NOM : AUFORT Prénom : William Jury :

~~Algèbre~~ ← Entourez l'épreuve → ~~Analyse~~ Informatique

Sujet choisi : 928 : Problèmes NP-complets : Exemples de réduction.

~~Autre sujet~~ : Références : Sipser, Garey Johson, Cormen

[Sipser]

I] Temps polynômial et NP Complétude

a) Pourquoi P et NP ?

Déf 1 : Le temps d'execution d'une machine de Turing déterministe est une fonction $f : \mathbb{N} \to \mathbb{N}$ où $f(n)$ est le plus grand nombre d'étapes de la machine sur une entrée de taille n.
L'ensemble des langages dont le temps d'execution par une telle machine est $O(T(n))$ est noté $TIME(T(n))$.

Déf 2 : $P = \bigcup_{k \in \mathbb{N}} TIME(n^k)$

Remarque 3 : P est central en théorie de la complexité car tous les modèles de calculs raisonnables sont polynômialement équivalents : on peut alors s'abstraire de ce modèle quand on raisonne sur un problème.

Ex 4 : Les machines de Turing déterministe à 1 bande, à k-bandes, et les machines RAM sont polynômialement équivalentes.

Prop 5 : Si $T(n) \geq n$, alors une machine de Turing NON-déterministe qui s'execute en temps $T(n)$ admet une machine de Turing équivalente déterministe qui s'execute en temps $2^{O(T(n))}$

Remarque 6 : Cette différence exponentielle souligne davantage l'intérêt de P.

Déf 7 On définit de même $NTIME(T(n))$ pour les machines non déterministes et $NP = \bigcup_{k \in \mathbb{N}} NTIME(n^k)$.

b) Vérificateurs et exemples de problèmes de NP

Déf 8 Un vérificateur d'un langage L est un algorithme V tel que $A = \{w \mid V \text{ accepte } \langle w, c\rangle \text{ pour un certain } c\}$. C est une chaîne de caractères appelé certificat.

Thm 9 NP est la classe des langages admettant un vérificateur qui s'execute en temps polynômial

Remarque 10. C'est la caractérisation utilisée en pratique pour montrer qu'un langage est dans NP

Remarque 11 : On étudie les problèmes de décision c'est-à-dire dont la réponse pour une entrée est "oui" ou "non". L'ensemble des entrées valides représente donc bien un langage.

Exemple 12. SAT (l'ensemble des codes de formules satisfiables) est dans NP.

Exemple 13 : $CLIQUE = \{\langle G, k\rangle,$ G est un graphe non orienté contenant une k-clique$\}$ est dans NP.

c) Réduction et NP-Complétude

Idée 14 : On cherche les problèmes "les plus difficiles" de NP : on va donc les comparer.

Déf 15. Soient A et B deux langages. On dit que A est réductible en temps polynômial en B, noté $A \leq_p B$, s'il existe une fonction f constructible en temps polynômial telle que $w \in A \iff f(w) \in B$.

Remarque 16 On voit que si on sait résoudre B, par f on sait résoudre A.

Remarque 17 : Cette notion est "invariante" par transformation polynômiale, ce qui montre encore l'étude de P (et de NP).

Prop 18 : Si $A \leq_p B$ et $B \in P$, alors $A \in P$.

Déf 19 : Un langage B est NP-Complet si : $B \in NP$ et $\forall A \in NP$, $A \leq_p B$.

Remarque 20 : Avec la remarque 16, un problème NP-Complet est plus difficile que tout problème de NP.

Exemple fondamental : Théorème de Cook-Levin (21) :
SAT est NP-Complet.

Remarque 22 : La preuve du théorème nécessite de montrer que $\forall A \in NP$, $A \leq_p SAT$, ce qui est fastidieux. A partir de maintenant, pour prouver qu'un problème est NP-Complet, on utilise encore la réduction polynômiale :

Théorème 23 : Si A est NP-Complet et $A \leq_p B$ avec $B \in NP$, alors B est NP-Complet.

