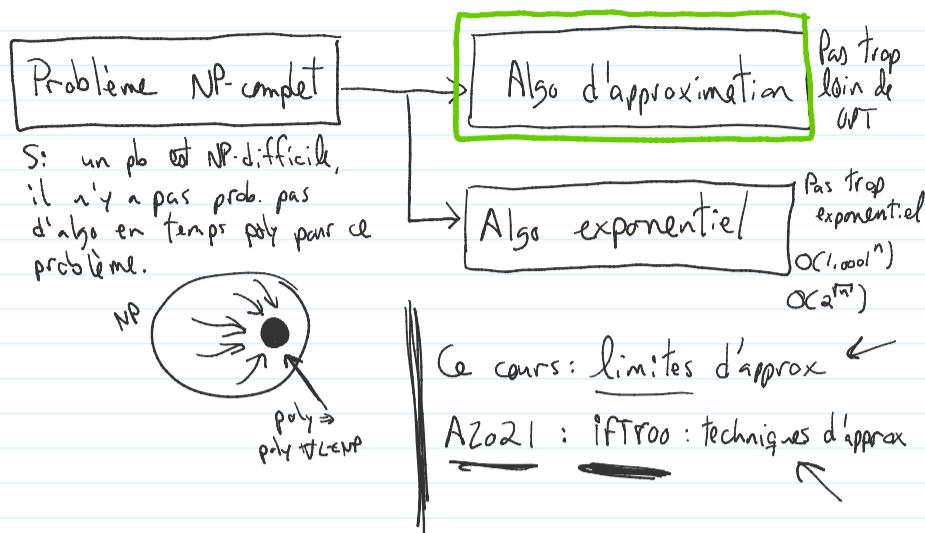


# Inapproximabilité

19 mars 2021

Examen final : en ligne (3h à distance)

si problème : dites-nous le



## Algos d'approx

Problème d'optimisation

Entrée : ensemble d'objets  $\Omega$

contraintes  $C$  sur la faisabilité d'une solution

critère d'optimisation  $\alpha$  (fct sol → val)

Sortie : solution faisable selon  $C$

et qui maximise (ou minimise)  $\alpha$

ex: MAX-CLIQUE

Entrée : graphe  $G$

Sortie :  $X \subseteq V(G)$  tel que  $X$  est une clique et  $X$  est de taille maximum

$$\Omega = \{G\}$$

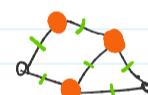
$$C = \{X \subseteq V(G) : X \text{ est une clique}\}$$

$$\alpha = \max |X|$$

MIN-VERTEX-COVER

Entrée : graphe  $G$

Sortie :  $X \subseteq V(G)$  tel que  $\forall u \in E(G), u \in X$  ou  $v \in X$  qui minimise  $|X|$ .

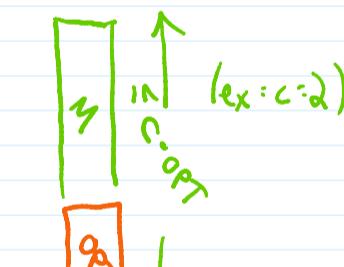


(ens. sommets qui touchent chaque arête)

Soit  $\mathcal{A}$  un problème de minimisation  $\mathcal{A} = \{\Omega, C, \alpha\}$

Un algo  $M$  est une  $c$ -approximation pour  $\mathcal{A}$  si: ( $c \geq 1$ )

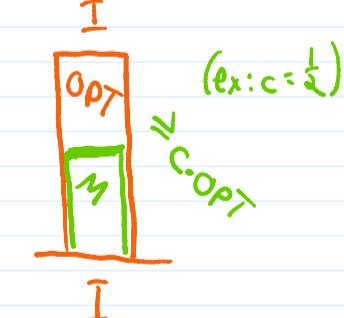
- $M$  s'exécute tjs en temps polynomial
- t'instance  $I$  de  $\mathcal{A}$ , si  $OPT(I)$  est la val. optimale de  $I$ , alors  $M$  retourne une solution faisable de valeur au plus  $c \cdot OPT(I)$ .



