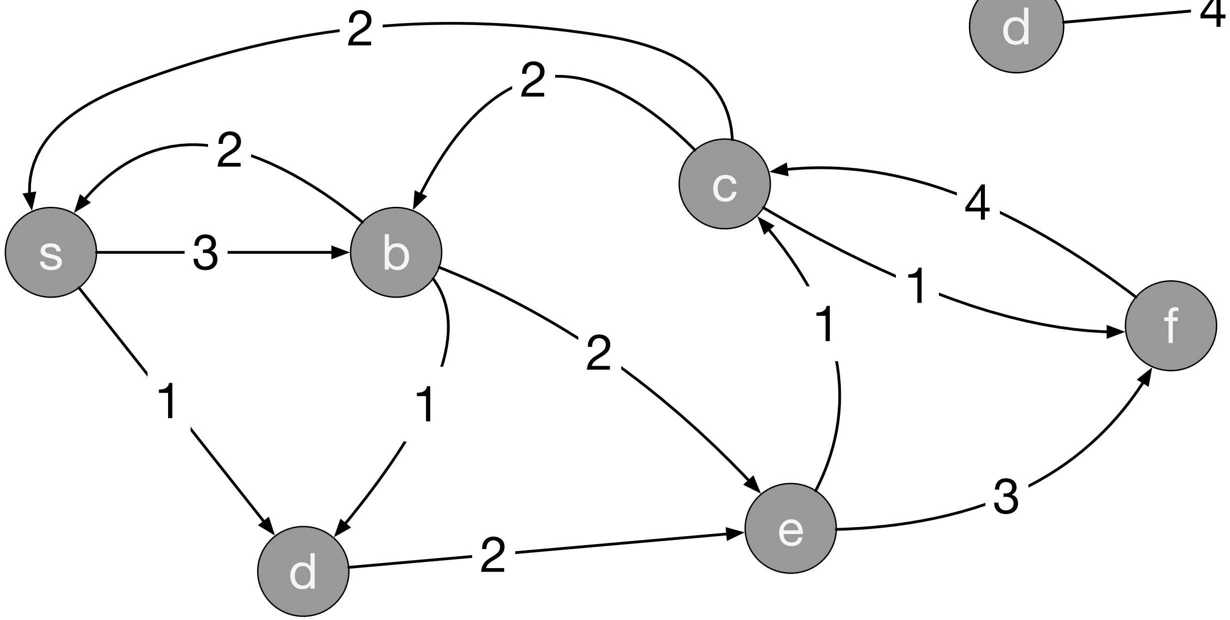
Equilibres

- Propriété d'équilibre global
  - les débits entrants compensent les débits sortants à une échelle globale
  - $\sum_{x \in S} f^{+}(x) = \sum_{x \in S} f^{-}(x)$
    - *rien ne se perd rien ne se crée, tout se consomme*

- Définition d'un flot
  - Propriété d'équilibre local ou loi de Kirchhoff
  - L'application f est appelé un **flot** si
    - $\forall\, x \in S,\ f^{+}(x) = f^{-}(x)$
  - on dit que f vérifie la propriété d'équilibre local

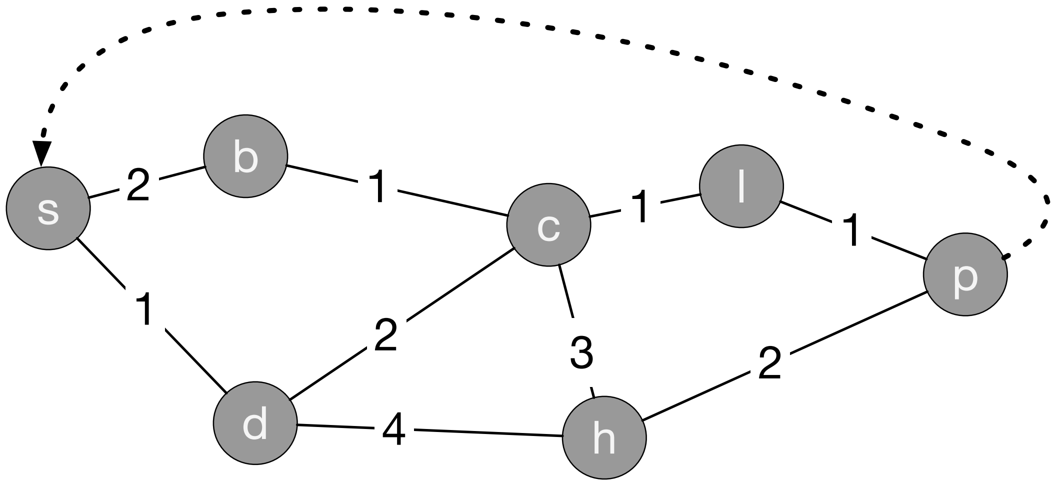
Application : est-ce un flot ?





Arc de retour

- Soit R un réseau de transport R= (S,A,s,p,c).
  - Arc retour (p,s)
    - ❖ R'= (S,A $\cup$ {(p,s)},s,p,c)
  - Par abus de langage, on continue à appeler R' un réseau de transport malgré le fait que s et p ne soit respectivement plus une source ni un puits.



- Soit R' un réseau de transport avec un arc de retour (p,s)
  On a alors $f((p,s))= f^+(s) = f^-(p)$.
  - Le flot sur l'arc retour est égal à la quantité qui transite sur le réseau de transport depuis la source s jusqu'au puits p.

