928 : Problèmes NP-Complets : Exemples de réduction.

But : Introduire une classe de problèmes de difficultés équivalentes auxquels aucun algorithme utilisable dans le cas général n'a été trouvé, mais dont on ne sait pas s'il existe. La classe NP comprend des problèmes présents dans la plupart des domaines scientifiques.

(I) Notion de NP-complétude

1) Un exemple [G-J]

Problème de voyageur de commerce [TSP1] :

→ Instance : Un ensemble fini $S = \{s_1, \dots, s_n\}$ et une fonction distance

$$d : S^2 \to \mathbb{N}^*$$

→ Problème : Trouver un cycle $\sigma = (s_{\sigma(1)}, \dots, s_{\sigma(n)})$ minimisant la longueur d'un tour : $\sum_{i=1}^{n-1} d(c_{\sigma(i)}, c_{\sigma(i+1)}) + d(c_{\sigma(n)}, c_{\sigma(1)})$

- Un algorithme consiste à calculer la longueur des $n!$ tours possibles et en prendre un minimal. La complexité d'un tel algorithme le rend inutilisable en pratique.
- Etant donné un tour, on peut vérifier en temps linéaire si son coût est inférieur à un entier k fixé en paramètre.

2) La classe NP [Cor] [F-B]

Def 1 : [Vérificateur / Classe NP] : - Un vérificateur en temps polynomial pour un langage L est une machine de Turing non déterministe V qui accepte des entrées de la forme $\langle w, c \rangle$ en temps polynomial en $|w|$ telle que $L = \{w \mid \exists c, \langle w, c \rangle \in L(V)\}$

- Un langage L est dans NP s'il admet un vérificateur en temps polynomial.

Ex 2 : - TSP2 : Etant donnés S, d et $k \in \mathbb{N}^*$, le problème de savoir si S admet un tour de longueur inférieure à k est NP. C'est le problème de décision associé au problème d'optimisation TSP1 étudié précédemment.

- L'ensemble des nombres premiers codés en binaire est dans Co-NP = $\{L \mid \overline{L} \in NP\}$

Rmq 3 : - Intuitivement, NP représente les problèmes où l'on sait vérifier par un algorithme polynomial si une solution potentielle est effectivement solution ou non.

- NP est l'ensemble des langages acceptés par machines de Turing non déterministes en temps polynomial.

Rmq 4 : $P \subset NP \subset$ Exptime et on ne sait pas si les inclusions sont strictes. (au moins une des deux l'est)

3) Réduction polynomiale [Cor] [K-T]

Def 5 [Réduction polynomiale] : Soient A et B deux problèmes codés par L_A et L_B sur Σ_A et Σ_B. Une réduction polynomiale de A à B est une fonction

$f : \Sigma_A^* \to \Sigma_B^*$ calculable en temps polynomial par machine de Turing telle que

$$(w \in L_A) \iff (f(w) \in L_B)$$

Dans ce cas, on note $A \leq_p B$.

Ex 6 : - Problème du cycle Hamiltonien : (Ham-cycle)

→ Instance : Un graphe orienté $G = (S, A)$

→ Question : Existe-t-il un cycle hamiltonien, i.e. un cycle visitant chaque sommet une unique fois (sauf le sommet de départ qui est aussi le sommet d'arrivée).

Réduction de Ham-Cycle à TSP2 :

Soit $G = (S, A)$ instance de Ham-Path, on définit l'instance (S', d, k) de TSP2 de la manière suivante : → S' est une copie de S

→ $d(s'_i, s'_j) = \begin{cases} 1 \text{ si } (s_i, s_j) \in A \\ 2 \text{ sinon} \end{cases}$

→ $k = n = |S|$

On obtient une correspondance directe entre un tour de longueur $\leq n$ dans TSP2 et un cycle hamiltonien dans Ham-Path. Cette réduction se construit en temps $\dots O(|S|^2)$.

De tout algorithme résolvant TSP2, on en déduit un algorithme résolvant Ham-Cycle.

Prop 6: Si $A \leq_p B$ et $B \in P$, alors $A \in P$.

4) Notion de NP-Complétude et exemples de références [Car]

Def 7: [NP-dur / NP-complet]:
- Un problème A est NP-dur si tout problème de NP se réduit à A en temps polynomial
- Un problème est NP-complet s'il est NP et NP-dur.

Thm 8: [Cook-Levin]: SAT: → instance: φ formule du calcul propositionnel sous forme normale conjonctive sur un ensemble fini de variables.

DVPT 1

→ question: existe-t-il une valuation satisfaisant φ?

Le problème SAT est NP-complet.

Prop 9: Si $A \leq_p B$ et A est NP-dur, alors B est NP-dur.

Rmq 10: Il n'est donc plus nécessaire de montrer que tout problème NP se réduit à A dans une preuve de NP-difficulté. Une réduction d'un problème NP-complet suffit.
- SAT constitue l'exemple de référence des problèmes NP-complets et fut le premier classifié comme NP-complet.

Problème 3-SAT: → instance: une formule φ du calcul propositionnel sous forme normale conjonctive sur un ensemble fini de variables où chaque clause possède au plus 3 littéraux.
→ existe-t-il une valuation satisfaisant φ?

3-SAT est NP-complet: - vérifier qu'une valuation satisfait φ se fait linéairement en évaluant les littéraux

- Réduction de SAT à 3-SAT: Soit φ instance de SAT. On cherche à remplacer les clauses de longueurs ≥ 4 par une conjonction de clauses de longueurs ≤ 3. Soit $c = a_1 \vee a_2 \vee \dots \vee a_k$ clause de longueur $k \geq 4$.