Soit  $\beta$  un problème de maximisation ( $c \leq 1$ )

$M$  est une  $c$ -approx. pour  $\beta$  si:

- $M$  s'exéc. en temps poly
- t'instance  $I$ ,  $M$  retourne une sol faisable de valeur au moins  $c \cdot OPT$



ex: MAX-SAT

$$x_1 \vee \bar{x}_2 \vee x_3 \vee x_4$$

Entrée: clauses booléennes  $\varphi = C_1 \wedge C_2 \wedge C_3 \wedge \dots \wedge C_m$

Sortie: assignation qui maximise le # de clauses satisfaites

Voici une  $\frac{1}{2}$ -approx: (on satisfait  $\geq \frac{1}{2} OPT(\varphi)$  clauses)

maxsat( $\varphi$ )

soit  $A$  l'assignation où  $x_1 = x_2 = \dots = x_n = T$

maxsat( $\varphi$ )

Soit  $A$  l'assignation où  $x_1 = x_2 = \dots = x_n = T$   
si  $A$  satisfait  $\geq m/2$  clauses  
return  $A$   
sinon  
soit  $\bar{A}$  l'assign. où  $x_1 = x_2 = \dots = x_n = F$   
return  $\bar{A}$

C'est une  $\frac{1}{2}$ -approx car:

→ si  $\text{OPT}(\varphi)$  le # max de clauses satisfaisables  
soit  $m$  le # de clauses de l'entrée

Il est évident que  $\text{OPT}(\varphi) \leq m$ .  $b = m$

Si l'algorithme retourne  $A$ , on sait que  $A$  satisfait  $\geq \frac{1}{2}m$ .  
Dans ce cas, notre  $\bar{A}$  satisfait  $\geq \frac{1}{2}m \geq \frac{1}{2}\text{OPT}(\varphi)$  clauses. ✓

Sinon, l'algorithme retourne  $\bar{A}$ . Cette dernière satisfait toute clause non-satisfait par  $A$ .

Puisque  $A$  sat.  $< \frac{1}{2}m$  clauses,  $\bar{A}$  sat.  $\geq \frac{1}{2}m$  clauses.

⇒  $\bar{A}$  satisfait  $\geq \frac{1}{2}m > \frac{1}{2}\text{OPT}(\varphi)$  clauses. ✓

Idee: ① Trouver une bonne  $b$  sur  $\text{OPT}$  ( $\text{OPT} \leq m$ )

② Comparer la sat. de notre algo à  $b$

Question: est-ce qu'on pourrait faire mieux que  $\frac{1}{2}$ ?

$\frac{3}{4}$ -approx

$\frac{7}{8}$ -approx

ideal:  $(1 - \varepsilon)$ -approx  $\nexists \varepsilon < 1$ ?



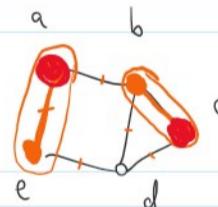
Réponse partielle:  $\frac{7}{8}$  est faisable si chaque clause a  $\geq 3$  var (IFPQO)

Conjecture:  $\frac{7}{8}$  est le mieux possible si  $P \neq NP$ .

## MIN-VERTEX-COVER

$\min VC(G)$

\*  $X = \emptyset$   
done = false  
tant que not done  
| si  $\exists u, v \in E(G)$  t.q.  $u \notin X, v \notin X$   
| |  $X \leftarrow X \cup \{u, v\}$   
| | sinon  
| | | done = true  
| return  $X$



$$X = \{a, e, b, c\}$$



$\min VC$  est une  $\frac{1}{2}$ -approx

Soit  $u_1, v_1, u_2, v_2, \dots, u_k, v_k$  la séquence d'arêtes considérées par l'algorithme à la ligne \*.

On a  $X = \{u_1, v_1, u_2, v_2, \dots, u_k, v_k\}$ .

On note que si  $u_i, v_i$  de la séquence,  $\text{OPT}$  doit contenir  $u_i$  ou  $v_i$  (ou les 2).

