

# Cours 2 - Algorithmes Probabilistes

## Coupe Minimum

Semestre Automne 2021-2022 - Université Claude Bernard Lyon 1

Christophe Crespelle

`christophe.crespelle@univ-lyon1.fr`



département

**Informatique**

Faculté des Sciences et Technologies

Université Claude Bernard Lyon 1

\* Merci à A. Parreau et E. Duchene pour le matériel pédagogique ayant servi de base à ces slides.

# Algorithmes probabilistes

- **Algos probabilistes :**

- ▶ font des choix de manière aléatoire

Rq : algo déterministes font aussi des choix !

# Algorithmes probabilistes

- **Algos probabilistes :**

- ▶ font des choix de manière aléatoire

Rq : algo déterministes font aussi des choix !

- ▶ Aléatoire  $\neq$  quelconque !!!

# Algorithmes probabilistes

- **Algos probabilistes :**

- ▶ font des choix de manière aléatoire
  - Rq : algo déterministes font aussi des choix !
- ▶ Aléatoire  $\neq$  quelconque !!!
  - ▶ souvent : aléatoirement **uniformement**
  - ▶ pas toujours, par ex. : choix aléatoire d'un sommet proportionnellement au degré

# Algorithmes probabilistes

- **Algos probabilistes :**
  - ▶ font des choix de manière aléatoire  
Rq : algo déterministes font aussi des choix !
  - ▶ Aléatoire  $\neq$  quelconque !!!
    - ▶ souvent : aléatoirement **uniformement**
    - ▶ pas toujours, par ex. : choix aléatoire d'un sommet proportionnellement au degré
- **Qu'est-ce qu'on perd ?**
  - on obtient pas toujours l'optimum

# Algorithmes probabilistes

- **Algos probabilistes :**
  - ▶ font des choix de manière aléatoire
    - Rq : algo déterministes font aussi des choix !
  - ▶ Aléatoire  $\neq$  quelconque !!!
    - ▶ souvent : aléatoirement **uniformement**
    - ▶ pas toujours, par ex. : choix aléatoire d'un sommet proportionnellement au degré
- **Qu'est-ce qu'on perd ?**
  - on obtient pas toujours l'optimum
- **Qu'est-ce qu'on gagne ?**
  - la complexité de l'algo baisse
  - ▶ par exemple : exponentiel  $\rightarrow$  polynomial
  - ▶ ici :  $O(n^3 m^2) \rightarrow O(n^4 \log n)$

# Algorithmes probabilistes

- **Algos probabilistes :**

- ▶ font des choix de manière aléatoire
  - Rq : algo déterministes font aussi des choix !
- ▶ Aléatoire  $\neq$  quelconque !!!
  - ▶ souvent : aléatoirement **uniformement**
  - ▶ pas toujours, par ex. : choix aléatoire d'un sommet proportionnellement au degré

- **Qu'est-ce qu'on perd ?**

→ on obtient pas toujours l'optimum

- **Qu'est-ce qu'on gagne ?**

→ la complexité de l'algo baisse

- ▶ par exemple : exponentiel → polynomial
- ▶ ici :  $O(n^3 m^2) \rightarrow O(n^4 \log n)$

- **Pourquoi ça marche ?**

tout dépend de la probabilité d'obtenir l'optimum → pas trop faible

## Une version un peu différente de coupe minimum

- **Entrée** : un multigraphe non orienté (aretes multiples possibles pour un couple de sommet)



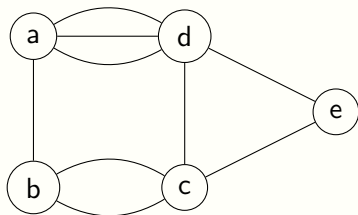
## Une version un peu différente de coupe minimum

- **Entrée** : un multigraphe non orienté (arêtes multiples possibles pour un couple de sommet)
- **Sortie** : une coupe  $(U, U')$  de  $G$  (=bipartition de  $V(G)$ ) ayant le nombre minimum d'arêtes qui traversent

