IFT503/711 - Exercices sur l'approximation/ETH

Manuel Lafond

Hiver 2020

Exercice 1

Montrez que P $=$ PCP$(0, \log n)$.

Solution

(1) P $\subseteq$ PCP$(0, \log n)$.

Soit $L \in$ P et M qui décide L. On peut traiter M comme un vérificateur qui utilise 0 bits aléatoires et consulte 0 bits d'un certificat. Puisque $0 \in O(\log n)$, M est un vérificateur $(0, \log n)$-PCP.

(2) PCP$(0, \log n) \subseteq$ P.

Soit $L \in$ PCP$(0, \log n)$ et soit V un vérificateur $(0, \log n)$-PCP pour L. Soit $w \in \Sigma^*$, et soit c un certificat. On note que V est entièrement déterministe et consulte toujours les mêmes $d \log n$ bits de c pour une constante d. On peut donc supposer que $|c| = d \log n$ (car des bits additionnels ne seraient jamais lus de toute façon). Donc, pour tout certificat c, on a $c \in \{0, 1\}^{d \log n}$, et qu'il y a $2^{d \log n} = n^d$ certificats possibles (ce qui est polynomial).

Soit M la MT qui, sur entrée w, fait :

- pour chaque $c \in \{0, 1\}^{d \log n}$:
-        simuler V sur $\langle w, c\rangle$
-        si V accepte, Accepter
- Rejeter

Notez que M prend un temps polynomial car le nombre de c est polynomial, et V s'exécute en temps polynomial sur chaque entrée.

Si $w \in L$, alors $\exists c$ tel que V accepte $\langle w, c\rangle$, et M acceptera car elle énumère tous les c. Si $w \notin L$, alors $\forall c$, $Pr[V$ accepte $\langle w, c\rangle] \leq 1/2$. mais puisque V est entièrement déterministe, $Pr[V$ accepte $\langle w, c\rangle] \in \{0, 1\}$. Quand $w \notin L$, on a donc $Pr[V$ accepte $\langle w, c\rangle] = 0$, et donc M rejettera.

On a démontré que $L \in$ P.

Exercice 2

Montrez que PCP($poly(n), 1$) $\subseteq$ NEXPTIME.

Solution

Soit $L \in$ PCP($poly(n), 1$) et V un vérificateur ($poly(n), 1$)-PCP. Donc V utilise au plus dn^k lancers de pièces, pour des constantes d et k. Le comportement de V sur entrée $\langle w, c\rangle$ est donc déterminé par une chaîne r dans $\{0,1\}^{dn^k}$, qui correspond aux choix aléatoires faits par V (ou fournis par l'oracle, si vous préférez le voir de cette façon). Il y a 2^{dn^k} choix possibles pour r.

Pour chaque r, V consulte $O(1)$ bits de certificat. Supposons que V consulte au plus b bits, où b est une constante. On peut donc supposer que tout certificat est de taille au plus $b \cdot 2^{dn^k}$, car des bits additionnels ne seraient jamais lus. On peut donc toujours deviner un tel certificat en temps exponentiel.

On peut donc concevoir une MT M non-déterministe qui, sur entrée w, fait :

- deviner un certificat c de façon non-déterministe
- pour chaque $r \in \{0,1\}^{dn^k}$:
-     simuler V sur $\langle w, c\rangle$ lorsque ses choix aléatoires sont r
- si V a toujours accepté, Accepter
- Rejeter

On peut imaginer plusieurs façons de forcer les choix r pour V, par exemple en "surchargeant" l'oracle aléatoire. Notez que M prend un temps non-déterministe exponentiel car elle devine c en temps $O(|c|) = O(2^{dn^k})$, puis énumère 2^{dn^k} choix de r, et pour chaque choix, V prend un temps polynomial.

Si $w \in L$, alors $\exists c$ tel que V accepte pour tout r. Dans ce cas, M devinera c et acceptera. Si $w \notin L$, alors $\forall c$, V rejettera sur au moins la moitié des r. Dans ce cas, M rejettera peut importe le c deviné.

Donc, $L \in$ NEXPTIME.

Exercice 3

Montrez que MAX-CUT admet une $\frac{1}{2}$-approximation.
Indice : démarrez avec une bipartition arbitraire. Si déplacer un sommet améliore le nombre d'arêtes traversante, déplacez-le. Répétez jusqu'à ce que vous ne puissiez plus. Argumentez que la solution alors obtenue donne un $\frac{1}{2}$-approximation.