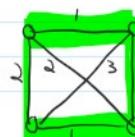
⇒  $\text{OPT} \geq |X| \frac{1}{2} \Rightarrow |X| \leq 2 \cdot \text{OPT} \Rightarrow$  l'algorithme est une  $\frac{1}{2}$ -approx.

Min TSP n'est pas  $c$ -approximable si  $P \neq NP$   
 $c$  constante

T Traveling Salesperson Problem

Entrée: graphe  $G$ , coût  $w: V \times V \rightarrow \mathbb{R}^{>0}$

Sortie: cycle qui passe 1 fois par sommet et qui minimise la somme des coûts des arêtes.



$$\text{coût} = 1 + 2 + 1 + 2 = 6$$

Sous-problème: cycle qui passe 1 fois par sommet et qui minimise la somme des coûts des arêtes.

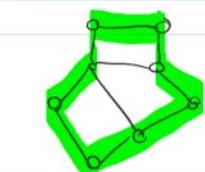
$$\text{coût} = 1+2+1+2 = 6$$

On va montrer H constante  $c \geq 1$ , si on avait une  $c$ -approx, on aurait  $P = NP$ .



Langage HAMCYCLE = { $G$ : cycle qui passe 1 fois par sommet}

Hamiltonien



Thm: HAMCYCLE est NP-complet.

Thm: MinTSP n'a pas de  $c$ -approx (si  $P \neq NP$ )

// Si on avait une  $c$ -approx à MinTSP, on pourrait décider HAMCYCLE

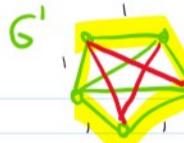
en temps poly

Supposons que MinTSP admet une  $c$ -approx. ( $c$  est connu)

Soit  $G$  une instance de HAMCYCLE. On crée une instance  $\langle G', w \rangle$  de MinTSP comme suit:

$$V(G') = V(G)$$

$$w(uv) = \begin{cases} 1 & \text{si } uv \in E(G) \\ (cn)^{1000} & \text{sinon} \end{cases}$$



On montre comment ceci sert à décider HAMCYCLE.

- Si  $G \in \text{HAMCYCLE}$ , alors  $\text{OPT}(G') = n$  car un cycle ham. de  $G$  n'entreprend que des arêtes de coût 1 dans  $G'$ .

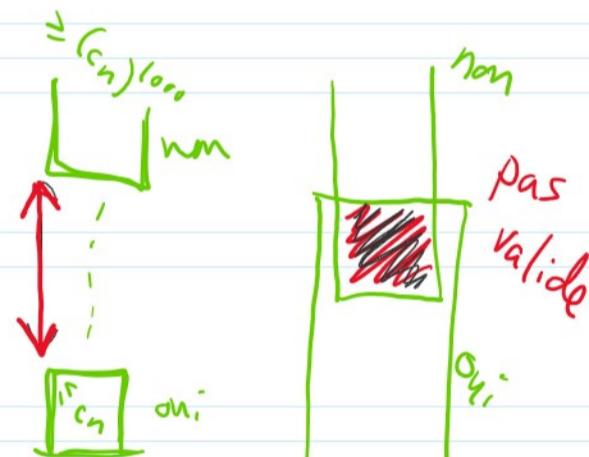
- Si  $G \notin \text{HAMCYCLE}$ , alors tout cycle Ham. de  $G'$  entreprend au moins une arête de coût  $(cn)^{1000}$ .

Dans  $\text{OPT}(G') \geq (cn)^{1000}$

→ | Dichotomie:  $\begin{cases} G \in \text{HAMCYCLE} \Rightarrow \text{OPT}(G') = n \\ G \notin \text{HAMCYCLE} \Rightarrow \text{OPT}(G') \geq (cn)^{1000} \end{cases}$

La  $c$ -approx pour MinTSP hypothétique peut alors nous permettre de distinguer si  $G \in \text{HAMCYCLE}$  ou non.

