Université de Bordeaux
M1 Informatique

Complexité et Calculabilité : TD2

NP et réductions

1 Rappels

Vous avez vu en cours deux notions fondamentales de la théorie de la complexité, les *réductions polynomiales* et la *classe NP*.

1. Une réduction polynomiale du problème A au problème B est un algorithme polynomial qui transforme les instances de A en instances de B, de telle manière que
   — les instances positives de A sont transformées en instances positives de B, et
   — les instances négatives de A sont transformées en instances négatives de B.
2. NP est une classe de problèmes. Un problème A est dans NP s'il existe un algorithme polynomial V – appelé *vérificateur* – et un polynôme p tels que :
   — pour toute instance positive x de A, il existe une preuve (ou certificat) y de taille au plus $p(|x|)$, montrant que l'instance x est positive, et
   — V accepte une paire (x, y) ssi y est une preuve que x est une instance positive.
3. Un problème A est NP-complet si (1) A est dans NP et (2) tout problème de NP se réduit à A (par réduction polynomiale).

Les réductions polynomiales peuvent s'utiliser de 2 façons : supposons que A se réduit à B par une réduction polynomiale, alors :

1. Si B est résoluble par un algorithme de temps polynomial, alors il en est de même pour A.
2. Si B est dans NP, alors il en est de même pour A.

Exemples d'utilisation de réduction :

1. SAT se réduit à 3-COL : on voit dans le cours que SAT est NP-difficile, donc la réduction ci-dessus nous dit que 3-COL est NP-difficile.
2. 3-COL se réduit à SAT : comme SAT est dans NP, on déduit que 3-COL est dans NP. En fait, on utilise cette réduction pour donner un algorithme "convenable" pour 3-COL : l'algorithme basé sur les SAT-solveurs est exponentiel au pire des cas, mais sera souvent plus performant qu'un algorithme naif.

2 Réductions

Exercice 2.1
On considère 3 problèmes A, B et C et on suppose qu'il existe une réduction polynomiale π_1 de A à B en $O(p_1(n))$ et une réduction polynomiale π_2 de B à C en $O(p_2(n))$, où p_1 et p_2 sont des polynômes.

1. Montrer que $\pi_2 \circ \pi_1$ (partant d'une instance x de A, on lui applique π_1, et on applique π_2 au résultat de π_1) est une réduction de A à C. Quelle est sa complexité ? (**Indication :** remarquer que la *taille* de $\pi_1(x)$ est bornée par le *temps de calcul* de π_1 sur x) Si par exemple $p_1(n) = n^3$ et $p_2(n) = n^4$, que vaut cette complexité ?
2. Peut-il exister une réduction polynomiale de B à A ? Est-ce toujours le cas ?
3. Existe-t'il toujours une réduction polynomiale de A à A ?
4. Que vient-on de démontrer sur la relation $R(A, B) :=$ "il existe une réduction polynomiale de A à B" ?

3 Algorithme naïf

Lorsqu'un problème est dans NP, on a l'assurance que s'il existe une solution, il y en a une qui est de taille polynomiale et qu'il est possible de vérifier en temps polynomial qu'une donnée est une solution ou non. L'exercice suivant a pour but de vous faire toucher ce fait et de vous amener à réfléchir à la complexité d'un algorithme naïf l'utilisant.

Exercice 2.2
Un *arbre couvrant* d'un graphe G est un arbre constitué uniquement d'arêtes de G et contenant tous les sommets de G [1]. La profondeur $d(v)$ d'un sommet dans l'arbre est la distance par rapport à la racine, c'est-à-dire le nombre d'arêtes sur le chemin direct de la racine à v. La hauteur d'un arbre est la profondeur maximale d'un de ses sommets [2] $max_v(d(v))$. Formellement, le problème HAC est le suivant :

Entrée : Un graphe non-orienté G et un entier k.

Sortie : Si il existe un arbre couvrant de G de hauteur exactement k.

Ce problème est NP-complet (il n'est pas demandé ici de le montrer). L'objet de cet exercice est de vous amener à trouver un algorithme naïf le résolvant et de déterminer sa complexité.

1. On considère une instance du problème G, k. Étant donné un ensemble d'arêtes $E' \subseteq E$, à quelle condition E' est-il un arbre couvrant de G ? Étant donné un sommet x, décrivez un algorithme déterminant si E' décrit un arbre couvrant de racine x et de profondeur k.
2. Quelle est la complexité de l'algorithme précédent ? Vous justifierez en indiquant combien d'opérations vous effectuez et de combien de mémoire vous avez besoin en fonction du nombre de sommets et d'arêtes de votre graphe.
3. Combien de sous-ensembles de E' devez-vous tester avec votre algorithme si vous voulez déterminer si G dispose d'un arbre couvrant de profondeur k ?

1. En particulier, tout graphe admettant un arbre couvrant doit être connexe.
2. Remarquez qu'il suffit de regarder les feuilles.

4. Déduisez-en la complexité d'un algorithme énumérant tous les E' pertinents et testant pour chacun d'entre eux s'il décrit un arbre couvrant ou non.

Exercice 2.3
On considère maintenant le problème Couverture par sommets (Vertex Cover). Soit $G = (V, E)$ un graphe, un sous-ensemble de sommets $C \subseteq V$ est dit *couvrant* si pour toute arrête $(u, v) \in E$ au moins une de ses extrémités u ou v appartient à C. Le problème Couverture par sommets est défini ainsi :

Entrée : Un graphe non-orienté $G = (V, E)$ et un entier p.

Sortie : Existe-t-il un sous-ensemble couvrant $C \subseteq V$ tel que $|C| = p$.