Solution

Sur entrée G, considérez l'algorithme suivant :

- choisir une bipartition aléatoire (V_1, V_2)
- fini = false
- tant que pas fini
- s'il existe $v \in V_1$ tel que $(V_1 \setminus \{v\}, V_2 \cup \{v\})$ est une meilleure coupe
- $(V_1, V_2) \leftarrow (V_1 \setminus \{v\}, V_2 \cup \{v\})$
- sinon s'il existe $v \in V_2$ tel que $(V_1 \cup \{v\}, V_2 \setminus \{v\})$ est une meilleure coupe
- $(V_1, V_2) \leftarrow (V_1 \cup \{v\}, V_2 \setminus \{v\})$
- sinon
- fini = true
- retourner (V_1, V_2)

Par "meilleure coupe", on veut dire que le nombre d'arêtes donc les bouts sont dans des ensembles différents augmente.

Montrons que ceci est une $\frac{1}{2}$-approximation. Soit OPT le nombre maximum d'arêtes dans une coupe. Soit (V_1, V_2) la coupe retournée par l'algorithme, et soit APP le nombre d'arêtes dans cette coupe.

Soit $m = |E(G)|$. Il est évident que $OPT \leq m$. Comparons maintenant APP avec cette borne sur OPT. Soit $v \in V_1$ et n_v le nombre de voisins de v dans G. De plus, soit n_1 son nombre de voisins dans V_1 et n_2 son nombre de voisins dans V_2 (donc $n_v = n_1 + n_2$). Si $n_1 > n_2$, alors envoyer v dans V_2 augmenterait la taille de la coupe. On peut donc supposer que $n_1 \leq n_2$, i.e. que la moitié des voisons de v sont de l'autre côté. Cette propriété est aussi vraie pour tout $v \in V_2$.

Le nombre d'arêtes dans la coupe est donc au moins

$$APP \geq \frac{1}{2} \sum_{v \in V(G)} \frac{1}{2} n_v = \frac{1}{2} \cdot \frac{1}{2} \sum_{v \in V(G)} n_v$$

On doit diviser par deux car chaque arête est comptée deux fois dans cette somme, une fois pour chaque bout. D'un autre côté, le nombre d'arêtes dans un graphe est

$$m = \frac{1}{2} \sum_{v \in V(G)} n_v$$

Ceci est parce qu'en comptant la somme des nombre de voisins, on compte chaque arête exactement deux fois. Au final, on a

$$APP \geq \frac{1}{2} \cdot \frac{1}{2} \sum_{v \in V(G)} n_v = \frac{1}{2} m \geq \frac{1}{2} OPT$$

et on a donc une $\frac{1}{2}$-approximation.

Exercice 4

On sait qu'il existe une constante $\rho < 1$ telle que pour MAX-3-SAT, il est difficile de déterminer si une instance est satisfaisable, ou si on peut satisfaire au plus une fraction ρ de ses clauses. Montrez qu'il existe une constante α telle que MAX-CLIQUE n'admet pas de α-approximation.

Solution

Nous ramenons la réduction classique de 3-SAT vers CLIQUE, mais ici en version problème d'optimisation.

Soit ϕ une instance de MAX-3-SAT sur variables $x_1, \ldots, x_n$ et clauses $C_1, \ldots, C_m$. On génère une instance G de MAX-CLIQUE comme suit : on ajoute dans $V(G)$ un sommet pour chaque litéral apparaissant dans une clause (par exemple, pour chaque clause $C_i = (x_1 \vee \overline{x_2} \vee x_3)$, on ajoute un sommet pour cette occurrence particulière de x_1, un autre sommet pour le $\overline{x_2}$ et un sommet pour le x_3). Donc, G a $3m$ sommets. Ensuite, pour chaque paire de litéraux x_i et x_j provenant de deux clauses différentes (ces litéraux peuvent être négatifs ou positifs), on ajoute une arête entre x_i et x_j si l'un n'est pas la négation de l'autre (donc pas d'arête si $x_j = \overline{x_i}$).

Si ϕ est satisfaisable, alors G admet une clique de taille m (ceci a déjà été argumenté dans la réduction classique de SAT vers CLIQUE). Supposons qu'on peut plutôt satisfaire au plus ρm clauses de ϕ. Considérez la plus grande clique C de G. Cette clique contient au plus un sommet par triplet correspondant à une clause, et les litéraux de cette clique ne se contradisent pas. Donc, C correspond à une assignation qui satisfait $|C|$ clauses. Il s'ensuit que $|C| \leq \rho m$. Il est donc difficile de déterminer si G a une clique de taille m, ou une clique de taille ρm. Donc, si P $\neq$ NP, il ne peut pas y avoir d'approximation meilleure que ρ, car sinon on pourrait distinguer les deux cas.