→ [ Si  $G \in \text{HAMCYCLE}$ , l'approx retourne une sol.  $\leq c \cdot \text{OPT}(G') = cn$   
Si  $G \notin \text{HAMCYCLE}$ , l'approx retourne une sol.  $\geq \text{OPT}(G') \geq (cn)^{1000}$  ]



[ Puisque  $(cn)^{1000} > cn$ , la  $c$ -approx nous permettrait de distinguer les 2 situations. On pourrait ainsi décider HAMCYCLE en temps poly avec l'algorithme ]

sur entrée  $G$

générer  $\langle G', w \rangle$

$k = c \cdot \text{Approx}(G')$

si  $k \leq cn$ , accepter  
si  $k \geq (cn)^{1000}$ , rejeter

Ce qui contredit l'hypothèse  $P \neq NP$ . ■

Note contre-intuitive: MaxTSP admet une 2-approx

Langage "gap": langage avec une promesse sur chaque instance  $I$ :

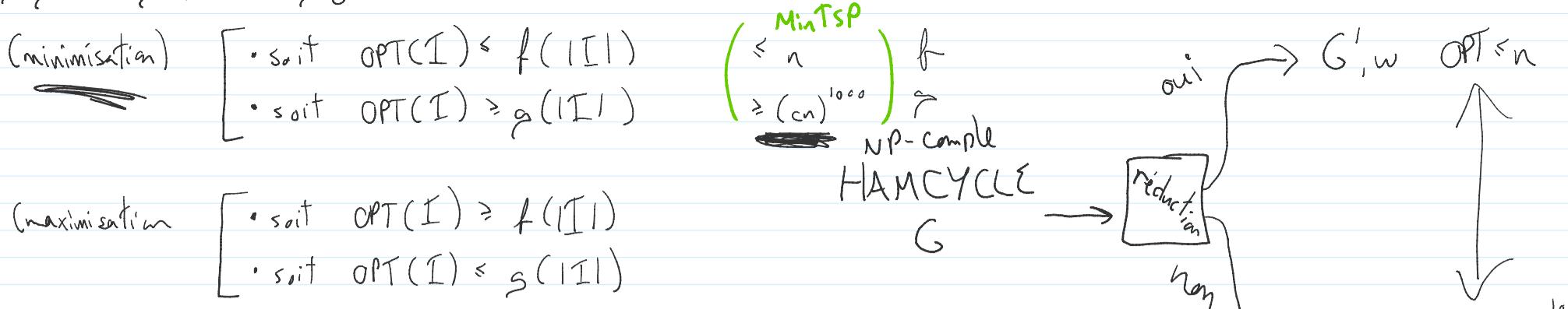
(minimisation)  $\exists$  soit  $\text{OPT}(I) \leq f(|I|)$

$\leq n$

$f$

MinTSP

:  $\rightarrow G', w$   $\text{OPT} \leq n$



Les  $\langle G', w \rangle$  générés par la réduction ci-haut forment le langage gap  $\text{MinTSP}$  avec

$$\begin{aligned} f(|I|) &= n \\ g(|I|) &= (cn)^{1000} \end{aligned}$$

[Intuition] : si un pb. d'optimisation A a une version "langage gap" qui permet de résoudre un pb NP-difficile en temps poly, alors le problème A n'admet pas de  $\frac{g(|I|)}{f(|I|)}$ -approximation.

Les instances créées par cette réduction forment un langage gap pour min-TSP

- Un langage gap A (disons de minimisation) est appelé difficile s'il existe un langage L NP-complet et une fct

$$t: \Sigma^* \rightarrow A \quad \text{telle que} \quad \begin{aligned} \bullet \quad w \in L &\Rightarrow OPT(t(w)) \leq f(|t(w)|) \\ \bullet \quad w \notin L &\Rightarrow OPT(t(w)) \geq g(|t(w)|) \end{aligned}$$

[Le "gap" généré par t est  $\frac{g(|t(w)|)}{f(|t(w)|)} = \frac{(cn)^{1000}}{n}$  (dans MinTSP)]

Idée: le gap généré détermine le meilleur ratio d'approx permis

Donc MinTSP n'admet pas de  $\frac{(cn)^{1000}}{n} = c n^{1000/999}$ -approx

- Importance de la réduction MinTSP: pas tous les pb qui sont approximables à un ratio arbitrairement près de 1. (si P=NP)
- MAX-SAT est-il inapproximable?