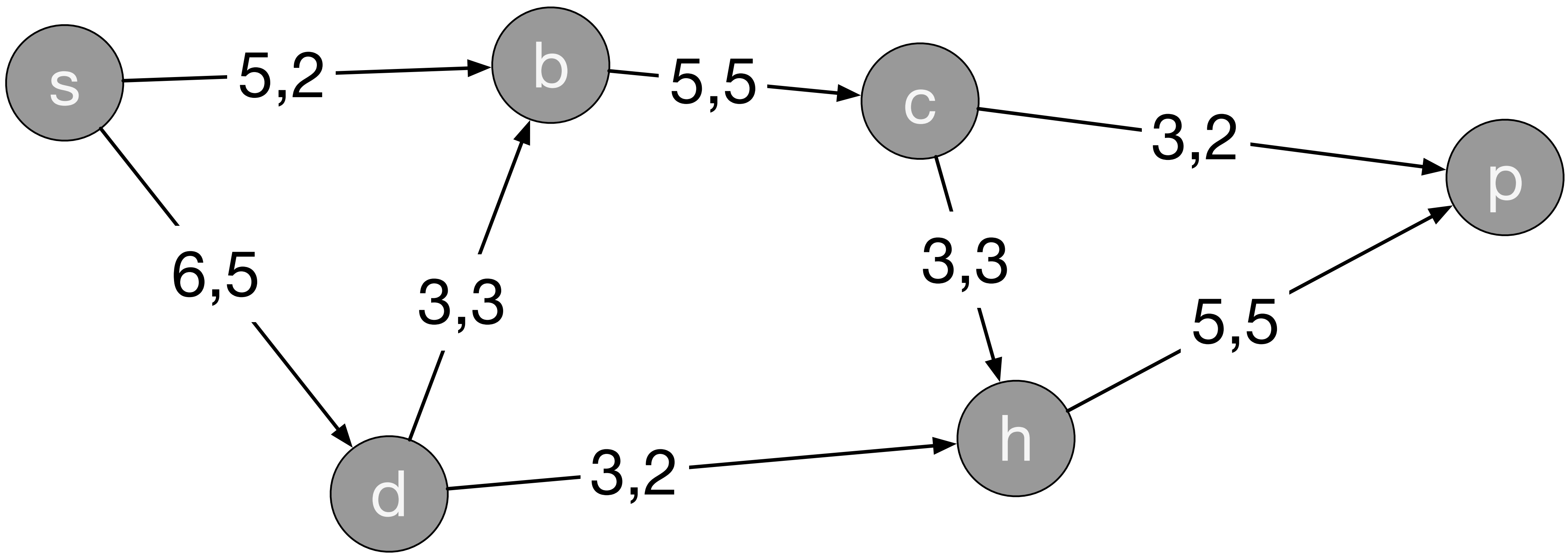
Flot compatible

- Un flot f est compatible avec un réseau de transport R si pour tout arc a appartenant à A, on a f(a) <= c(a)
  - La quantité de trafic est *admissible* dans le réseau
    - ❖ Le flot *respecte-t-il* les contraintes de capacité de l'arc ?
  - Remarque : le flot nul (i.e. f(a)=0 pour tout a) est compatible avec tout réseau de transport

- Flot compatible maximal sur un réseau de transport R
  - Un flot compatible f est maximal pour R si pour tout flot f' compatible, on a
    - ❖ $f'\big((p,s)\big) \leq f\big((p,s)\big)$
      - définition simplifiée via l'arc retour
      - On ne peut pas envoyer plus de trafic
    - ❖ équivalent à
      - $\sum_{a \in A} f'(a) \leq \sum_{a \in A} f(a)$

Flot compatible

c((y,x)), f((x,y))



Coupes et Flots

- Un réseau de transport R=(S,A,s,p,c)
  - Soit une coupe quelconque c
    - ❖ S un ensemble de sommets $\mathcal{S} \in S, p \notin S$
      - – Coupe = ensemble des arcs séparateurs
      - – $c = \{ (x, y) \in A \setminus x \in \mathcal{S}, y \notin \mathcal{S} \}$
    - ❖ On note $C(\mathcal{S}) = \sum_{u \in \omega_+(\mathcal{S})} c(u)$
      - – $\omega_+(\mathcal{S}) = \{ (x, y) \in A \mid x \in \mathcal{S}, y \notin \mathcal{S} \}$
      - – La somme des capacités des arcs sortants de c
      - – NB : si un arc possède une capacité infinie → la capacité de la coupe n'est pas non plus finie !

