IFT503/711 - Exercices sur P et NP
Solutions

Manuel Lafond

Exercice 1

Montrez que P $\subseteq$ NP $\cap$ co-NP.

Solution.

On sait que P $\subseteq$ NP, car s'il y a une MT déterministe M qui accepte un langage $L \in$ P, M peut être vue comme une MT non-déterministe qui n'a qu'un seul choix pour chaque transition. Le langage accepté par M est L, et donc $L \in$ NP.

On montre ensuite que $L \in P \iff \overline{L} \in P$ (dans le jargon, on dit que P est fermé sous le complément). Ceci est relativement facile à voir : une MT qui décide L peut servir directement à déterminer si un mot est dans $\overline{L}$ ou non. Voici les détails.

Soit $L \in$ P, et soit M une MT qui décide L en temps polynomial. Considérez $\overline{L}$. On a $w \in L \iff w \notin \overline{L}$. Donc, on peut prendre la machine M' qui décide $\overline{L}$ de la façon suivante : sur entrée w, simuler M sur entrée w et retourner le résultat inverse. Lorsque $w \in \overline{L}$, M rejettera w et donc M' l'acceptera. Inversément, lorsque $w \notin \overline{L}$, M acceptera w et M' le rejettera. Donc M' décide $\overline{L}$. Puisque simuler M prend un temps polynomial, ceci veut dire que $\overline{L} \in P$.

Puisque L a été choisi arbitrairement dans P, ceci montre que $L \in P \iff \overline{L} \in P$. En particulier, $\overline{L} \in NP$. Puisque le complément de L est dans NP, on a $L \in$ co-NP. Puisque ceci est vrai pour tout $L \in$ P, on a P $\subseteq$ co-NP.

Exercice 2

Soit $L \in$NP. Montrez qu'il existe une constante k telle que $L \in DTIME(2^{n^k})$, où n représente la taille de l'entrée.

Solution.

Si $L \in$ NP, alors il existe une MT non-déterministe M qui accepte précisément les mots de L en temps $O(n^k)$ pour une constante k. Soit c la constante telle que M accepte les mots de L en temps au plus cn^k.

On peut créer une machine déterministe M' qui décide L en implémentant la procédure récursive suivante :
simulerM($w, cfg, nbtrans$)
– si $nbtrans > cn^k$, retourner $faux$
– si cfg est une configuration acceptante, retourner $true$
– pour chaque configuration $cfg2$ que M peut emprunter à partir de cfg :
—- retourner simulerM($w, cfg2, nbtrans + 1$)

On ferait un appel initial avec l'entrée w, cfg étant la configuration initiale de M sur entrée w, et $nbtrans = 0$.

L'idée est que M' explore de façon déterministe tous les branchements non-déterministes de M. Si $w \in L$, alors M a une séquence de configurations acceptante de longueur au plus nc^k, et M' acceptera car une de ses récursions explorera cette séquence. Si $w \notin L$, alors M n'atteint jamais de configuration acceptante et donc M' n'acceptera pas. Donc, M' décide bel et bien L.

Il faut maintenant argumenter que M est en temps $O(2^{n^{k'}})$ pour un certain k'. On note que pour M', le nombre de transitions possibles à partir d'une configuration cfg est borné par une constante d (ceci est parce que le nombre d'états et le nombre de symboles sont des constantes). Ceci veut dire qu'à chaque récursion, M fait au plus d appels récursifs. Donc, dans l'arbre de récursion de M, chaque noeud a au plus d enfants et la profondeur de cet arbre est au plus cn^k. Le nombre d'appels récursifs est donc $O(d^{cn^k})$. Puisque $d = 2^{\log d}$, ceci est $O(2^{\log dcn^k})$ et, puisque $\log dc \in O(1)$, ceci est $O(2^{n \cdot n^k}) = O(2^{n^{k+1}})$.

Exercice 3

Un langage L est co-NP-complet si $L \in$ co-NP, et si, pour chaque langage $A \in$ co-NP, $A \leq_P L$.

Montrez qu'un langage L est NP-complet si et seulement si $\overline{L}$ est co-NP-complet.