Exercice 5

Soit $G = (V, E)$ un graphe. Le *produit* de G avec lui-même est le graphe dénoté par $G \times G$ dans lequel $V(G \times G) = V \times V$ et

$$E(G \times G) = \{\{(a, b), (x, y)\} : (ax \in E \text{ ou } a = x) \text{ et } (by \in E \text{ ou } b = y)\}$$

On écrit $G^k = G \times \ldots \times G$ (k fois). Il est possible de montrer que pour tout entier k, G a un clique de m sommets $\iff$ G^k a une clique de m^k sommets. (suggestion : démontrez-le !)

En utilisant ce fait, montrez que si P $\neq$ NP, MAX-CLIQUE n'admet pas de c-approximation pour tout $c < 1$.
Suggestion : transformez des instances gap de l'exercice précédent pour créer des instances avec un gap amplifié.

Solution

On sait qu'il est difficile de distinguer si un graphe G a une clique de taille m, ou bien une clique de taille au plus ρm pour une certaine constante ρ. Supposons que G admet une c-approximation pour une certaine constante $c \leq 1$.

Soit G une instance de MAX-CLIQUE. On génère l'instance G^k de MAX-CLIQUE (donc une réduction de MAX-CLIQUE vers MAX-CLIQUE).

Selon la question, si G a une clique de taille m, alors G^k a une clique de taille m^k. Sinon, G a une clique de taille au plus ρm et G^k a une clique de taille au plus $\rho^k m^k$. Le "gap" est donc de taille ρ^k, et il peut être amplifié à n'importe quelle constante (i.e. puisque $\rho < 1$, ρ^k peut être rendu arbitrairement petit en choisissant une constante k appropriée).

On peut donc choisir k tel que $\rho^k < c/2$. Si on a une c-approxmiation M pour MAX-CLIQUE, on peut l'exécuter sur G^k. Si G a une clique de taille m, alors l'approximation retourne une clique de taille au moins cm^k. Si G a une clique de taille au plus ρm, alors M retourne une clique de taille au plus $\rho^k m^k < (c/2)m^k$. Puisque $(c/2)m^k < cm^k$, M peut distinguer si G a une clique de taille m ou ρm, ce qui impliquerait P = NP.

Exercice 6

Montrez que si la ETH est vraie, alors CLIQUE ne peut pas être décidé en temps $2^{o(n)}poly(n)$, où n est le nombre de sommets.

Solution

Soit ϕ une instance de 3-SAT dans laquelle il y a n variables et cn clauses, avec c une constante. La ETH stipule qu'on ne peut décider si ϕ est satisfaisable en temps $2^{o(n)}poly(n)$.

Considérez la réduction classique de SAT vers CLIQUE, la même que celle de l'exercice 4. La réduction transforme ϕ en un graphe G qui contient $3m$ sommets, où m est le nombre de clauses de ϕ. Puisque $m = cn$, G a $3cn$ sommets.

Si un algorithme M peut décider CLIQUE en temps $2^{o(n)}poly(n)$, alors M décide G en temps $2^{o(3cn)}poly(n)$. Puisque $3c$ est une constante, M décide donc G en temps $2^{o(n)}poly(n)$. Rappelons que ce n est le nombre de variables de ϕ. Donc on pourrait transformer ϕ en G et exécuter M pour décider si $\phi \in$ SAT en temps $2^{o(n)}poly(n)$, ce qui contredit la ETH.

Exercice 7

Considérez le problème SET-COVER défini dans la série d'exercices précédente. Pour une instance donnée de SET-COVER, on dénote par n son nombre d'ensembles et par m son nombre d'éléments à couvrir.

Montrez que si la ETH est vraie, alors SET-COVER ne peut pas être décidé en temps $2^{o(n+m)}poly(n)$.

Solution

Dans la série d'exercices précédente, on transformait une instance 3-SAT ϕ, sur n variables et m clauses, en une instance SET-COVER avec ensembles $\mathcal{S}$ et univers U. Consultez les solutions pour les détails. Le nombre d'ensembles était égal à $2n$ (un ensemble pour x_i et pour $\overline{x_i}$) et le nombre d'éléments égal à m, un élément par clause. On génère donc des instances avec $O(n)$ ensembles et $O(m)$ éléments.

La ETH stipule qu'on ne peut pas résoudre 3-SAT en temps $2^{o(n+m)}poly(n)$. Avec un algorithme M pour résoudre SET-COVER en temps $2^{o(n+m)}poly(n)$, on pourrait utiliser la réduction pour résoudre 3-SAT en temps $2^{o(n+m)}poly(n)$.