M1 Info - Optimisation et Recherche Opérationnelle

Cours 2 - Algorithmes Probabilistes

Coupe Minimum

Semestre Automne 2020-2021 - Université Claude Bernard Lyon 1

Christophe Crespelle
christophe.crespelle@inria.fr

département Informatique
Faculté des Sciences et Technologies
Université Claude Bernard Lyon 1

* Merci a A. Parreau et E. Duchene pour le materiel pedagogique ayant servi de base a ces slides.

Algorithmes probabilistes

- **Algos probabilistes :**
  - font des choix de maniere aleatoire

    Rq : algo deterministes font aussi des choix !

Algorithmes probabilistes

- **Algos probabilistes :**
  - font des choix de maniere aleatoire

    Rq : algo deterministes font aussi des choix !
  - Aleatoire $\neq$ quelconque ! ! !

Algorithmes probabilistes

- **Algos probabilistes :**
  - font des choix de maniere aleatoire

    Rq : algo deterministes font aussi des choix !
  - Aleatoire $\neq$ quelconque ! ! !
    - souvent : aleatoirement **uniformement**
    - pas toujours, par ex. : choix aleatoire d'un sommet proportionnellement au degre

Algorithmes probabilistes

- **Algos probabilistes :**
  - font des choix de maniere aleatoire

    Rq : algo deterministes font aussi des choix !
  - Aleatoire $\neq$ quelconque ! ! !
    - souvent : aleatoirement **uniformement**
    - pas toujours, par ex. : choix aleatoire d'un sommet proportionnellement au degre

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

  $\rightarrow$ on obtient pas toujours l'optimum

Algorithmes probabilistes

- **Algos probabilistes :**
  - font des choix de maniere aleatoire

    Rq : algo deterministes font aussi des choix !
  - Aleatoire $\neq$ quelconque ! ! !
    - souvent : aleatoirement **uniformement**
    - pas toujours, par ex. : choix aleatoire d'un sommet proportionnellement au degre

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

  $\rightarrow$ on obtient pas toujours l'optimum

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

  $\rightarrow$ la complexite de l'algo baisse
  - par exemple : exponentiel $\rightarrow$ polynomial
  - ici : $O(\underline{n^3 m^2}) \rightarrow O(\underline{n^4 \log^2 n})$

Algorithmes probabilistes

- **Algos probabilistes :**
  - font des choix de maniere aleatoire

    Rq : algo deterministes font aussi des choix !
  - Aleatoire $\neq$ quelconque ! ! !
    - souvent : aleatoirement **uniformement**
    - pas toujours, par ex. : choix aleatoire d'un sommet proportionnellement au degre

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

  $\rightarrow$ on obtient pas toujours l'optimum

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

  $\rightarrow$ la complexite de l'algo baisse
  - par exemple : exponentiel $\rightarrow$ polynomial
  - ici : $O(n^3 m^2) \rightarrow O(n^4 \log^2 n)$

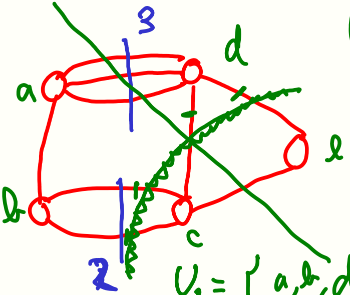
- **Pourquoi ca marche ?**

  tout depend de la probabilite d'obtenir l'optimum $\rightarrow$ pas trop faible

Une version un peu differente de coupe minimum

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

ex :



$U = \{a, b, c\}$

$U' = \{d, e\}$

$U_2 = \{a, b, d\}$

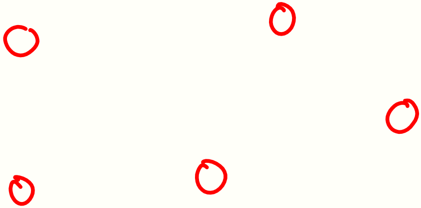
$U_2' = \{c, e\}$

$C(U_2, U_2') = 4$

$\# \text{coupes} = \dfrac{2^n - 2}{2} = \boxed{2^{n-1} - 1}$

Une version un peu differente de coupe minimum

- **Entrée :** un multigraphe non oriente (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