## Une version un peu différente de coupe minimum

- **Entrée** : un multigraphe non orienté (aretes multiples possibles pour un couple de sommet)
- **Sortie** : une **coupe**  $(U, U')$  de  $G$  (=bipartition de  $V(G)$ ) ayant le nombre minimum d'aretes qui traversent

Ex. 1 :



## Une version un peu différente de coupe minimum

- **Entrée** : un multigraphe non orienté (arêtes multiples possibles pour un couple de sommet)
- **Sortie** : une **coupe**  $(U, U')$  de  $G$  (=bipartition de  $V(G)$ ) ayant le nombre minimum d'arêtes qui traversent

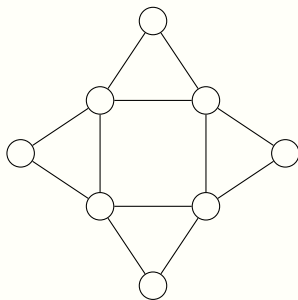
Differences avec precedemment :

- **ni puits ni source**
- graphe non orienté (cas particulier de graphe orienté)
- arêtes multiples (equiv. capacités entières)

## Une version un peu différente de coupe minimum

- **Entrée** : un multigraphe non orienté (aretes multiples possibles pour un couple de sommet)
- **Sortie** : une **coupe**  $(U, U')$  de  $G$  (=bipartition de  $V(G)$ ) ayant le nombre minimum d'aretes qui traversent

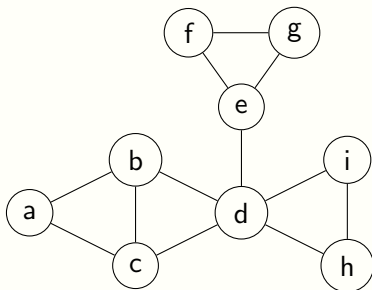
Ex. 2 :



## Une version un peu différente de coupe minimum

- **Entrée** : un multigraphe non orienté (aretes multiples possibles pour un couple de sommet)
- **Sortie** : une **coupe** ( $U, U'$ ) de  $G$  (=bipartition de  $V(G)$ ) ayant le nombre minimum d'aretes qui traversent

Ex. 3 :



## Une version un peu differente de coupe minimum

- Solution ?

## Une version un peu différente de coupe minimum

- **Solution ?**

$n(n - 1)/2$  fois l'algo d'EK, avec toutes les paires  $\{source, puits\}$  possibles

## Une version un peu différente de coupe minimum

- **Solution ?**

$n(n - 1)/2$  fois l'algo d'EK, avec toutes les paires  $\{source, puits\}$  possibles

- **Complexite ?**



## Une version un peu différente de coupe minimum

- **Solution ?**

$n(n - 1)/2$  fois l'algo d'EK, avec toutes les paires  $\{source, puits\}$  possibles

- **Complexite ?**

$O(n^3 m^2)$

( $O(n^3 m)$  avec le meilleur algo connu pour le flot max, qui est en  $O(nm)$ )

# Une version un peu différente de coupe minimum

- **Solution ?**

$n(n-1)/2$  fois l'algo d'EK, avec toutes les paires  $\{source, puits\}$  possibles

- **Complexite ?**

$O(n^3 m^2)$

( $O(n^3 m)$  avec le meilleur algo connu pour le flot max, qui est en  $O(nm)$ )

- **Algo probabiliste :**

- ▶ On va faire un algo probabiliste en  $O(n^4 \log n)$   
(le meilleur algo proba est en  $O(n^2 \log^3 n)$ )

# Une version un peu differente de coupe minimum

- **Solution ?**

$n(n-1)/2$  fois l'algo d'EK, avec toutes les paires  $\{source, puits\}$  possibles

- **Complexite ?**

$O(n^3 m^2)$

( $O(n^3 m)$  avec le meilleur algo connu pour le flot max, qui est en  $O(nm)$ )

- **Algo probabiliste :**

- ▶ On va faire un algo probabiliste en  $O(n^4 \log n)$   
(le meilleur algo proba est en  $O(n^2 \log^3 n)$ )
- ▶ et en plus d'une simplicité déconcertante !