1. Justifiez que Couverture par sommets est dans NP en donnant un algorithme le résolvant.
2. Donnez une réduction polynomiale de Couverture par sommets vers SAT.
3. Que peut-on dire d'un problème qui se réduit polynomialement vers SAT ?

4 Réductions depuis 3-SAT

Le problème 3-SAT est un cas particulier de SAT, dans lequel toutes les clauses de la formule que l'on cherche à vérifier contiennent exactement 3 littéraux. Il est NP-complet, et est très utilisé pour montrer qu'un problème est NP-difficile, car il est plus commode à manipuler que SAT dans sa version générale.

Exercice 2.4
On considère le problème Programmation entière qui teste l'existence d'une solution à un système d'égalités et d'inégalités linéaires. Si x, y, z sont des noms de variables voici par exemple un tel système :

$$\begin{aligned} x + 2 \times y &= z \\ z - x &\leq 3 \times y \end{aligned}$$

Une solution à ce système est par exemple $x = 1$, $y = 1$ et $z = 3$. Un système est dit *gardé* s'il contient une inégalité de la forme $0 \leq x \leq a$ (avec a un entier) pour tout variable x existante dans le système. On considère le problème Programmation entière suivant :

Entrée : Un système $\mathcal{S}$ gardé d'égalités et inégalités linéaires.

Sortie : Existe-t-il une solution (en nombres entiers) au système ?

Le but de l'exercice est de montrer que Programmation entière est NP-complet.

1. À votre avis, pourquoi restreint-on le problème aux systèmes qui sont gardés ?
2. Justifiez que Programmation entière est dans NP.
3. On désire maintenant montrer que Programmation entière est NP-difficile. Pour celà on construit une réduction polynomiale de 3-SAT à Programmation entière. À toute clause de 3-SAT on va associer une égalité linéaire. On commence par un exemple.

(a) Soit $C = p_1 \vee p_2 \vee \neg p_3$ une clause de 3-SAT. Donnez une expression $E(x_1, x_2, x_3)$ telle que $p_1 \vee p_2 \vee \neg p_3$ s'évalue à vrai ssi $E(p_1, p_2, p_3)$ est différent de 0 (où $E(p_1, p_2, p_3)$ s'obtient en substituant x_i par 1 si p_i est vrai et par 0 sinon).

(b) Généraliser l'exemple précédent et donner une réduction polynomiale de 3-SAT à Programmation entière. En conclure que Programmation entière est une problème NP-difficile.

4. Conclure que Programmation entière est une problème NP-complet.

5 Réductions de problèmes de graphes

Exercice 2.5

On étudie maintenant le problème Clique :

Entrée : Un graphe non-orienté G et un entier q.

Sortie : Existe-t-il un sous-graphe *complet* de G ayant q sommets (une *q-clique*) ?

1. Montrer que Couverture par sommets et Clique se réduisent mutuellement l'un à l'autre.
2. En déduire que que Clique est NP-complet.

Indication : Considérer le graphe H *complémentaire* à G : l'ensemble des sommets est le même, $V(G) = V(H)$, et pour deux sommets $u, v \in V(G)$ une arête (u, v) appartient à $E(H)$ si et seulement si elle n'appartient pas à $E(G)$.

6 Problèmes de circuits hamiltoniens

Exercice 2.6

On définit deux versions du problème de l'existence d'un cycle ou circuit hamiltonien, selon que les graphes considérés sont orientés ou non :
CircuitHamiltonienOrienté :

Entrée : Un graphe orienté $G = (V, A)$

Sortie : Existe-t-il un circuit hamiltonien dans G *i.e.* une suite $(s_0, \ldots, s_n)$ de sommets de G telle que
— $s_0 = s_n$
— chaque sommet de G apparaît une fois et une seule dans $\{s_1, \ldots, s_n\}$ (en particulier $n = |V|$)
— pour chaque entier $1 \leq i \leq n$, (s_{i-1}, s_i) est un des arcs de G.

CircuitHamiltonien :

Entrée : Un graphe non orienté $G = (V, E)$

Sortie : Existe-t-il un cycle hamiltonien dans G, *i.e.* une suite $(s_0, \ldots, s_n)$ de sommets de G telle que
— $s_0 = s_n$

— chaque sommet de G apparaît une fois et une seule dans $\{s_1, \ldots, s_n\}$ (en particulier $n = |V|$)
— pour chaque entier $1 \leq i \leq n$, $\{s_{i-1}, s_i\}$ est une des arêtes de G.

1. Décrire une réduction polynomiale de CircuitHamiltonien à CircuitHamiltonienOrienté. Si le graphe de départ a n sommets et m arêtes, combien le graphe d'arrivée aura-t-il de sommets et d'arcs ?
2. On définit la transformation R suivante : $G = (V, A)$ étant un graphe orienté, $G' = R(G)$ est un graphe non orienté défini par $G' = (V', E)$, où
   — $V' = \{(u, i) : u \in V, i \in \{0, 1, 2\}\}$;
   — $E = \{\{(u, 0), (u, 1)\} : u \in V\} \cup \{\{(u, 1), (u, 2)\} : u \in V\} \cup \{\{(u, 2), (v, 0)\} : (u, v) \in A\}$

   Quels sont, en fonction des nombres de sommets et d'arêtes de G, les nombres de sommets et d'arêtes de G' ? Montrer que R est bien une réduction polynomiale de CircuitHamiltonienOrienté à CircuitHamiltonien. (**Question subsidiaire :** On utilise trois copies de chaque sommet, est-ce que deux suffiraient ?)
3. On a montré que les deux problèmes hamiltoniens précédents se réduisent l'un à l'autre. Que reste-t-il à faire pour montrer que les deux problèmes sont des problèmes de NP ? Pour montrer que les deux problèmes sont NP-difficiles ?
4. Montrer que ces deux problèmes hamiltoniens sont NP.