Solution.

($\Rightarrow$) : supposons que L est NP-complet. Alors $L \in$ NP, et donc $\overline{L} \in$ co-NP. Maintenant, soit $A \in$ co-NP. On veut montrer que A se réduit à $\overline{L}$. Par définition, on a $\overline{A} \in$ NP, et donc $\overline{A} \leq_P L$ puisque L est NP-complet. Soit f une fonction de réduction polynomiale telle que $w \in \overline{A} \iff f(w) \in L$. Il s'avère qu'on peut aussi utiliser f pour réduire A à $\overline{L}$. C'est-à-dire, on montre que $w \in A \iff f(w) \in \overline{L}$.

Soit $w \in A$. Alors $w \notin \overline{A}$, et donc $f(w) \notin L$, ce qui implique que $f(w) \in \overline{L}$. Dans l'autre sens, soit $w \notin A$. Alors $w \in \overline{A}$, et donc $f(w) \in L$, ce qui implique que $f(w) \notin \overline{L}$. Ceci montre qu'un langage arbitraire dans co-NP se réduit à $\overline{L}$, et donc que $\overline{L}$ est co-NP-difficile.

($\Leftarrow$) : la preuve est presque identique à la précédente, donc nous donnons une version abrégée. Supposons que $\overline{L}$ est co-NP-complet. Alors $\overline{L} \in$ co-NP et $L \in$ NP. Maintenant, soit $A \in$ NP. On veut montrer que A se réduit à L. On a $\overline{A} \in$ co-NP, et donc il existe une réduction f telle que $w \in \overline{A} \iff f(w) \in \overline{L}$. Ceci est équivalent à $w \in A \iff f(w) \in L$. Il s'ensuit que f est une réduction de A vers L.

Exercice 4

Montrez que $A_{NTM} = \{\langle M, x\rangle : M$ encode une MT non-déterministe qui accepte $x\}$ est NP-difficile.

Solution.

Soit $L \in$ NP, et soit M l'encodage d'une MT non-déterministe dont le langage est L. Soit $x \in \Sigma^*$. Notre réduction f transforme x en $f(x) = \langle M, x\rangle$. Puisque M est de taille constante, f peut être exécutée en temps polynomial par rapport à x. On montre que $x \in L \iff f(x) \in A_{NTM}$.

Si $x \in L$, alors M accepte x. Donc, $f(x) = \langle M, x\rangle \in A_{NTM}$. Si $x \notin L$, alors M n'accepte pas x. Donc, $f(x) = \langle M, x\rangle \notin A_{NTM}$. Ceci démontre l'équivalence.

Exercice 5

Soit EXPTIME $= \bigcup_{k\in\mathbb{N}} DTIME(2^{n^k})$. Montrez que NP $\subseteq$ EXPTIME, possiblement en utilisant la notion de certificat.

Solution.

Soit $L \in$ NP, et soit V un vérificateur pour L, c'est-à-dire une MT déterministe telle que $w \in L \iff$ il existe un mot c de taille $O(|w|^k)$, $k \in \mathbb{N}$, tel que V accepte $\langle w, c\rangle$ en temps $O(|w|^d)$, $d \in \mathbb{N}$. Supposons que c est toujours de taille au plus $\alpha|w|^k$, où α est la constante cachée dans la notation O.

On veut montrer que $L \in$ EXPTIME. Soit la MT M qui, sur entrée w, simule V sur toutes les valeurs possibles de $\langle w, c\rangle$, où c est de taille au plus $\alpha|w|^k$, et accepte w si et seulement si V a accepté au moins une telle paire. Puisque V est un vérificateur, il s'ensuit que M décide L.

Le nombre de mots de taille au plus αn^k est borné par $O(|\Sigma|^{\beta n^k})$ pour une certaine constante β (si vous ne le voyez pas, faites-le en exercice). Ceci est $O(2^{\log|\Sigma|\cdot\beta n^k})$. Puisque $|\Sigma|$ et β sont des constantes, il existe $k' \geq k$ tel que ceci est $O(2^{n^{k'}})$. La MT M' décide donc L en un temps borné par $O(n^d \cdot 2^{n^{k'}}) = O(2^{d\log n} \cdot 2^{n^{k'}})$, ce qui est $O(2^{n^{k''}})$, $k'' \in \mathbb{N}$. Ceci démontre que $L \in$ EXPTIME.