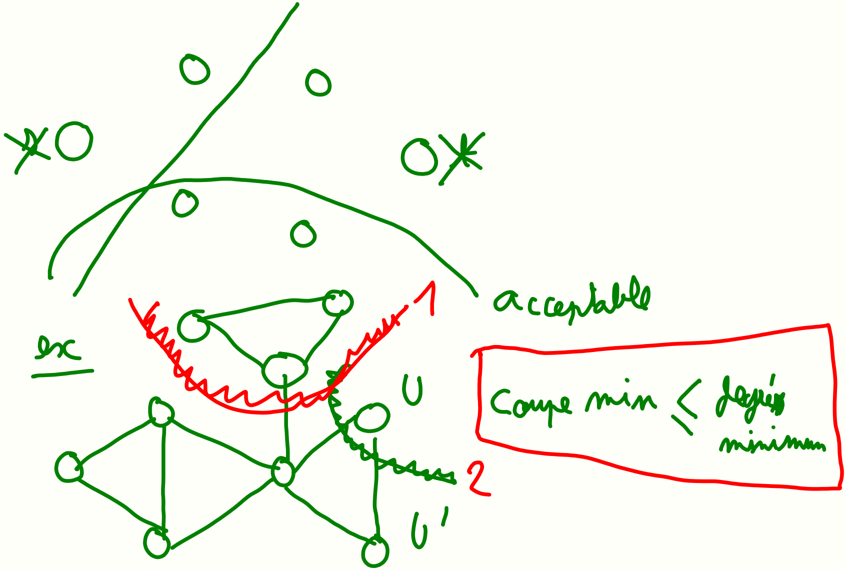
Une version un peu differente de coupe minimum

- **Entrée :** un multigraphe non oriente (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

Differences avec precedemment :

- **ni puits ni source**
- graphe non oriente (cas particulier de graphe oriente)
- aretes multiples (equiv. capacites entieres)

Une version un peu differente de coupe minimum



Une version un peu differente de coupe minimum

- <u>**Solution ?**</u>

$O(n^2)$ $O(nm^2)$

Essayer tous les couples (s,t) et faire E.K.

<u>modif de l'entrée :</u>

arêtes mult $\Rightarrow$ une seule arête avec une capacité entière

arêtes non orientées $\Rightarrow$ 2 arcs symétriques

Une version un peu differente de coupe minimum

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

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 ?**

Une version un peu differente de coupe minimum

- **Solution ?** = # Couples $\{s, t\}$
  $n(n-1)/2$ 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 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 $O(n^4 \log^2 n)$
    (le meilleur algo proba est en $O(n^2 \log^2 n)$)

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 $O(n^4 \log^2 n)$
    (le meilleur algo proba est en $O(n^2 \log^2 n)$)
  - et en plus d'une simplicite deconcertante !

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 ⇒ 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 → 1 lorsque la $\overset{N}{\text{taille de l'entree}}$ → +∞ (Waouh !)
    - ... en fait, ca suffit que la proba soit une constante.

$\boxed{P_{succes} \geq 0,2}$

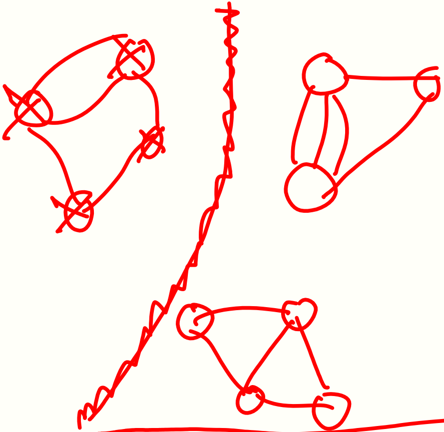
$P_{echec} \leq 0,8 \xRightarrow{\text{ln fois}} P_{echec} \leq 0,8^{\ln}$ gagné

$\ln = \log N$

$\ln = \log \log \log \log N$

Algo probabiliste pour coupe minimum

Cas des graphes non connexes



Coupe min ?

||

0

parcours en largeur (BFS)

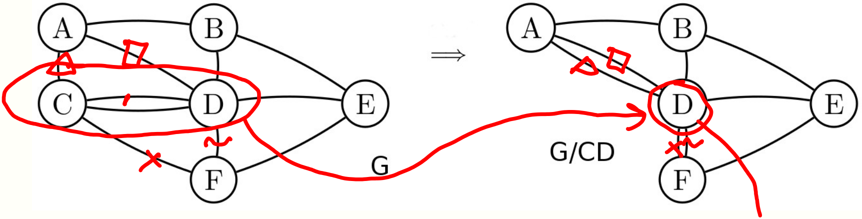
$O(m)$

Algo probabiliste pour coupe minimum

à partir de maintenant tous les graphes en entrée sont connexes

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 $G/uv = (V \setminus \{u\}, E \setminus \{\!\{ux \mid x \in N(u)\}\!\} \cup \{\!\{vx \mid ux \in E \text{ et } x \neq v\}\!\})$.