- Proposition
  - Quel que soit le flot compatible f
  - Quelle que soit la coupe c séparant s et p
  - $f\big((p, s)\big) \leq C(c)$

Coupes Min et Flots Max

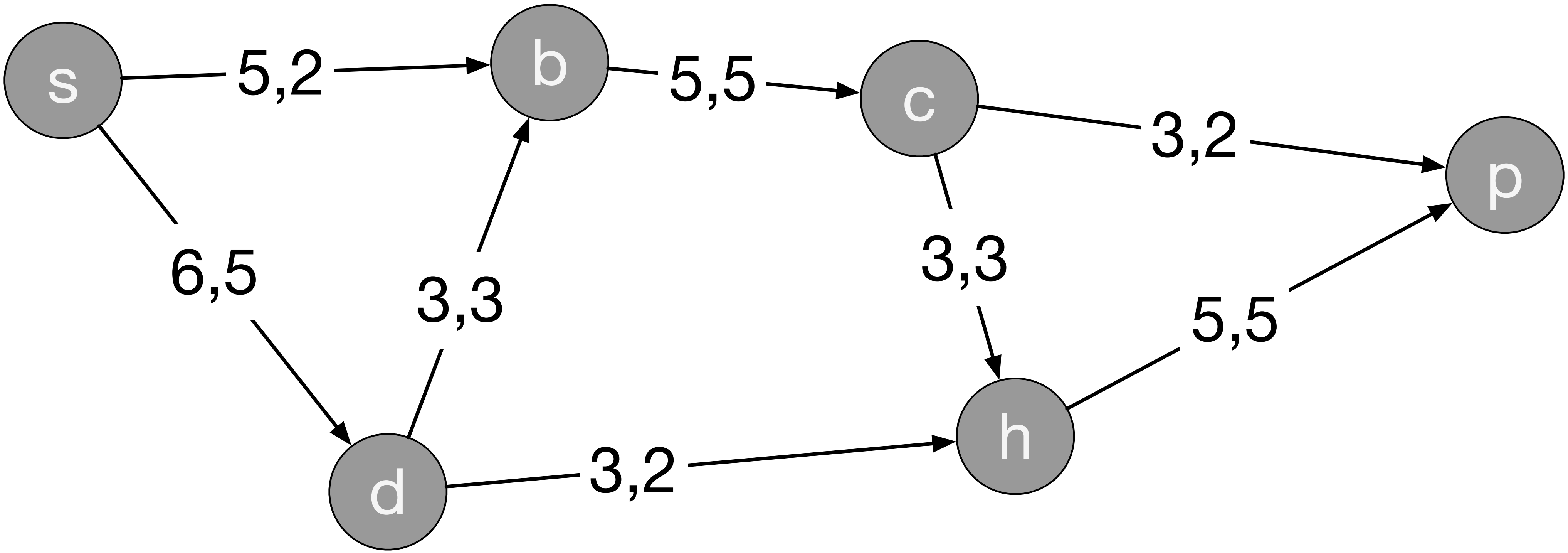
- Un réseau de transport R=(S,A,s,p,c)
  - Soit f* un flot compatible avec R
  - Soit $S^* \subset S$ une coupe de R
  - $f((s,p)) = \sum_{u \in \omega_+(S*)} c(u)$
  - Alors
    - F* est maximal
    - la coupe $\omega_+(S*)$ est minimale

- Conséquence
  - Les deux problèmes sont équivalents !

Exemple

- Quelles coupes ?

c((y,x)), f((x,y))



Arcs saturés, réseaux complets et valeur résiduelle

- Un arc a est dit saturé si f(a)=c(a)
  - Le flot f est dit complet si tout chemin de s à p passe par (au moins) un arc saturé
- Soit un chemin C = $(x_i)^l_{i=0}$ de s à p dans R.
  - l la longueur du chemin
  - On note r(C) la valeur résiduelle de C
  - $r(C) = \min\{c((x_i, x_{i+1})) - f((x_i, x_{i+1}))\}_{i=0}^{l-1}$