## Algo probabiliste pour coupe minimum

- Algo probabiliste  $\Rightarrow$  deux executions ne donnent pas le meme resultat !
- On l'execute plusieurs (= beaucoup de) fois et on garde la meilleure solution

## Algo probabiliste pour coupe minimum

- Algo probabiliste  $\Rightarrow$  deux executions ne donnent pas le meme resultat !
- On l'execute plusieurs (= beaucoup de) fois et on garde la meilleure solution
- Deux questions :
  - ▶ Quelle est la proba d'obtenir une solution optimale ?
  - ▶ Il faut faire combien d'essais ?

## Algo probabiliste pour coupe minimum

- Algo probabiliste  $\Rightarrow$  deux executions ne donnent pas le meme resultat !
- On l'execute plusieurs (= beaucoup de) fois et on garde la meilleure solution
- Deux questions :
  - ▶ Quelle est la proba d'obtenir une solution optimale ?
  - ▶ Il faut faire combien d'essais ?
    - ▶ On veut proba  $\rightarrow 1$  lorsque la taille de l'entree  $\rightarrow +\infty$  (Waouh !)

# Algo probabiliste pour coupe minimum

- Algo probabiliste  $\Rightarrow$  deux executions ne donnent pas le meme resultat !
- On l'execute plusieurs (= beaucoup de) fois et on garde la meilleure solution
- Deux questions :
  - ▶ Quelle est la proba d'obtenir une solution optimale ?
  - ▶ Il faut faire combien d'essais ?
    - ▶ On veut proba  $\rightarrow 1$  lorsque la taille de l'entree  $\rightarrow +\infty$  (Waouh !)
    - ▶ ... en fait, ca suffit que la proba soit une constante.

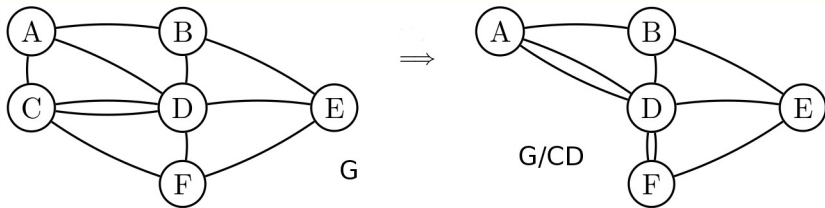
# Cas des graphes non connexes



# Algo probabiliste pour coupe minimum

## Définition (contraction d'arete)

Soit  $G = (V, E)$  un multigraphe sans boucle et  $uv$  une arete de  $G$ . Le raphe contracte de  $G$  selon l'arete  $uv$ , note  $G/uv$ , est defini par  $G/uv = (V \setminus \{u\}, E \setminus \{\{ux \mid x \in N(u)\}\} \cup \{\{vx \mid ux \in E \text{ et } x \neq v\}\})$ .



# Algo probabiliste pour coupe minimum

---

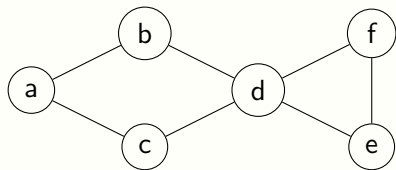
## Algorithme 1 : Algorithme Contraction Aleatoire

---

```
1 pour tout  $v \in V$  faire  
2   |  $S(v) \leftarrow \{v\}$   
3 fin  
4 tant que  $|V| > 2$  faire  
5   | choisir une arete  $uv$  de  $G$  uniformement aleatoirement;  
6   |  $G \leftarrow G/uv$ ;  
7   |  $S(v) \leftarrow S(v) \cup S(u)$ ;  
8 fin  
9 retourner la coupe  $(S(x), S(y))$ ;   (ou  $\{x, y\} = V$ )
```