Exercice 6

Dans le problème de la couverture minimum, on reçoit des ensembles, et on veut choisir un minimum de ces ensembles pour couvrir tous les éléments. Plus formellement, soit $\mathcal{S} = \{S_1, \ldots, S_n\}$ une collection d'ensembles, et soit U un ensemble qu'on appelle *l'univers*. Le langage correspondant au problème de la couverture minimum est le suivant :

$$\text{SET-COVER} = \{\langle \mathcal{S}, U, k\rangle : \text{il existe } \mathcal{S}' \subseteq \mathcal{S} \text{ tel que } |\mathcal{S}'| \leq k \text{ et } \textstyle\bigcup_{S\in\mathcal{S}'} S = U\}$$

Montrez que SET-COVER est NP-complet.

Solution.

Il est clair que SET-COVER est dans NP, car une sous-collection $\mathcal{S}' \subseteq \mathcal{S}$ peut servir de certificat : il est facile de vérifier que $|\mathcal{S}'| \leq k$ et que $\mathcal{S}'$ couvre U.

On montre que SET-COVER est NP-difficile, par réduction via CNF-SAT (CNF-SAT est l'ensemble des formules satisfaisables sous la forme CNF, i.e. un ET de plusieurs clauses). On pourrait utiliser 3-SAT, mais en fait le 3 n'est pas vraiment utile ici.

Soit ϕ une formule CNF-SAT, et soient $x_1, \ldots, x_n$ les variables utilisées par ϕ. De plus, soient $C_1, \ldots, C_m$ les clauses de ϕ.

On créé une instance correspondante $\langle \mathcal{S}, U, k\rangle$ de SET-COVER. Les éléments de l'univers sont $U = \{C'_1, \ldots, C'_m, x'_1, \ldots, x'_n\}$, chaque élément C'_i correspondant à une clause et chaque élément x'_i correspondant à une variable. Les ensembles sont $\mathcal{S} = \{S_1, \overline{S}_1, \ldots, S_n, \overline{S}_n\}$, où S_i correspond à x_i et $\overline{S}_i$ à $\overline{x}_i$. Pour chaque i, $1 \leq i \leq n$, le contenu de S_i et $\overline{S}_i$ sont

$$S_i = \{x'_i\} \cup \{C'_j : C_j \text{ est satisfaite par } x_i\}$$
$$\overline{S}_i = \{x'_i\} \cup \{C'_j : C_j \text{ est satisfaite par } \overline{x}_i\}$$

On pose $k = n$.

On montre maintenant que ϕ est satisfaisable si et seulement si U peut être couvert avec au plus n ensembles de $\mathcal{S}$.

($\Rightarrow$) : supposons qu'il existe une affectation A de $x_1, \ldots, x_n$ qui satisfait ϕ. On construit $\mathcal{S}' \subseteq \mathcal{S}$ de la façon suivante. Pour chaque i, où $1 \leq i \leq n$, on ajoute S_i à $\mathcal{S}'$ si x_i est positif dans A, et on ajoute plutôt $\overline{S}_i$ à $\mathcal{S}'$ si x_i est négatif dans A. Il est clair que $|\mathcal{S}| \leq n$. De plus, $\mathcal{S}$ couvre tous les éléments $\{x'_1, \ldots, x'_n\}$ car on ajouté ajouté un ensemble S_i ou $\overline{S}_i$ pour chaque variable x_i. Chaque élément-clause C'_i est aussi couvert : puisque A satisfait ϕ, une variable x_j est affectée de façon à satisfaire la clause C_i, ce qui implique que S_j ou $\overline{S}_j$, selon le cas, couvre C_i'. L'existence de $\mathcal{S}'$ montre que $\langle S, U, n\rangle$ est dans SET-COVER.