II) Exemples de problèmes NP-Complets et de réductions

Remarque 24 : Avec le théorème 23 et Cook-Levin, on peut obtenir une collection de problèmes NP-Complet sous forme d'"arbre de réduction" (voir annexe).

a) Méthode de réduction : premiers exemples

Méthode 25 : Pour prouver que A est NP-Complet, la méthode est donc la suivante :
1) Montrer que $A \in NP$.
2) Choisir un problème B dont on sait qu'il est NP-Complet.
3) Trouver la transformation f dans $B \leq_p A$,
(4) Montrer qu'elle est polynômiale).

Exemple 26 : 3SAT est l'ensemble des formules satisfiables de la forme $\bigwedge_{i=1}^{n} C_i$, avec C_i de la forme $\bigvee_{j=1}^{3} a_{ij}^{*}$ $\forall i$.

Prop 27 : 3SAT est NP-Complet par réduction à partir de SAT.

Exemple 28 : VERTEX-COVER = $\{\langle G=(V,E), k\rangle$, $\exists V' \subseteq V$ tel que $|V'| \leq k$ et $\forall (u,v) \in E$, $u \in V'$ ou $v \in V'\}$ est NP-Complet, par réduction à partir de 3-SAT.

Exemple 29 : 3D-MATCHING = $\{M \subseteq X \times Y \times Z$, X, Y, Z ensembles disjoints de cardinal n, $\exists M'$, $M' \subseteq M$, $|M'| = n$ et toute paire d'éléments de M' a des coordonnées 2 à 2 disjointes$\}$
3D-MATCHING est NP-Complet par réduction à partir de 3SAT. (difficile)

b) Méthode du "remplacement local"

Idée 30 : Pour construire f, on peut à partir de "sous-unités" d'une instance de B, construire une "sous-unité" d'une instance de A, puis les assembler.

Exemple 31 : Pour $3SAT \leq_p SAT$, on associe indépendamment à chaque clause C_i, une clause 3SAT C_i' dont la construction dépend uniquement de la taille de C_i.

Exemple 32 : On définit DEV-AGREG le problème : " Etant donné un ensemble de leçons, un ensemble de développements (chaque développement couvrant un certain nombre de leçons), et un entier k, est-il possible de couvrir toutes les leçons avec au plus k développements ? "

Prop 33 : DEV-AGREG est NP-Complet. (DEV1)

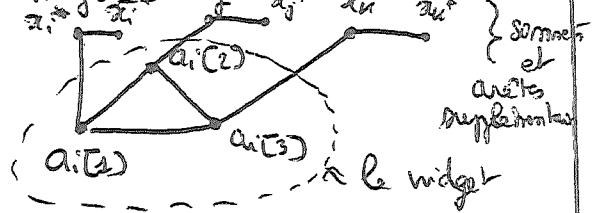
Exemple 34 : CLIQUE est NP-Complet, (SET-COVER) par réduction à partir de VERTEX-cover.

c) Construction de widgets

Idée 35 : On construit des composants, comme dans l'idée 30, mais plus compliqués, ayant des propriétés intéressantes.

Exemple 36 : HAM-CYCLE = $\{G$ contenant un circuit Hamiltonien$\}$ est NP-Complet, par réduction à partir de VERTEX-COVER. On utilise un widget détaillé en annexe. (Canon)

Exemple 37 : Pour $3SAT \leq_p$ VERTEX-COVER, on a pour chaque clause $C_i = x_i^* \vee x_j^* \vee x_k^*$ un widget :



d) Autres exemples [Carmen]

Remarque 38 : Beaucoup de domaines contiennent des problèmes NP-Complets

Exemple 39 : $SUBSET\text{-}SUM = \{\langle S, t\rangle, \exists S' \subseteq S, \sum_{s \in S'} s = t\}$ est NP-Complet, par réduction à partir de 3SAT

Remarque 40 On a donc vu des exemples en logique (SAT, 3SAT), arithmétique (SUBSET-SUM), graphes, ensembles...

