

Inapproximabilité

29 mars 2021

Examen final : en ligne (3h à distance)

si problème : dites-nous le



Algos d'approx

Problème d'optimisation

Entrée: ensemble d'objets O
contraintes C sur la faisabilité d'une solution
critère d'optimisation α (fct sol \rightarrow val)

Sortie: solution faisable selon C
et qui maximise (ou minimise) α

ex: MAX-CLIQUE

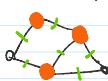
Entrée: graphe G
Sortie: $X \subseteq V(G)$ tel que X est une clique
et X est de taille maximum

$$\begin{aligned} O &= \{G\} \\ C &= \{X \subseteq V(G) : X \text{ est une clique}\} \\ \alpha &= \max |X| \end{aligned}$$

MIN-VERTEX-COVER

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

(ens. sommets qui touchent chaque arête)



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

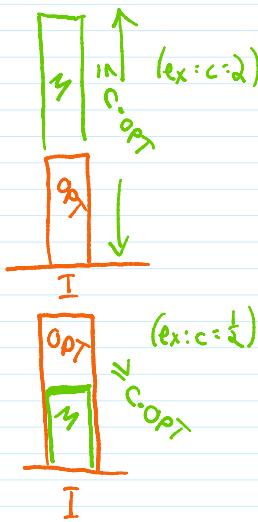
Un algo M est une c -approximation pour \mathcal{A} si: ($c \geq 1$)

- M s'exécute tjs en temps polynomial
- \forall 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 β un problème de maximisation ($c \leq 1$)

M est une c -approx. pour β si:

- M s'exécute en temps poly
- \forall instance I , M retourne une sol faisable de valeur au moins $c \cdot OPT$



$$\boxed{v_1 v_2 v_3 v_4}$$

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 $\geq \frac{1}{2} \text{OPT}(\varphi)$ clauses)

$\text{maxsat}(\varphi)$

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 sol. 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 $\geq \frac{1}{2}m \geq \frac{1}{2}\text{OPT}(\varphi)$ clauses.

Idee: ① Trouver une bonne b sur 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 ≥ 3 var ('FT800')

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

MIN-VERTEX-COVER

$\min \text{VC}(G)$

$X = \emptyset$

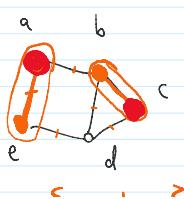
done = false

Tant que non done

| si $\exists uv \in E(G)$ t.g. $u \notin X, v \notin X$

| | $X \leftarrow X \cup \{u, v\}$

| simm.



* tant que non arrêté
 si $\exists uv \in E(G)$ t.g. $u \notin X, v \notin X$
 $X \leftarrow X \cup \{u, v\}$
 sinon
 done = true
 return X



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

minVC est une 2-approx

Soit $u_1v_1, u_2v_2, \dots, u_kv_k$ la séquence d'arêtes considérées par l'algo à la ligne *.
 On a $X = \{u_1, v_1, u_2, v_2, \dots, u_k, v_k\}$.
 On note que si uv de la séquence, OPT doit contenir u_i , ou v_i (au moins 2).

$$\Rightarrow OPT \geq \lceil X \rceil / 2 \Rightarrow |X| \leq 2 \cdot OPT \Rightarrow \text{l'algo est une 2-approx.}$$

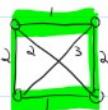
Min TSP n'est pas c -approximable si $P \neq NP$

T Traveling Salesperson Problem

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

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

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

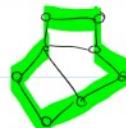


$$\text{cost} = 1+2+1+2 = 6$$



Langage HAMCYCLE = $\{G: \exists \text{cycle qui passe } | \text{ fois par sommet}\}$

Hamiltonien



cycle 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

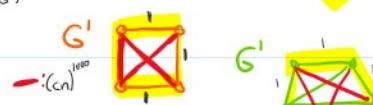
en temps poly

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

Soit G une instance de HAMCYCLE. On crée une instance (G', w) 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 $(cn)^{1000}$.

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

$$G \in \text{HAMCYCLE} \Rightarrow \text{OPT}(G') = n$$



$$\stackrel{?}{=} G$$

n

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

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

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

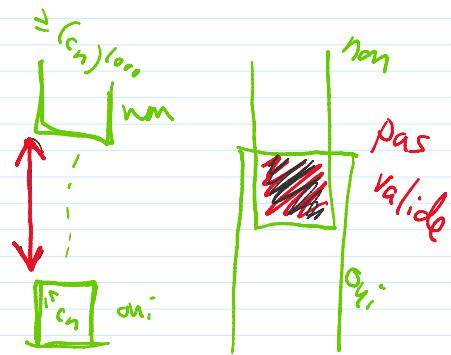
générer $\langle G', w \rangle$

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

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 :

$$(\text{minimisation}) \quad \begin{cases} \text{soit } \text{OPT}(I) \leq f(|I|) \\ \text{soit } \text{OPT}(I) \geq g(|I|) \end{cases} \quad \begin{array}{l} \text{MinTSP} \\ (\leq n) \\ (\geq (cn)^{1000}) \end{array}$$

$$(\text{maximisation}) \quad \begin{cases} \text{soit } \text{OPT}(I) \geq f(|I|) \\ \text{soit } \text{OPT}(I) \leq g(|I|) \end{cases}$$

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

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

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.

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

$t: \Sigma^* \rightarrow A$ telle que

- $w \in L \Rightarrow \text{OPT}(t(w)) \leq f(|t(w)|)$
- $w \notin L \Rightarrow \text{OPT}(t(w)) \geq g(|t(w)|)$

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

Idee: le gap généré détermine le meilleur ratio d'approx permisble
 Donc MinTSP n'admet pas de $\frac{(cn)^{1000}}{n} = cn^{999}$ -approx HC