---

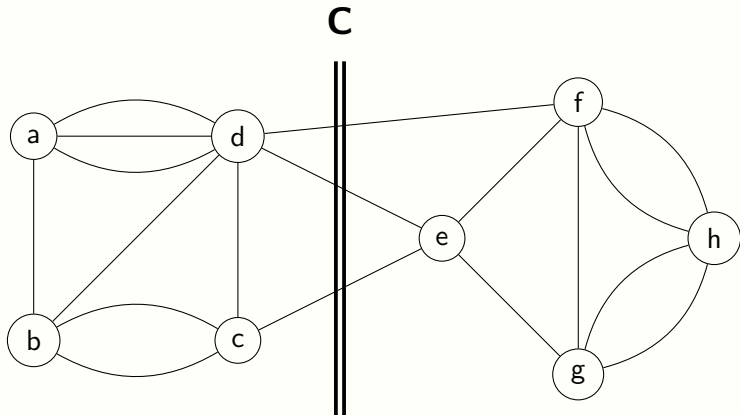
## Exemple



# Analyse de l'algo de Contraction Aleatoire

- l'algo fait toujours  $n - 2$  contractions, pour finir avec un graphe a 2 sommets
- on note  $k$  la taille d'une coupe minimum de  $G$
- on fixe une coupe minimum  $C = (X, Y)$  de  $G$  et on calcule la proba que l'algo retourne la coupe  $C$
- Rq. : l'algo retourne  $C$  ssi il ne contracte aucune arete traversant  $C$

## Analyse de l'algo de Contraction Aleatoire

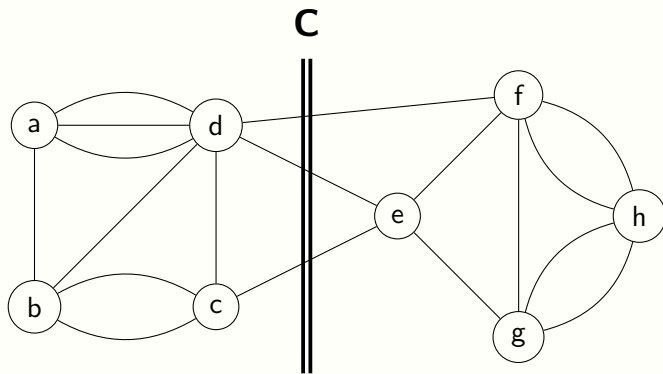


## Analyse de l'algo de Contraction Aleatoire

- A la premiere contraction, la proba  $p$  de choisir une arete ne traversant pas  $C$  est  $p = 1 - \frac{k}{m}$ .
- on note  $E_j$  l'evenement suivant : " a la  $j^{eme}$  iteration de la boucle principale, l'arete  $uv$  selectionnee ne traverse pas  $C$ "
- on calcule la proba  $p_j = Pr[E_j | \bigcap_{i=1}^{j-1} E_i]$
- $G_j$  le graphe avant l'iteration  $j$ , dans l'evenement  $\bigcap_{i=1}^{j-1} E_i$

## proba d'obtenir la coupe minimum $C$

- $G_j$  a exactement  $n - j + 1$  sommets et la coupe minimum de  $G_j$  a taille  $k$



## proba d'obtenir la coupe minimum $C$

- $G_j$  a exactement  $n - j + 1$  sommets et la coupe minimum de  $G_j$  a taille  $k$
- ainsi, le degre minimum dans  $G_j$  est au moins  $k$ , et donc  $G_j$  a au moins  $k(n - j + 1)/2$  aretes.