($\Leftarrow$) : supposons qu'il existe $\mathcal{S}' \subseteq \mathcal{S}$ de taille au plus n qui couvre U. On remarque que pour couvrir $x_1', \ldots, x_n'$, il faut ajouter au moins un de S_i ou $\overline{S}_i$ pour chaque $1 \leq i \leq n$. Puisque $|\mathcal{S}'| \leq n$, ceci veut dire qu'on a en fait dans $\mathcal{S}'$ exactement un de S_i ou $\overline{S}_i$, mais pas les deux (si on avait les deux, on aurait strictement plus de n ensembles dans $\mathcal{S}'$ car il faut quand même choisir un de S_j or $\overline{S}_j$ pour chaque $x_j \neq x_i$).

Pour construire une assignation qui satisfait ϕ, on affecte x_i à vrai si $S_i \in \mathcal{S}$, et x_i à faux si $\overline{S}_i \in \mathcal{S}$, pour chaque $1 \leq i \leq n$. Notons que x_i n'est pas à la fois vrai et faux. De plus, chaque clause C_j est satisfaite par le x_i tel que S_i ou $\overline{S}_i$ couvre C_j', selon le cas. Donc, ϕ est satisfaisable.

Exercice 7

Dans un graphe, un sous-ensemble de sommets $X \subseteq V(G)$ est *dominant* si chaque sommet de $V(G) \setminus X$ a au moins un voisin dans X. Considérez le langage

$$\begin{aligned} \text{DOM-SET} = \{\langle G, k\rangle :\ & G \text{ est un graphe dans lequel il existe} \\ & \text{un ensemble dominant } X \subseteq V(G) \text{ de taille } k \text{ ou moins}\} \end{aligned}$$

Montrez que DOM-SET est NP-complet.

Solution.

Cette réduction est très similaire à celle de SET-COVER, donc nous donnons une version abrégée et plus informelle.

On réduit de CNF-SAT. Sur instance ϕ, on créé un graphe G dans lequel il y a les sommets $s_1, \overline{s}_1, \ldots, s_n, \overline{s}_n$. Pour chaque x_i, $1 \leq i \leq n$, on ajoute une arête entre s_i et $\overline{s}_i$. On ajoute ensuite les sommets $c_1, \ldots, c_m$, et une arête entre s_i et c_j si x_i satisfait C_j, ou entre $\overline{s}_i$ et c_j si $\overline{x}_i$ satisfait C_j. De plus, on ajoute les sommets $x_1', \ldots, x_n'$ tels que x_i' est voisin de s_i et $\overline{s}_i$. On cherche à savoir si G a un ensemble dominant de taille au plus n.

Si ϕ est satisfaisable par une assignation A, on les éléments correspondants dans $s_1, \overline{s}_1, \ldots, s_n, \overline{s}_n$ pour dominer G. C'est-à-dire, pour construire un ensemble dominant X, si $x_i = T$, on choisit s_i dans X et si $x_i = F$, on choisit $\overline{s}_i$ dans X. Chacun de s_i est soit dans X, ou dominé par $\overline{s}_i$. IL en est de même pour chaque $\overline{s}_i$. Chaque x_i' est dominé par s_i ou $\overline{s}_i$. Finalement, chaque sommet c_j est dominé par le s_i ou $\overline{s}_i$ qui correspont au choix de valeur de x_i de A qui a permis de satisfaire C_j. On a donc un ensemble dominant de taille n.

Si G a un ensemble dominant X de taille n ou moins, il doit contenir un sommet entre s_i, $\overline{s}_i$ ou x_i' pour dominer x_i'. Si X contient x_i, on peut voir que $(X \setminus \{x_i'\}) \cup \{s_i\}$ est aussi un ensemble dominant, car chaque voisin de x_i' est aussi un voisin de s_i (à part s_i). On peut donc supposer que X contient au moins 1 de s_i ou $\overline{s}_i$ pour tout i. Puisque $|X| \leq n$, X doit en fait contenir exactement un de s_i ou $\overline{s}_i$. Pour satisfaire ϕ, on affecte $x_i = T$ si $s_i \in X$ et $\overline{x}_i = F$ si

$\overline{s}_i \in X$. Puisque les sommets de X dominent tous les sommets c_j, l'assignation correspondante satisfait ϕ.