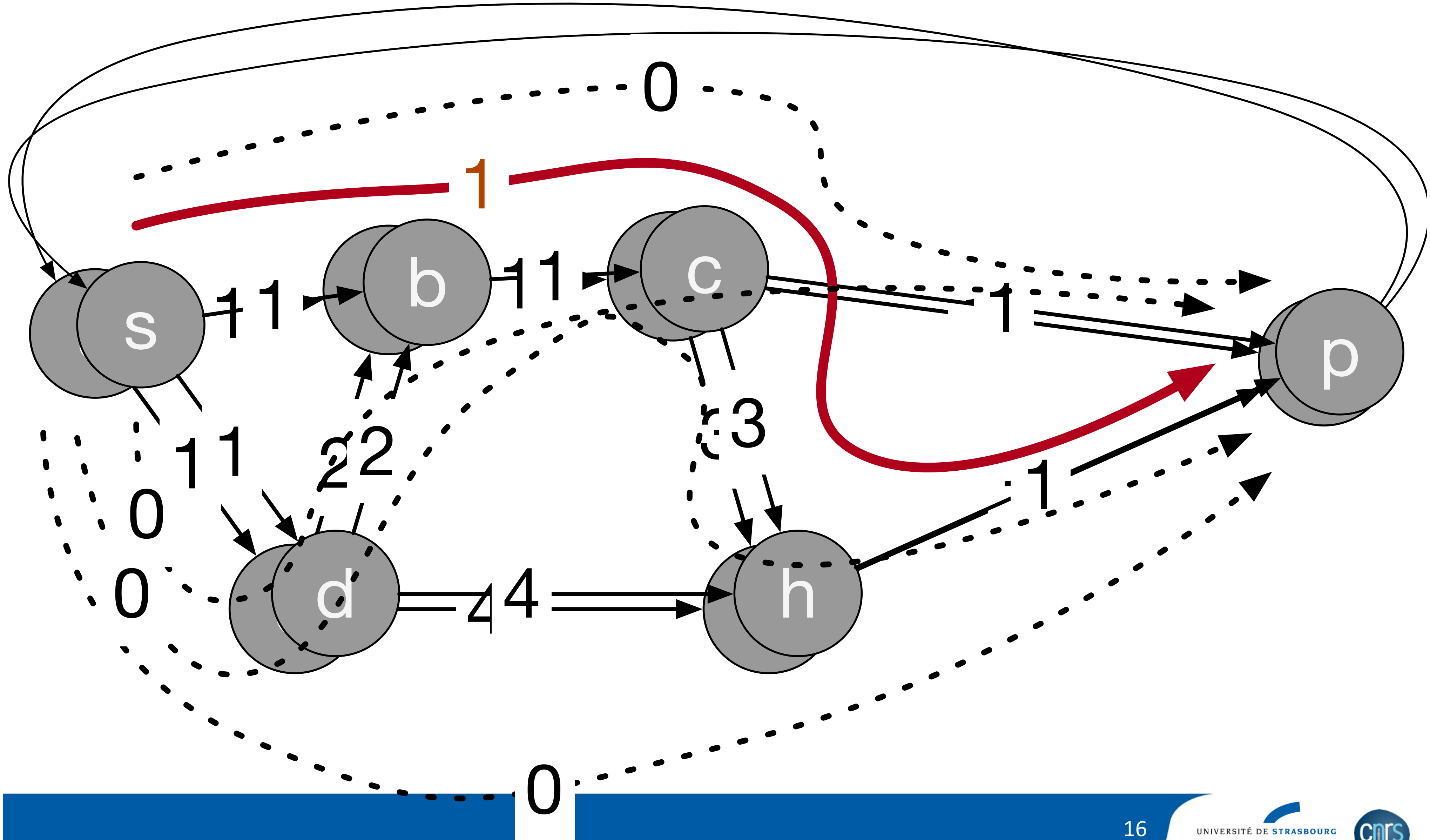
- Soit f un flot compatible avec R
  - S'il existe un chemin C = $(x_i)^l_{i=0}$ de s à p dans R tel que r(C)=+∞
    - ❖ ou de manière équivalente t.q. $\forall\ 0 \leq i \leq l-1, c((x_i, x_{i+1})) = +\infty$
    - ❖ alors R n'admet pas de flot compatible maximal.
  - ➔ pré-condition pour l'existence d'un flot compatible maximal
    - ❖ Vérifier qu'il n'existe pas un chemin "de capacité infini" de s à p

Recherche de flots compatibles max

- Soit un réseau de transport R
  - son arc de retour (p,s) de capacité infinie
  - un flot compatible avec R
  - S'il existe un chemin C tel que r(C) > 0 alors il est possible de définir un flot f' compatible avec R de valeur f'((p,s)) > f((p,s)) en posant f'(a)=f(a)+r(C) pour tout a dans C ∪ (p,s) (f'(a)=f(a) pour tout autre arc de R).
  - Tant qu'on arrive à trouver des chemins de s à p dont la valuer résiduelle est non nulle, on peut maximiser le flot jusqu'à obtention du réseau complet.
    - ➔ base des algorithmes recherchant un flot maximum
    - Il suffit de trouver un tel chemin, et de lui ajouter du trafic

- Si un flot f est compatible et maximal alors f est complet.
  - Néanmoins f peut être complet sans être maximal
    - Tout chemin passe par un arc saturé
    - Exercice : représentez un tel cas.
    - Conclusion ?

Complet mais pas maximal?



Réseaux d'écart et chemins améliorant

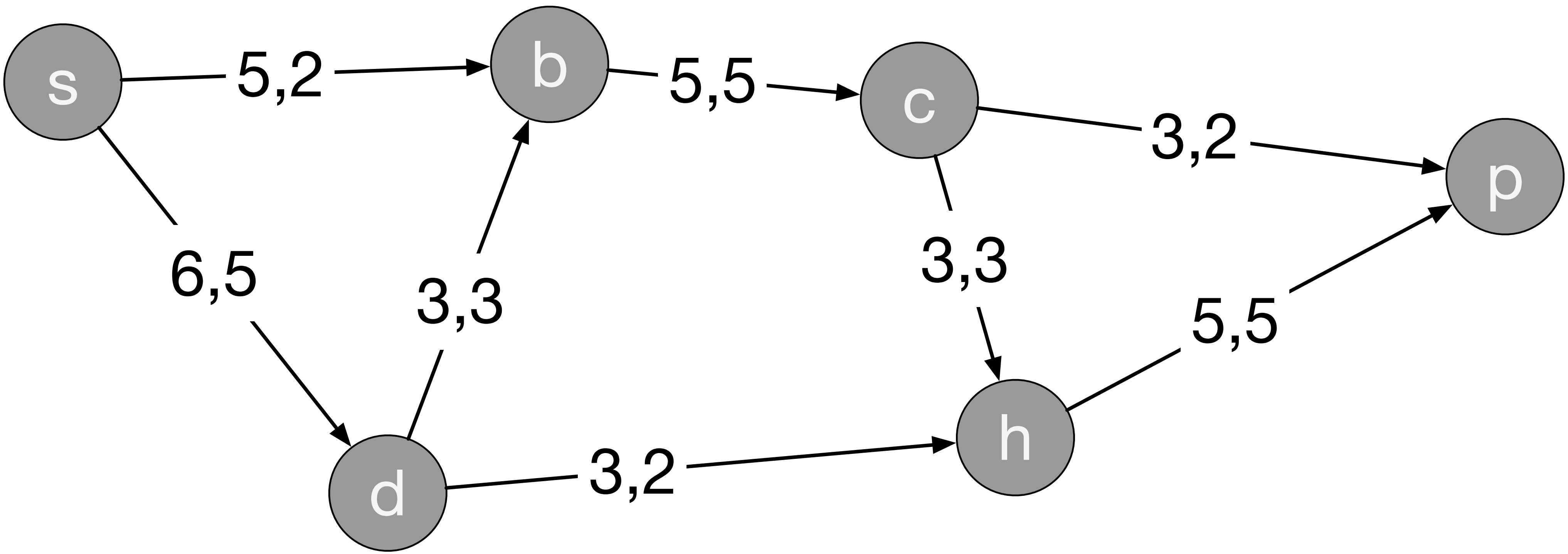
- Un réseau d'écart est défini par R' = (S,A',c') dans A :
  - Associé au réseau de transport R=(S,A,s,p,c)