## proba d'obtenir la coupe minimum $C$

- $G_j$  a exactement  $n - j + 1$  sommets et la coupe minimum de  $G_j$  a taille  $k$
- ainsi, le degre minimum dans  $G_j$  est au moins  $k$ , et donc  $G_j$  a au moins  $k(n - j + 1)/2$  aretes.
- on obtient

$$1 - p_j = Pr[\overline{E_j} | \bigcap_{i=1}^{j-1} E_i] \leq k / (k(n - j + 1)/2) = 2 / (n - j + 1)$$

## proba d'obtenir la coupe minimum $C$

- $G_j$  a exactement  $n - j + 1$  sommets et la coupe minimum de  $G_j$  a taille  $k$
- ainsi, le degre minimum dans  $G_j$  est au moins  $k$ , et donc  $G_j$  a au moins  $k(n - j + 1)/2$  aretes.
- on obtient
$$1 - p_j = Pr[\overline{E_j} | \bigcap_{i=1}^{j-1} E_i] \leq k / (k(n - j + 1)/2) = 2 / (n - j + 1)$$
- d'ou  $p_j \geq 1 - 2 / (n - j + 1) = (n - j - 1) / (n - j + 1)$

## proba d'obtenir la coupe minimum $C$

- comme  $Pr[A \cap B] = Pr[A|B] \times Pr[B]$ , on obtient

$$p = Pr\left[\bigcap_{j=1}^{n-2} E_j\right] = \prod_{j=1}^{n-2} p_j$$

## proba d'obtenir la coupe minimum $C$

- comme  $Pr[A \cap B] = Pr[A|B] \times Pr[B]$ , on obtient

$$p = Pr\left[\bigcap_{j=1}^{n-2} E_j\right] = \prod_{j=1}^{n-2} p_j$$

- ainsi la proba d'obtenir la coupe minimum  $C$  par l'algo est

$$p \geq \frac{n-2}{n} \times \frac{n-3}{n-1} \times \frac{n-4}{n-2} \times \cdots \times \frac{3}{5} \times \frac{2}{4} \times \frac{1}{3} = \frac{2}{n(n-1)}$$

## proba d'obtenir la coupe minimum $C$

- comme  $Pr[A \cap B] = Pr[A|B] \times Pr[B]$ , on obtient

$$p = Pr\left[\bigcap_{j=1}^{n-2} E_j\right] = \prod_{j=1}^{n-2} p_j$$

- ainsi la proba d'obtenir la coupe minimum  $C$  par l'algo est

$$p \geq \frac{n-2}{n} \times \frac{n-3}{n-1} \times \frac{n-4}{n-2} \times \cdots \times \frac{3}{5} \times \frac{2}{4} \times \frac{1}{3} = \frac{2}{n(n-1)}$$

- on fait maintenant  $t$  iterations de l'algo de contraction, chacune ayant une proba de succes  $p$

## proba d'obtenir la coupe minimum $C$

- comme  $Pr[A \cap B] = Pr[A|B] \times Pr[B]$ , on obtient

$$p = Pr\left[\bigcap_{j=1}^{n-2} E_j\right] = \prod_{j=1}^{n-2} p_j$$

- ainsi la proba d'obtenir la coupe minimum  $C$  par l'algo est

$$p \geq \frac{n-2}{n} \times \frac{n-3}{n-1} \times \frac{n-4}{n-2} \times \cdots \times \frac{3}{5} \times \frac{2}{4} \times \frac{1}{3} = \frac{2}{n(n-1)}$$

- on fait maintenant  $t$  iterations de l'algo de contraction, chacune ayant une proba de succes  $p$
- Comment choisir  $t$  pour avoir une proba constante de trouver la coupe  $C$  dans au moins une des  $t$  iterations ?