Exercice 8

Une collection d'ensembles $\{S_1, \ldots, S_\ell\}$ est dite *disjointe* si, pour tout i, j avec $1 \le i < j \le \ell$, on a $S_i \cap S_j = \emptyset$. Dans le problème SET-PACKING, on reçoit une collection d'ensembles $\mathcal{S}$ et on doit choisir une sous-collection disjointe de taille maximum. En terme de langage, on a :

SET-PACKING $= \{\langle \mathcal{S}, k\rangle$:il existe une sous-collection $\mathcal{S}' \subseteq \mathcal{S}$ disjointe telle que $|\mathcal{S}'| \ge k\}$

Montrez que SET-PACKING est NP-complet.

Solution.

SET-PACKING est dans NP car une sous-collection $\mathcal{S}$ peut servir de certificat : on peut vérifier en temps polynomial que $|\mathcal{S}| \ge k$ et que chaque paire d'ensembles de $\mathcal{S}$ ont une interesection vide.

On montre que SET-PACKING est NP-difficile par réduction via IND-SET. Soit $\langle G, k\rangle$ une instance de IND-SET. Notre collection $\mathcal{S}$ sera sur l'univers $U = E(G)$, i.e. on doit couvrir un élément correspondant à chaque arête. Pour chaque sommet $u \in V(G)$, on ajoute à $\mathcal{S}$ un ensemble

$$S_u = \{uv : v \text{ est un voisin de } u \text{ dans } G\}$$

i.e. S_u permet de couvrir chaque arête touchant à u. On montre que G a un ensemble indépendant de taille k si et seulement si il existe une sous-collection disjointe $\mathcal{S}' \subseteq \mathcal{S}$ telle que $|\mathcal{S}'| \ge k$ (donc ici, le k est le même pour les deux problèmes).

($\Rightarrow$) soit I un ensemble indépendant de G, avec $|I| \ge k$. Soit $\mathcal{S}' = \{S_u : u \in I\}$. Il est évident que $|\mathcal{S}'| \ge k$. De plus, puisque pour toute paire $u, v \in I$, il n'y a pas d'arête entre u et v, les arêtes incidentes à u sont disjointes des arêtes incidentes à v. Donc, $S_u \cap S_v = \emptyset$. On conclut que $\mathcal{S}'$ est une sous-collection disjointe de taille au moins k.

($\Leftarrow$) soit $\mathcal{S}'$ une sous-collection disjointe de $\mathcal{S}$ de taille au moins k. Soit $I = \{u : S_u \in \mathcal{S}'\}$. Bien sûr, $|I| \ge k$. Par construction de $\mathcal{S}$, si $S_u, S_v \in \mathcal{S}'$, alors u et v ne partagent pas d'arête. Puisque ceci est vrai pour tout $u, v \in I$, il s'ensuit que I est un ensemble indépendant.

Exercice 9

Soit G un graphe. Un *arbre couvrant* de G est un sous-graphe G' de G tel que G' est un arbre connectant tous les sommets de G. Dans le problème MINDEG-AC, on veut savoir s'il existe un arbre couvrant dont le degré maximum ne dépasse pas k :

$$\text{MINDEG-AC} = \{\langle G, k\rangle : G \text{ est un graphe dans lequel il existe un arbre couvrant tel que chaque sommet a un maximum de } k \text{ voisins}\}$$

Montrez que MINDEG-AC est NP-complet.
Suggestion : considérez un k très petit.

Solution.

Je ne donne que le sketch d'une preuve. MINDEG-AC est dans NP car un arbre couvrant peut servir de certificat.

Pour la NP-difficulté, on peut réduire de HAMPATH. Dans le problème HAMPATH, on veut savoir si G contient un chemin qui contient chaque sommet exactement une fois, ce qui est NP-complet. La réduction conserve le même graphe, mais pose $k = 2$. Il suffit ensuite d'observer qu'un arbre couvrant est un chemin Hamiltonien si et seulement si son degrée maximum est 2.