  - Soit un flot compatible f
  - Pour tout arc a = (x,y) on crée dans le graphe d'écart
    - ❖ arc avant $a^+ = (x,y)$
      - – de capacité $c'(a^+) = c(a) - f(a)$
    - ❖ arc rétrograde $a^- = (y,x)$
      - – de capacité $c'(a^-) = f(a)$ $a^+ = (x,y)$

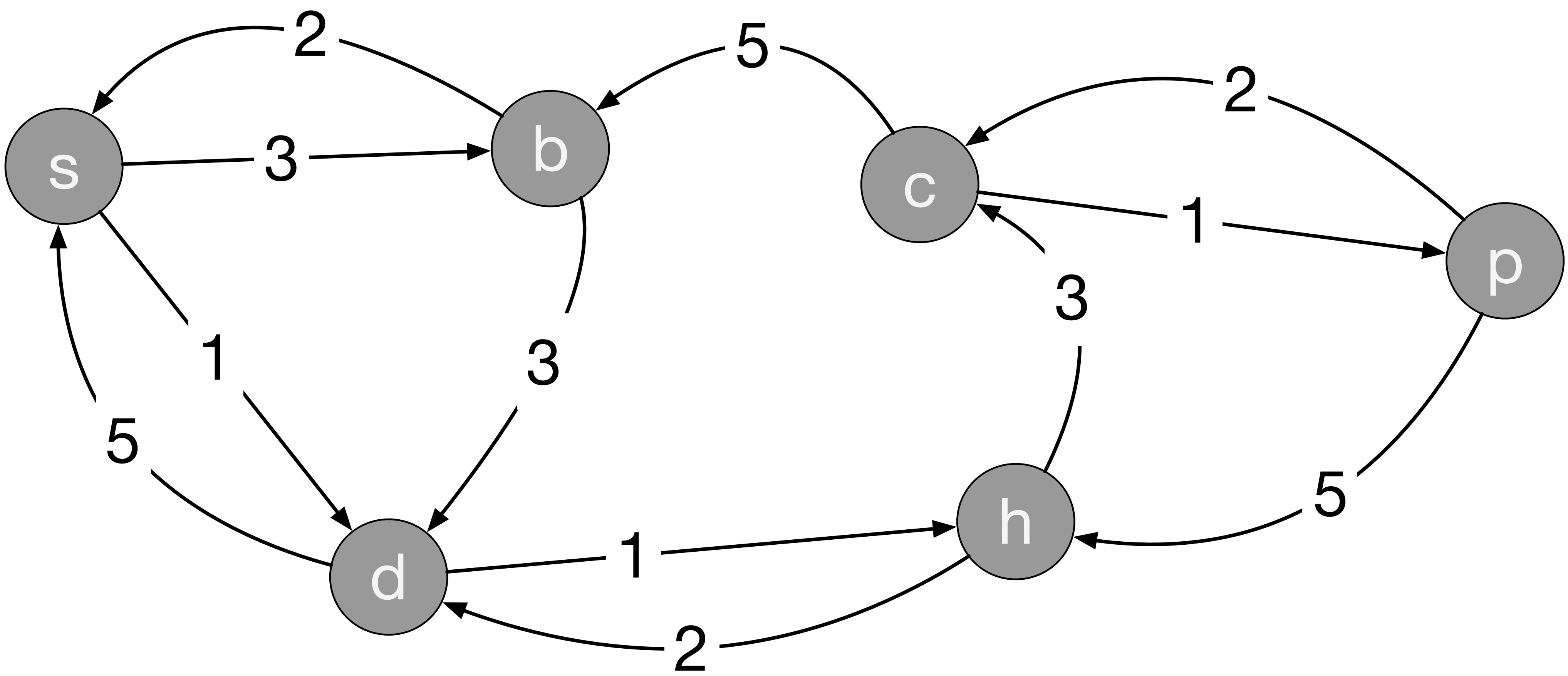
- Chemin améliorant
  - Chemin C de s à p dans le réseau d'écart R'
  - La capacité résiduelle de C, noté r'(C) se définit par
    - ❖ $\min\{c'((x_i, x_{i+1}\}_{i=1}^{l-1}$

Application

c((y,x)), f((x,y))



Application



- Intérêt ?