III La NP-complétude "en pratique"

a) Restriction et problèmes similaires

Idée 41 : Dans des cas simples, des preuves de NP-complétude peuvent être faite par "restriction" ; si un problème contient, comme cas particulier, un problème NP-Complet.

Exemple 42 : $X3C = \{\langle X, C\rangle, X$ est un ensemble de cardinal $3n$, C est une collection d'éléments de $\mathcal{P}(X)$ de taille 3, $\exists C' \subseteq C$, $\forall x \in X$, x appartient dans exactement un élément de $C'\}$ est NP-Complet, par restriction de 3D-MATCHING.

Remarque 43 : On observe ici des problèmes similaires. Mais attention, on peut avoir des problèmes similaires de complexité différente !

Exemple 44 2SAT $\in$ P, mais 3SAT est NP-Complet.

Exemple 45 : Le plus court chemin entre 2 sommets d'un graphe est résoluble en temps polynomial, alors que le problème de décision associé au plus long chemin est NP-Complet.

Exemple 46 : VERTEX-COVER est NP-Complet, alors que EDGE-COVER (couverture par arêtes), que l'on peut voir comme un cas particulier de SET-COVER, peut être résolu en temps polynômial, via couplage maximal. (SET-COVER étant NP-Complet)

b) Algorithmes d'approximation [Carmen]

Idée 47 : Si le problème de décision associé à un problème d'optimisation est NP-Complet, le problème d'optimisation semble difficile à résoudre. On cherche alors à trouver une solution "proche" de l'optimale, en temps raisonnable, i.e. polynômial.

Déf 48 : Un algorithme associé à un problème d'optimisation est une $\rho(n)$-approx si, pour toute entrée de taille n, le coût C de la solution donnée par l'algorithme et C^* le coût optimal pour l'entrée vérifie $\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq \rho(n)$.

Exemple 49 L'algorithme glouton suivant pour VERTEX-COVER est une 2-approx :

```
C ← ∅   E' ← E.
tant que E' ≠ ∅ faire
    prendre (u,v) ∈ E' quelconque
    C ← C ∪ {u, v}
    supprimer de E' toute arête incidente à
    u ou v
retourner C.
```

Déf 50 : Le problème du voyageur de commerce est :

$TSP = \{\langle G, c, k\rangle, G$ graphe complet, $c : V \times V \to \mathbb{N}$, $k \in \mathbb{N}$, G admet un tour de coût au plus $k\}$

Prop 51 : TSP est NP-Complet, par réduction à partir de HAM-CYCLE.

Prop 52 : TSP reste NP-Complet si on suppose que c vérifie l'inégalité triangulaire.

Remarque 53 : C'est une hypothèse raisonnable, en pratique c'est le cas !