Exercice 10

Dans le problème 2-SAT, on reçoit une formule ϕ en forme CNF dans laquelle chaque clause a deux variables. On veut savoir si ϕ est satisfaisable. Montrez que 2-SAT est dans P.

Solution

J'ai écrit assez de solutions à ce point-ci. Je redirige vers : `http://philippe.gambette.free.fr/SCOL/Graphes/Gambette%20-%20Un%20graphe%20pour%20resoudre%202SAT.pdf`.

Exercice 11

Dans le problème MAX-2-SAT, on reçoits un ensemble de clauses CNF avec chacune deux variables, et on veut en satisfaire un maximum. En terme de langage, on a

$$\text{MAX-2-SAT} = \{\langle C, k\rangle : C \text{ est un ensemble de clauses CNF, et il existe une assigntion qui en satisfait au moins } k\}$$

Montrez que MAX-2SAT est NP-complet.
Suggestion : ce n'est pas trivial ! Une réduction à partir de MAX-2-XOR est possible.

Solution

Ce problème était volontairement plus difficile. Il fallait creuser un peu. En cherchant, on peut voir que le problème MAX-CUT est NP-complet. Voici une preuve : `http://www.cs.cornell.edu/courses/cs4820/2014sp/notes/reduction-maxcut.pdf`.

Ceci implique que le problème MAX-2-XOR est NP-complet. Dans le problème MAX-2-XOR, on reçoit un ensemble de clauses XOR avec 2 variables, de la forme $(x_i \oplus x_j)$, et on veut décider si on peut satisfaire au moins k clauses (notez que x_i ou x_j peuvent apparaître sous la forme négative). MAX-CUT se réduit trivialement à MAX-2-XOR, alors personne n'a écrit la preuve en détail. Vous trouverez l'idée à `https://cs.stackexchange.com/questions/100233/np-completeness-and-reduction-of-max-xor-sat-and-max-2-xor-sat`.

Ensuite, sachant que MAX-2-XOR est NP-complet, on le réduit à MAX-2-SAT. Considérez une instance de MAX-2-XOR ϕ sur variables $x_1, \ldots, x_n$ et clauses $C_1, \ldots, C_m$, où on veut décider s'il est possible de satisfaire k clauses. Ici, chaque C_i a la forme $(x_a \oplus x_b)$, ou $(\overline{x}_a \oplus x_b)$, ou $(\overline{x}_a \oplus \overline{x}_b)$ (un cas symétrique a été omis).

Pour construire une instance ϕ' de MAX-2-SAT, on transforme chaque $C_i = (a \oplus b)$ en deux clauses $A_i = (a \vee b)$ et $B_i = (\overline{a} \vee \overline{b})$. Ici, on utilise a et une variable x_i ou sa négation $\overline{x}_i$ (même chose pour b). Donc si ϕ a m clauses, ϕ' en a $2m$. On veut décider s'il est possible de satisfaire $m + k$ clauses de ϕ'.

Supposons qu'une assignation A peut satisfaire k clauses de ϕ. On garde la même assignation pour ϕ'. Pour chaque $C_i = (a \oplus b)$ satisfaite, a et b prennent une valeur différente. Donc, l'assignation satisfait à la fois $(a \vee b)$ et $(\overline{a} \vee \overline{b})$. Pour chaque clause non-satisfaite $C_i = (a \oplus b)$, a et b ont la même valeur. Donc on satisfait un seul de $(a \vee b)$ et $(\overline{a} \vee \overline{b})$. On satisfait donc $2k + (m - k) = m + k$ clauses de ϕ'.

Dans l'autre sens, supposons qu'on peut satisfaire $m + k$ clauses de ϕ'. Forcément, il y a k paires de A_i et B_i telles que $A_i = (a \vee b)$ et $B_i = (\overline{a} \vee \overline{b})$ sont toutes les deux satisfaites. Ceci implique que a et b ont une valeur différente, et donc que $(a \oplus b)$ est satisfaite. En gardant la même assignation, on satisfait donc k clauses XOR de ϕ.