Thm:  $\exists$  constante  $p < 1$  t.q. MAX-SAT n'admet pas de p-approx. si  $P \neq NP$  ( $\sim 1995$ )

Thm équivalent: le langage gap MAX-SAT-p est difficile, où

$$\text{MAX-SAT-}p = \left\{ \varphi : \begin{array}{l} \text{on peut satisfaire toutes les m clauses} \\ \text{ou} \\ \text{on peut satisfaire} \leq pm \text{ clauses} \end{array} \right\}$$

[On utilise ce Thm en boîte noire]

Thm:  $\exists$  constante  $\alpha < 1$  t.q. MAX-CLIQUE n'a pas de  $\alpha$ -approx., si  $P \neq NP$ .

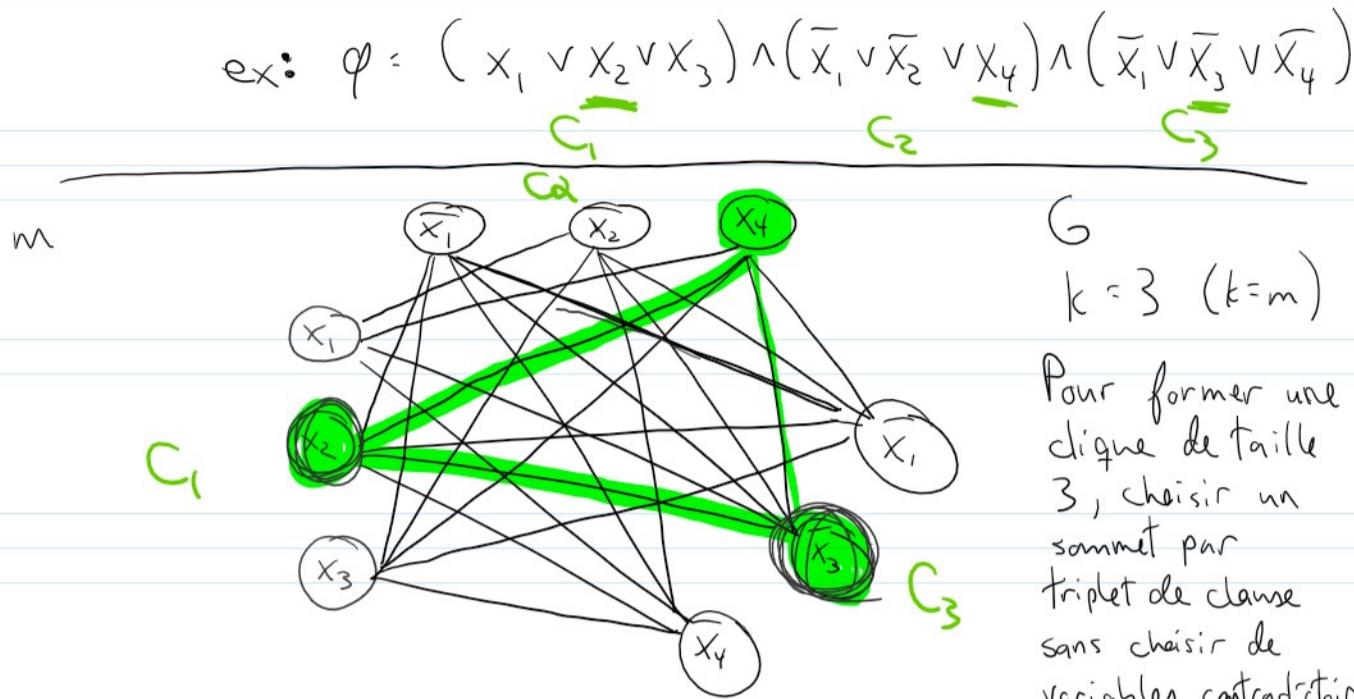
Soit  $\varphi$  une instance de MAX-SAT-p.

$$\varphi \xrightarrow{\substack{OPT=m \\ OPT < pm}}$$

[On transforme  $\varphi$  en une instance G de MAX-CLIQUE et on va trouver un  $\alpha$  t.q. une  $\alpha$ -approx pour MAX-CLIQUE

et on va trouver un  $\alpha$  t.q. une  $\alpha$ -approx pour MAX-CLIQUE permettrait de distinguer si  $\varphi$  à  $m$  clauses satif., ou  $< pm$  clauses sat.