$S(D) = \{C, D\}$

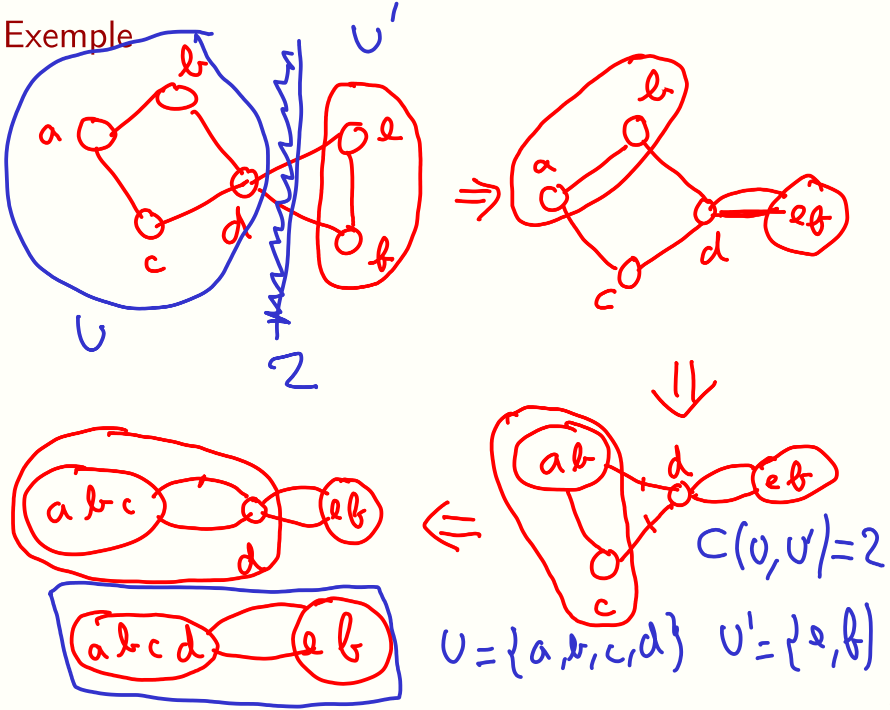
Algo probabiliste pour coupe minimum

Algorithme 1 : Algorithme Contraction Aleatoire

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

$O(n)$

Exemple



$C(U,U')=2$

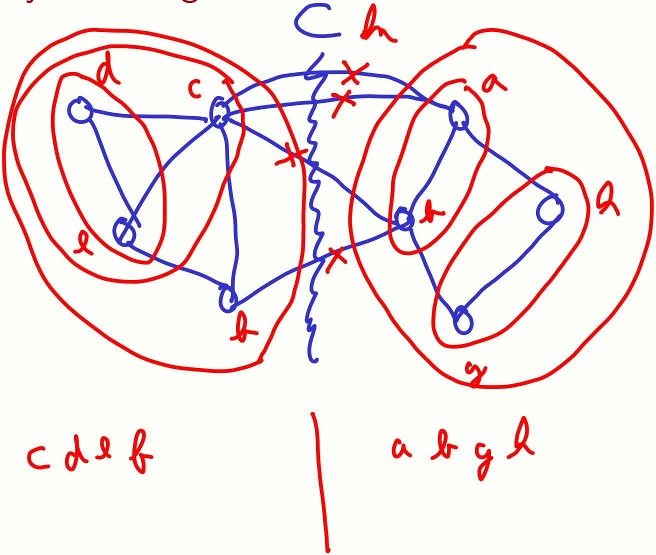
$U=\{a,b,c,d\}$ $U'=\{e,f\}$

Analyse de l'algo de Contraction Aleatoire

perd un sommet à chaque contraction

- 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

\# arêtes traversant C

- A la premiere contraction, la proba p de choisir une arete ne traversant pas C est $p = 1 - \frac{k}{m}$.

\# arêtes total

- 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 $p_j = Pr[E_j | \bigcap_{i=1}^{j-1} E_i]$ $\Rightarrow$ bien passé jusqu'à j-1

bien passé en j

- G_j le graphe avant l'iteration j, dans l'evenement $\bigcap_{i=1}^{j-1} E_i$

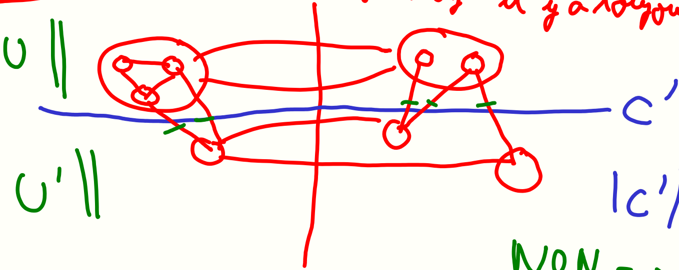
proba d'obtenir la coupe minimum C

$n - (j - 1)$

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

C

dans G_j il y a toujours C



$|C'| < k$?