Remarque : c' est bien différent de c

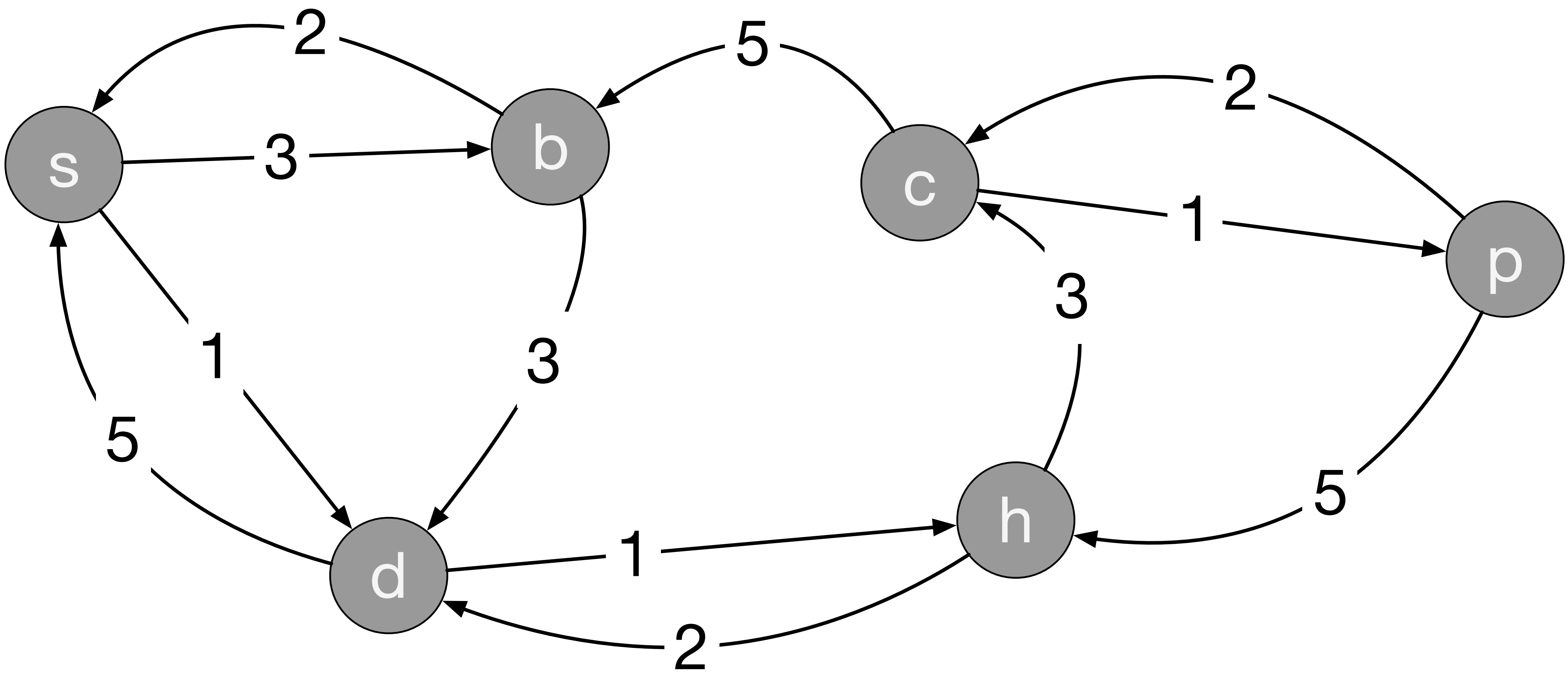
Recherche de flots compatibles max

- Corollaire : il existe un chemin améliorant de s à p dans le réseau d’écart R’_ssi f n’est pas un flot maximal pour R.
  - chemin améliorant

- Soit un chemin améliorant C = $(x_i)^l_{i=0}$ de capacité résiduelle $r'(C)=\varepsilon \in \mathbb{R}^{+*}$
  - Soit f’: A -> $\mathbb{R}^+$ le flot défini de telle sorte que
  - pour tout a dans A, on a :
    - ❖ f’(a) = f(a) + ε si a est un arc avant sur le chemin C
    - ❖ f’(a) = f(a) - ε si a est un arc rétrograde sur le chemin C
  - f’((p,s)) = f((p,s)) + ε
    - ❖ On a augmenté le flot de ε
  - f’(a) = f(a) pour tout autre arc a

- Alors f’ est un flot compatible avec R et f’((p,s)) > f((p,s))
  - f’ maximise le flot f

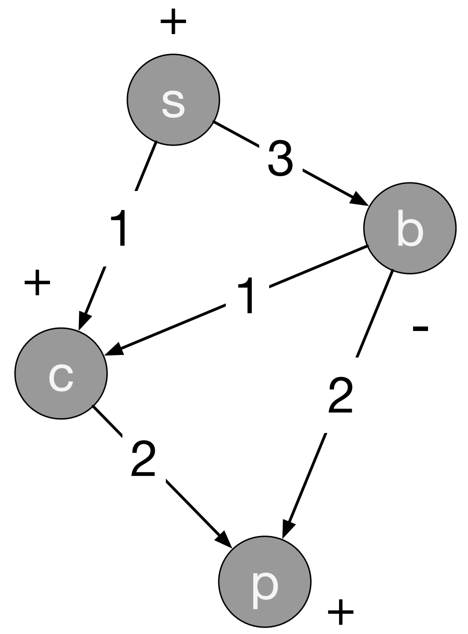
Application



- Ici, on réduit le précédent flot qui passait de c à h !
  - et on le bascule sur cp
- Et pour hp
  - besoin de lui enlever la capacité déroutée de ch?