## proba d'obtenir la coupe minimum $C$

- la proba de ne trouver  $C$  dans aucune des  $t$  executions est  
 $p_{echec} = (1 - p)^t$

## proba d'obtenir la coupe minimum $C$

- la proba de ne trouver  $C$  dans aucune des  $t$  executions est  
 $p_{echec} = (1 - p)^t$
- maths  $\Rightarrow (1 - p)^{\frac{1}{p}} \leq 1/e$



## proba d'obtenir la coupe minimum $C$

- la proba de ne trouver  $C$  dans aucune des  $t$  executions est  
 $p_{echec} = (1 - p)^t$
- maths  $\Rightarrow (1 - p)^{\frac{1}{p}} \leq 1/e$
- en faisant  $t = \frac{1}{p}$  executions de l'algo, on obtient  
 $p_{echec} \leq 1/e = cte$

## proba d'obtenir la coupe minimum $C$

- la proba de ne trouver  $C$  dans aucune des  $t$  executions est  
 $p_{echec} = (1 - p)^t$
- maths  $\Rightarrow (1 - p)^{\frac{1}{p}} \leq 1/e$
- en faisant  $t = \frac{1}{p}$  executions de l'algo, on obtient  
 $p_{echec} \leq 1/e = cte$
- pour avoir  $p_{echec} \rightarrow 0$ , on peut faire par exemple  $t = \frac{\log n}{p}$   
executions de l'algo et on obtient  $p_{echec} \leq (1/e)^{\log n} = 1/n$

## proba d'obtenir la coupe minimum $C$

- la proba de ne trouver  $C$  dans aucune des  $t$  executions est  
 $p_{echec} = (1 - p)^t$
- maths  $\Rightarrow (1 - p)^{\frac{1}{p}} \leq 1/e$
- en faisant  $t = \frac{1}{p}$  executions de l'algo, on obtient  
 $p_{echec} \leq 1/e = cte$
- pour avoir  $p_{echec} \rightarrow 0$ , on peut faire par exemple  $t = \frac{\log n}{p}$  executions de l'algo et on obtient  $p_{echec} \leq (1/e)^{\log n} = 1/n$
- c'est a dire que la proba de trouver une coupe minimum  $C$  fixee apres  $\frac{n(n-1) \log n}{2}$  executions de l'algo contraction est au moins  $1 - \frac{1}{n} \rightarrow 1$  quand  $n \rightarrow +\infty$

# Analyse de complexite de l'algo proba pour coupe minimum

- soit  $f(n)$  la complexite de l'algo de contraction, la complexite totale de l'algo proba pour coupe minimum est  $O(f(n) \cdot n^2 \log n)$
- Complexite de Contraction ?
  - ▶ boucle principale :  $O(n)$  fois
  - ▶ ligne 5 :  $O(1)$
  - ▶ ligne 6 :  $O(n)$
  - ▶ ligne 7 :  $O(n)$

complexite totale :  $O(n^2)$

## Conclusion de l'algo proba pour coupe minimum

- Algo avec proba de succes  $p \geq 1 - \frac{1}{n}$  et complexite  $O(n^4 \log n)$
- rien n'empêche de :
  - ▶ ne faire que  $t = \frac{n(n-1)}{2}$  executions :  $p \geq 1 - \frac{1}{e}$  et complexite  $O(n^4)$
  - ▶ faire  $t = \log 1000 \frac{n(n-1)}{2}$  executions :  $p \geq 1 - \frac{1}{1000}$  et complexite  $O(n^4)$  (la cte= $\log 1000$  est cachee dans le  $O(\cdot)$ )
  - ▶ faire  $t = n \cdot \frac{n(n-1)}{2}$  executions :  $p \geq 1 - \frac{1}{e^n}$  et complexite  $O(n^5)$

# Conclusion de l'algo proba pour coupe minimum

## Questions subsidiaires :

- Mq il y a au plus  $O(n^2)$  coupe minimum dans un graphe
- Mq il peut y avoir  $\Omega(2^n)$   $s, t$ -coupe minimum dans un graphe

## Une autre implementation de l'algo (dite "peridurale")

⇒ C'est celle qu'on adoptera pour le TP.