NON car

C' existe dans G

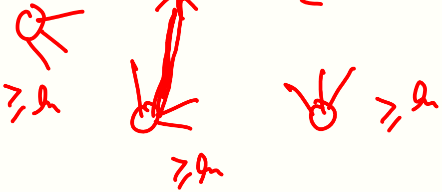
et $|C'|_G = |C'|_{G_j}$

proba d'obtenir la coupe minimum C

degres(u) = # aretes incidentes au sommet u

- 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.

$$\frac{k(n-j+1)}{2}$$



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

$$\underbrace{1-p_j} = Pr[\overline{E_j} | \bigcap_{i=1}^{j-1} E_i] \le \overset{\#\text{ arêtes de } G_j \text{ qui traversent } C}{\cancel{k}} / (\underbrace{\cancel{k}(n-j+1)/2}_{\#\text{ arêtes dans } G_j}) = \underbrace{2/(n-j+1)}$$

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 $p_j \geq 1 - 2/(n - j + 1) = (n - j - 1)/(n - j + 1)$

$$\frac{n-j+1}{n-j+1} - \frac{2}{n-j+1} = \frac{n-j+1-2}{n-j+1}$$

proba d'obtenir la coupe minimum C

- comme $Pr[A \cap B] = Pr[A|B] \times Pr[B]$, on obtient
  $p = Pr[\bigcap_{i=1}^{n-2} E_i] = \prod_{i=1}^{n-2} p_j$

proba d'obtenir la coupe minimum C

- comme $Pr[A \cap B] = Pr[A|B] \times Pr[B]$, on obtient
$$p = Pr[\bigcap_{i=1}^{n-2} E_i] = \prod_{i=1}^{n-2} p_j$$
- ainsi la proba d'obtenir la coupe minimum C par l'algo est
$$p \geq \frac{n-2}{n} \times \frac{n-3}{n-1} \times \frac{n-4}{n-2} \times \frac{n-5}{n-3} \times \cdots \times \frac{3}{5} \times \frac{2}{4} \times \frac{1}{3} = \frac{2}{n(n-1)} = O\left(\frac{1}{n^2}\right)$$

$$p_j \geq \frac{n-j-1}{n-j+1} \qquad \frac{n-2}{n} \times \frac{n-3}{n-1} \times \ldots \qquad p = \frac{1}{2^n}$$

proba d'obtenir la coupe minimum C

- comme $Pr[A \cap B] = Pr[A|B] \times Pr[B]$, on obtient $p = Pr[\bigcap_{i=1}^{n-2} E_i] = \prod_{i=1}^{n-2} p_j$
- 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 $p = Pr[\bigcap_{i=1}^{n-2} E_i] = \prod_{i=1}^{n-2} p_j$
- 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

p $\frac{1}{p}$ essais

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

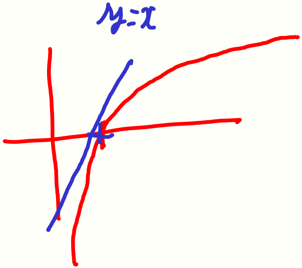
proba d'obtenir la coupe minimum C

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

$e \approx 2,7$ $\quad \frac{1}{e} \sim 0,37$

$$\frac{1}{p} \log(1-p) \leq -1$$

$$\log(1-p) \leq -p$$



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 $\Rightarrow (1-p)^{\frac{1}{p}} \leq 1/e$
- on fait donc $t = \frac{\log n}{p}$ executions de l'algo et on obtient $p_{echec} \leq (1/e)^{\log n} = 1/n$

$$(1-p)^{\frac{\log n}{p}} = \left((1-p)^{\frac{1}{p}}\right)^{\log n}$$

$$\leq \left(\frac{1}{e}\right)^{\log n} = \frac{1}{e^{\log n}} = n$$

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 $\Rightarrow (1-p)^{\frac{1}{p}} \leq 1/e$
- on fait donc $t = \frac{\log n}{p}$ 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 $1 - \frac{1}{n} \to 1$ quand $n \to +\infty$

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 $O(f(n)\ n^2 \log n)$
- Complexite de Contraction ?
  - boucle principale : $\underline{O(n) \text{ fois}}$ n-2 contractions
  - ligne 5 : $O(1)$
  - ligne 6 : $O(n)$
  - ligne 7 : $O(n)$

complexite totale : $\underline{O(n^2)}$

Conclusion de l'algo proba pour coupe minimum

- Algo avec proba de succes $p \geq 1 - \frac{1}{n}$ et complexite $\underline{O(n^4 \log n)}$
- rien n'empeche de :

  0,63

  - ne faire que $t = \frac{n(n-1)}{2}$ executions : $p \geq 1 - \frac{1}{e}$ et complexite $O(n^4)$

    < 10

  - 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 $O(.)$)
  - faire $t = n \cdot \frac{n(n-1)}{2}$ executions : $p \geq 1 - \frac{1}{e^n}$ et complexite $O(n^5)$

<u>Questions subsidiaires :</u> C $p \geq \frac{1}{n^2}$

- 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

10 Ok

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 (peridurale)