Méthode de Ford & Fulkerson

- Schéma de principe
  - Construire un flot qui progressivement respecte les conditions précédentes
  - Marquage itératif
    - ❖ Identifier des chemins *améliorant* (ou *augmentant*)
    - ❖ Flot nul → flot admissible
      - – Point de départ de l'algorithme
      - – Marquage de s
    - ❖ Marquer
      - – + → sommet extrémité d'un arc dont l'origine est marquée et sur lequel le flux f peut **augmenter**
      - – - → sommet origine d'un arc dont l'extrémité est marquée et sur lequel le flux f peut **diminuer**
      - – Une procédure TANT QUE jusqu'à atteindre le puits p
        - » Ou le marquage est devenu impossible



marquage: s / c / b / p

Méthode de Ford & Fulkerson

- 2 Cas de figure
  1. On peut marquer p

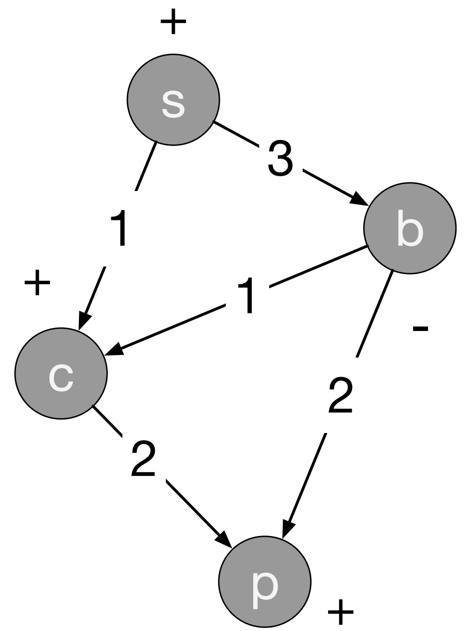
     → il existe un chemin possible

     ❖ Il *suffit* de modifier le flot en l'augmentant

     – Capacité minimale sur le chemin

  2. On ne peut pas marquer p

     ❖ Le flot est déjà maximal !



marquage: s / c / b / p

Méthode de Ford & Fulkerson

- Entrées : un réseau de transport R=(S,A,s,p,c)
- Pré-condition : pas de chemin de s à p de capacité infinie
- Sorties : flot f : A -> $\mathbb{R}^+$ compatible avec R et maximal

- Fin = FAUX
- Partir d'un flot initial compatible (e.g. nul)
  - Pour tout u in A, $f(a) = 0$
- Soit R' =(S,A',c') le réseau d'écart associé à R et f
- Tant qu'il existe un chemin C dans R'
  - $capa = \infty$
  - Pour tout $u \in C$, Si $c(u) \leq capa$: capa = c(u)
  - Pour tout $u \in C$
    - ❖ Si u est direct : f(a(u)) += capa
    - ❖ Sinon : f(a(u)) -= capa
  - f((p,s)) += capa
- Retourner f()

Implémentation en python

- Wikipedia ©

```python
def find_path(self, source, sink, path):
    if source == sink:
        return path
    for edge in self.get_edges(source):
        residual = edge.capacity - self.flow[edge]
        if residual > 0 and edge not in path:
            result = self.find_path( edge.sink, sink, path + [edge])
            if result != None:
                return result

def max_flow(self, source, sink):
    path = self.find_path(source, sink, [])
    while path != None:
        residuals = [edge.capacity - self.flow[edge] for edge in path]
        flow = min(residuals)
        for edge in path:
            self.flow[edge] += flow
            self.flow[edge.redge] -= flow
        path = self.find_path(source, sink, [])
    return sum(self.flow[edge] for edge in self.get_edges(source))
```

Terminaison et correction/optimalité ?

- “Facile” si les capacités sont entières
  - complexité en $O((|S|+|A|).f)$
- Pareil si les capacités sont des nombres rationnels
  - $c : A \to \mathbb{Q} \cup \{+\infty\}$

Résumé

- La méthode de Ford Fulkerson (1956)
  - pas de garantie théorique en terme de convergence avec des capacités non rationnelles
  - si les capacités sont des entiers alors complexité en temps O((|S|+|A|).f)
  - pas vraiment un algorithme car la recherche de chemin augmentant n'est pas spécifié

- Vers un vrai algorithme/une implémentation avec convergence garantie (c: A -> $\mathbb{R}$?)
  - Notion de chemins augmentant le plus court
    - ❖ Edmonds–Karp (1972)
      - O(|A|^2.|S|)
    - ❖ Dinic (1970)
      - O(|A|.|S|^2)
  - Aujourd'hui : <= O(|S|^3)…et encore grâce à Tarjan notamment !

Pseudo-code - Bruno Bachelet ©

Titre: FordFulkerson2
Entrées: R = (X;U;c) un réseau, s et p deux sommets.
Sorties: f() une fonction indiquant le flot circulant à travers chaque arc.
Variables intermédiaires: G = (X;U') un graphe, c'() une fonction indiquant la capacité d'un arc de G, a() une fonction indiquant à quel arc de R correspond un arc de G, C un sous-ensemble d'arcs, u un arc, m un réel.