On reprend la réduction classique de SAT vers CLIQUE.



Soit  $G$  le graphe généré par cette réduction, soit  $\text{OPT}(G)$  la taille de sa + grosse clique.

① Si on peut sat.  $m$  clauses de  $\varphi$ , alors  $\underline{\text{OPT}(G) = m}$ .

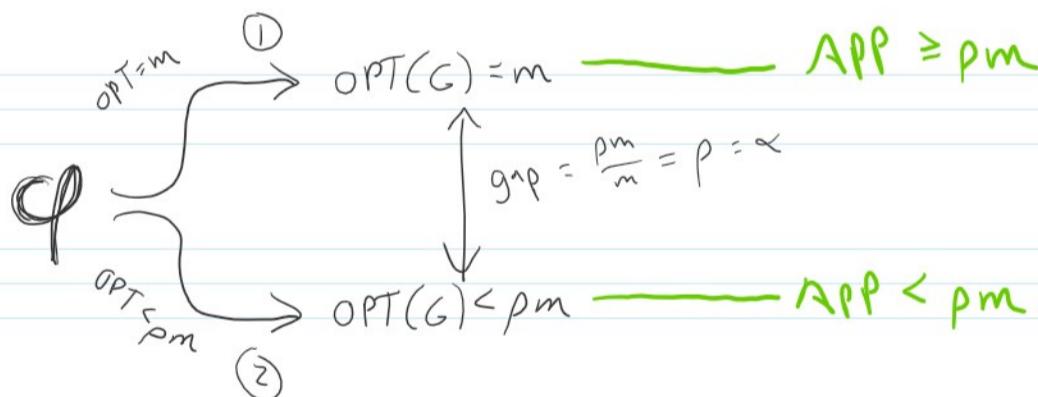
② À montrer: si on peut sat.  $< pm$  clauses de  $\varphi$ , alors  $\underline{\text{OPT}(G) < pm}$ .

Soit  $X$  une clique max. de  $G$ . Donc  $X$  contient un sommet-variable pour  $|X|$  clauses distinctes  $\Rightarrow X$  correspond à une assign. partielle qui satisf.  $\geq |X|$  clauses de  $\varphi$ .

$$|X| = \text{OPT}(G) \Rightarrow \text{OPT}(\varphi) \geq |X|$$

$$\text{contra. } \text{OPT}(\varphi) < |X| \Rightarrow \text{OPT}(G) < |X|$$

$$[\text{OPT}(\varphi) < pm \Rightarrow \text{OPT}(G) < pm]$$



Ceci veut dire que si on avait une  $p$ -approx ( $\alpha = p$ ) pour MAX-CLIQUE, on pourrait distinguer l'instance  $\varphi$  comme suit:

sur entrée  $\varphi$

générer le  $G$  ci-haut



APP: résultat de la  $p$ -approx pour  $G$

si  $\text{APP} \geq pm$ ,  $\varphi$  a  $m$  clauses satif.