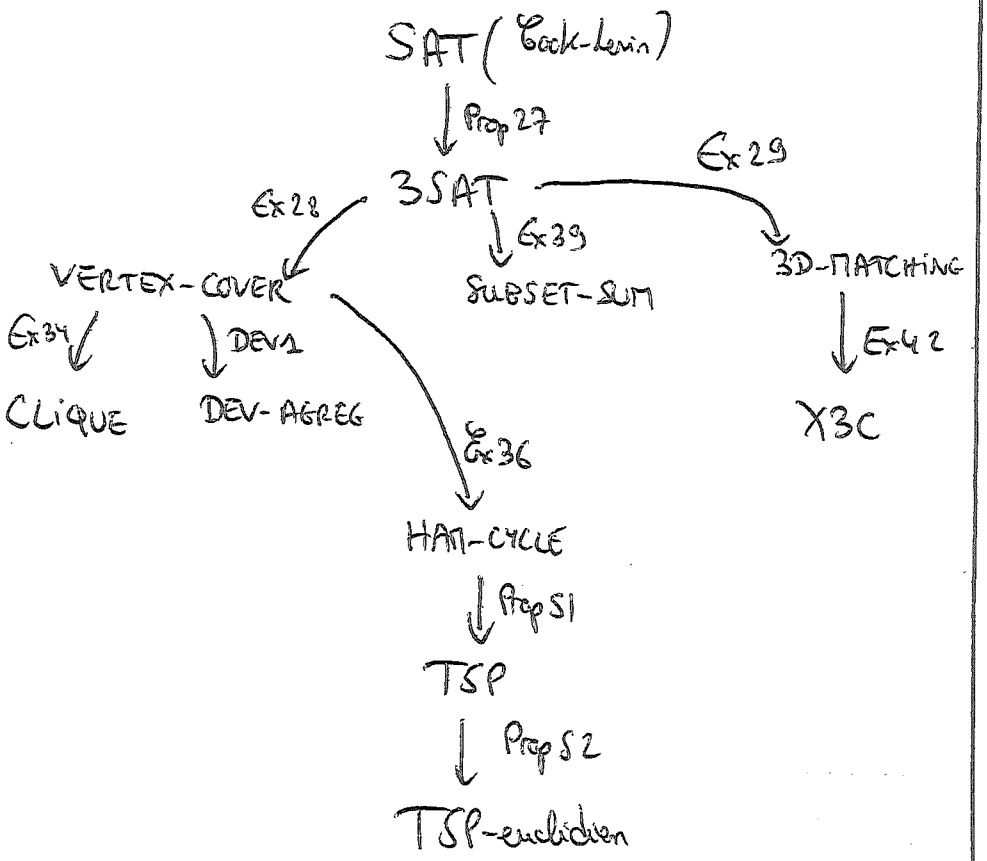
Exercice 54 : Ecrire un algorithme de 2-approx pour TSP-euclidien, et la preuve

c) Algorithmes pour des problèmes NP-Complets

Remarque 55 : Ce n'est pas parce qu'un problème est NP-Complet que des algorithmes "efficaces" en pratique n'existent pas, même s'ils ne sont pas polynomiaux.

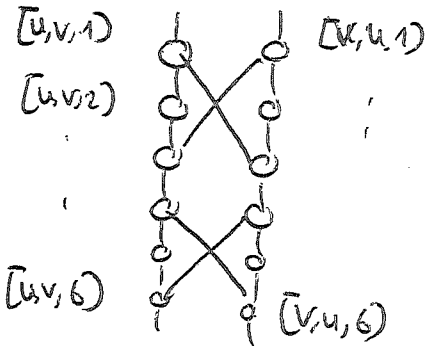
Exemple 56 Certains solveurs SAT, comme MINISAT, sont très performants en pratique pour résoudre des instances de SAT. On les utilise également pour d'autres algorithmes associés à des problèmes NP-complets, en modélisant le problème avec une formule. (par exemple des problèmes de routage, de coloriage...).

Annexe 1 : Arbre de réduction des problèmes NP-Complets rencontrés :

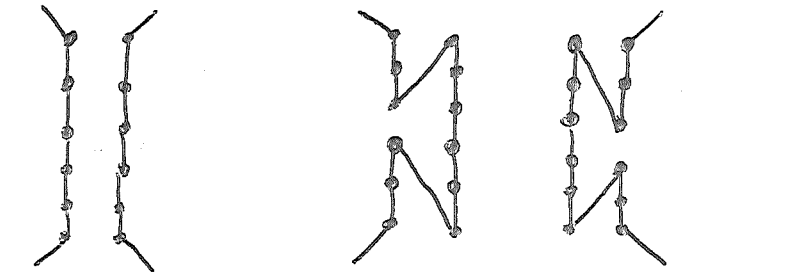


Annexe 2 : Widget pour Ham-CYCLE : [Cormen]

Pour $(u,v) \in E$, on associe le widget.



Les seules arêtes sortantes du widget sortent en $[u,v,1)$ $[u,v,6)$, $[v,u,1)$, $[v,u,6)$ de sorte que les seuls parcours possibles dans un cycle Hamiltonien sont :



on a pris u dans la couverture

on a pris v dans la couverture.

on a pris u et v dans la couverture