Début
```
 pour tout arc u ∈ U faire f(u) ← 0;
 construire le graphe d'écart G;

 tant que ∃ un chemin C entre s et p ∈ G faire
  m ← +oo;
  pour tout u ∈ C faire si c'(u) < m alors m ← c'(u);

  pour tout u ∈ C faire
   si a(u) est direct alors f(a(u)) ← f(a(u)) + m;
   sinon f(a(u)) ← f(a(u)) - m;
  fin pour;

  construire le graphe d'écart G;
 fin tant que;
```
Fin

Titre: ConstruireGrapheEcart
Entrées: R = (X;U;c) un réseau, f() le flot courant sur R.
Sorties: G = (X;U') le graphe d'écart, c'() une fonction indiquant la capacité d'un arc de G, a() une fonction indiquant à quel arc de R correspond un arc de G.
Variables intermédiaires: u un arc, x et y deux noeuds, m un réel.

Début
```
 pour tout u = (x;y) ∈ U faire
  si f(u) > 0 alors
   U' ← U' ∪ {(y;x)};
   c'((y;x)) ← f(u);
   a((y;x)) ← u;
  fin si;

  si f(u) < c(u) alors
   U' ← U' ∪ {(x;y)};
   c'((x;y)) ← c(u) - f(u);
   a((x;y)) ← u;
  fin si;
 fin pour;
```
Fin

Titre: RechercherChemin
Entrées: R = (X;U;c) un réseau, s et p deux sommets, f() le flot courant.
Sorties: pred() une fonction indiquant par quel arc on arrive à un noeud donné à partir de s.
Variables intermédiaires: X' un sous-ensemble de noeuds, accessible() une fonction qui indique si un noeud est accessible à partir de s.

Début
```
 X' ← {s};

 pour tout x ∈ X faire
  accessible(x) ← faux;
  pred(x) ← nil;
 fin pour;

 accessible(s) ← vrai;

 tant que X' ≠ ∅ et non accessible(p) faire
  choisir x ∈ X';
  X' ← X' - {x};

  pour tout u = (x;y) ∈ U faire
   si non accessible(y) alors
    X' ← X' ∪ {y};
    pred(y) ← x;
    accessible(y) ← vrai;
   fin si;
  fin pour;
 fin tant que;
```
Fin

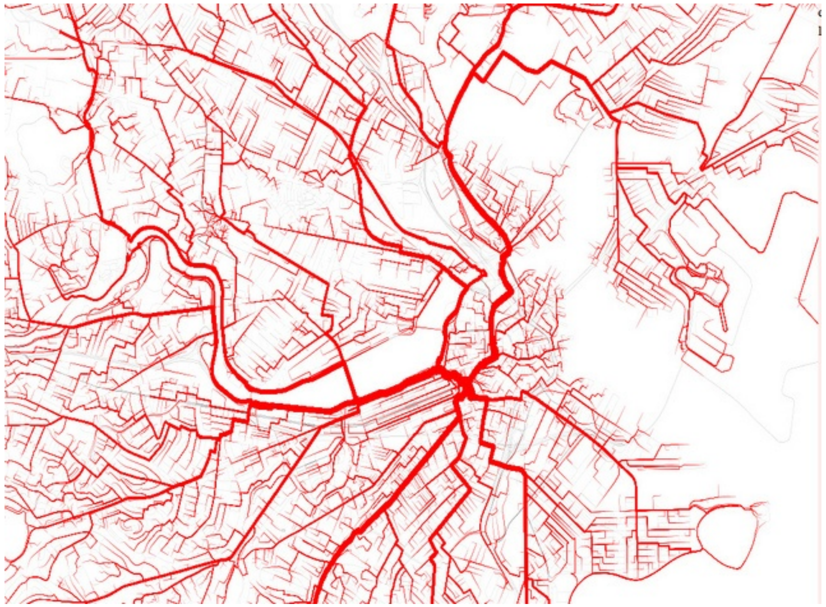
http://www.nawouak.net

Perspectives / Variantes

- Variantes possibles
  - Correspondances dans les graphes bipartis
    - ❖ agence matrimoniale, choix de sujets, etc.
  - Flot maximal de coût minimal
    - ❖ en plus de la fonction capacité, on applique une fonction coût à chaque arc
      - Ex: construction d'une fibre optique
    - ❖ Coût par sommet
      - Transbordement de marchandise
  - Flots à commodité multiple
    - ❖ satisfaire des demandes pour plusieurs couples entrées-sorties
    - ❖ encore plus compliqué (dans le cas général avec des flots entiers)
  - Respecter des contraintes
    - ❖ Qualité de service : délai max respecté pour X% des paquets
    - ❖ Caractéristiques : une piste cyclable seulement pour les vélos

Perspectives

- Réseaux de transport
  - Vitesse à prendre en compte ?
    - ❖ Piétons vs. Voitures vs. Vélos
  - Stochastique
    - ❖ La charge varie au cours du temps
  - Imprécisions d'estimation
    - ❖ Capacité
    - ❖ Charge actuelle
  - Coût de correspondance
    - ❖ Ligne de bus = pas de changement
    - ❖ Demande à modifier le graphe



UNIVERSITÉ DE STRASBOURG