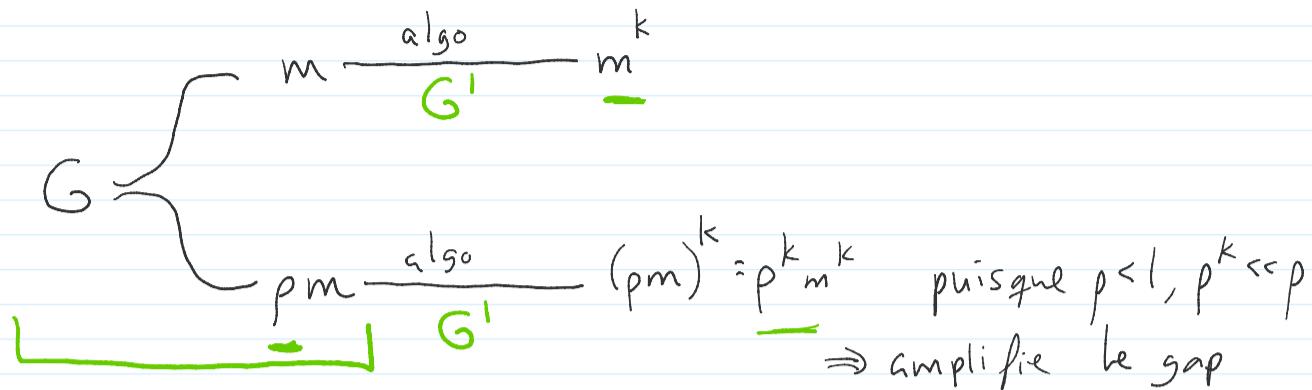
sinon  $\varphi$  peut satif.  $< pm$  clauses. □

Thm:  $\forall$  constante  $\alpha$ , MAX-CLIQUE n'admet pas de  $\alpha$ -approx. si  $P \neq NP$

On va utiliser un lemme en boîte noire!

On va utiliser un lemme en boîte noire :

Lem:  $\exists$  un algo en temps poly qui, sur entrée  $G$  et  $k \in \mathbb{N}$ , donne en sortie un graphe  $G'$  tel que si  $G$  a une clique max de taille  $\underline{\omega}$ , alors  $G'$  a une clique max de taille  $\underline{\omega^k}$ .

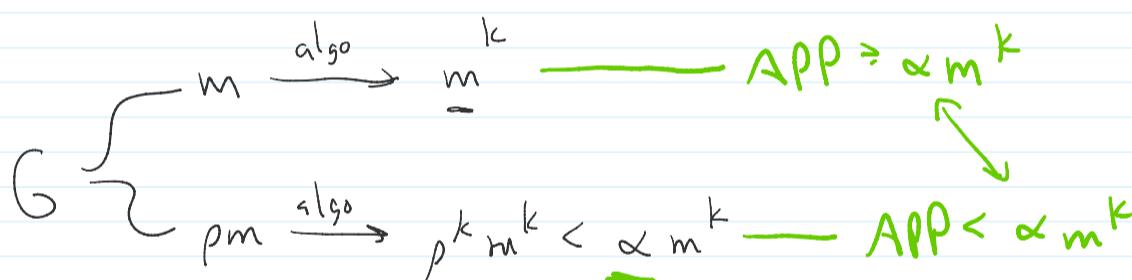


On démarre donc d'une instance gap de MAX-CLIQUE, i.e.

G t.q. soit  $G$  a une clique de taille  $m$ ,  
ou une clique max de taille  $\leq pm$ .

On génère  $G'$  en passant  $G$  à l'algo mystère avec un  $k$  à déterminer.

- Supposons que MAX-CLIQUE admet une  $\alpha$ -approx. ( $\alpha$  = constante)
- On choisit n'importe quel  $k$  tel que  $p^k < \alpha$ .
- Ce  $k$  est une constante car  $p < 1$  et  $\alpha < 1$  sont des constantes.



La  $\alpha$ -approx. permettrait de distinguer si  $G$  a une clique  $m$  ou  $\leq pm$

sur entrée  $G$   
trouver  $k$  t.q.  $p^k < \alpha$  ( $k < \frac{\log \alpha}{\log p}$ )

générer  $G'$  avec l'algo

APP = résultat de la  $\alpha$ -approx sur  $G'$

si  $\text{APP} \geq \alpha m^k$ ,  $G$  a une clique de taille  $m$

sinon,  $G$  a une clique max.  $\leq pm$



ex: MAX-INDSET n'admet pas de  $\alpha$ -approx  $\forall \alpha < 1$ , si  $P \neq NP$ .

Thm: MAX-CLIQUE n'admet pas de  $\frac{1}{n^\varepsilon}$ -approx  $\forall \varepsilon < 1$ .

pas de  $\frac{1}{n^{1/2}}$ -approx.

pas de  $\frac{1}{n^{99/100}}$ -approx