On introduit les nouvelles variables $y_1, \dots, y_{k-3}$ et la formule φ_c suivante:

$$\varphi_c = (a_1 \vee a_2 \vee y_1) \wedge (\overline{y_1} \vee a_3 \vee y_2) \cdots (\overline{y_{k-3}} \vee a_{k-1} \vee a_k)$$

Si a_i est évaluée à 1, alors en évaluant $\begin{cases} y_1, \dots, y_{i-2} \text{ à } 1 \\ y_{i+1}, \dots, y_{k-3} \text{ à } 0 \end{cases}$, alors φ_c est évaluée à vraie. D'où (c est satisfiable) $\Leftrightarrow$ (φ_c est satisfiable).

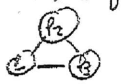
À chaque clause c de longueur ≥ 4, on introduit $(k-3)$ variables et $k-2$ nouvelles clauses. Si on note n le nombre de clauses de φ et p le nombre de variables, on construit une formule avec au plus $n(p-3)+p$ nouvelles variables et $n(p-2)$ nouvelles clauses. La réduction est bien polynomiale.

II) Exemples de réductions [Rap] [K-T]

Nous allons donner des exemples de réductions pour montrer le caractère NP-dur de problèmes NP classifiés par :
- Les dépendances des preuves (cf Annexe 1)
- Les catégories des problèmes (cf Annexe 2)

1) Réduction de 3-SAT à ENS_IND:

ENS_IND: → Graphe $G=(S,A)$ non orienté et un entier k
→ question: existe-t-il un ensemble I de cardinal k indépendant, c'est-à-dire: $\exists (a,b) \in I^2,\ (a,b) \notin A$?

Pour $\varphi = \bigwedge_{i=1}^{m} c_i$ instance de 3-SAT, on associe le graphe $G_\varphi = (S,A)$ défini par:
- Pour une clause $l_1 \vee l_2 \vee l_3$, on associe le gadget 
- Pour une clause $l_1 \vee l_2$, on associe 

• On pose $k = n$ le nombre de clauses de φ.

Par construction, au plus un sommet d'une clause peut appartenir à l'ensemble indépendant. Comme $k=n$, l'ensemble indépendant représente les littéraux évalués à 1. Pour ne pas évaluer x et $\bar{x}$ à 1, on construit un arc entre x et $\bar{x}$: (x)—$(\bar{x})$ pour toute variable x.

On obtient des réductions immédiates à VC couverture de sommets et Clique par les caractérisations : (S couverture de sommets) ssi (S\S' ensemble indépendant)

(S clique de G) ssi (S' ensemble indépendant de $\bar{G}=(S, \bar{S}\setminus A)$)

2) Réduction de 3-SAT à Ham-Cycle

Soit $\varphi = \bigwedge_{i=1}^{n} c_i$ instance de 3-SAT sur l'ensemble de variables $\{x_1, \dots, x_p\}$.

À chaque variable x on associe le gadget :

G_i :  de taille $2n+2$

de plus on crée de sommets pour chaque clause c_j et :

- si $x_i \in c_j$, on ajoute les arêtes (x_i^{2j}, c_j) et (c_j, x_i^{2j+1})
- si $\bar{x}_i \in c_j$, on ajoute les arêtes (x_i^{2j+1}, c_j) et (c_j, x_i^{2j})

Si x_i est évaluée à 1, on parcourt G_i de gauche à droite et de droite à gauche sinon, en passant dès que possible par les c_j.

3) Problème PSA

DUPT 2
- Instance : Deux ensembles finis de mots S et T et un entier $k \in \mathbb{N}$
- Question : Existe-t-il un automate fini à k états qui accepte tous les mots de S et rejette tous ceux de T ?

Le problème PSA est NP-complet.

4) Réduction de 3D-Matching à ZOE → Instance : $A \in M_{m,n}(\{0;1\})$

→ Question : existe-t-il un vecteur $x \in \{0;1\}^n$ tel que $Ax = 1$

Soit une instance de 3D-matching (X, Y, Z) de taille n chacun et m triplets $T \subseteq X \times Y \times Z$.

On pose $x_i = \begin{cases} 1 & \text{si le } i\text{ème triplet est choisi} \\ 0 & \text{sinon} \end{cases}$

Pour chaque élément x de X (ou Y, ou Z), supposons x apparaisse dans $x_{j_1}, \dots, x_{j_k}$, alors

$x_{j_1} + x_{j_2} + \dots + x_{j_k} = 1$ signifie que x se trouve dans exactement un triplet

III) Contourner la NP-complétude [KT] [Pap]

1) Se restreindre à un cas particulier

Si on renforce les hypothèses sur les instances, on peut se trouver face à des problèmes dans P.

Ex 11 :

NP-complet	Problème dans P
3-SAT	2SAT, HORNSAT
ENS-IND	ENS-IND sur les arbres

2) Recherche exhaustive

+ Branch & Bound

- Backtracking : faire un test d'échec à chaque étape
- séparation et numérisation : diviser le problème en sous-problèmes de plus en plus fins et éliminer les sous-problèmes qui ne vérifient pas une certaine borne inférieure

3) Par des heuristiques

On utilise des méthodes qui donne intuitivement une bonne solution, mais qui ne sont pas forcément optimales

Ex 12 : Problème TSP dans le plan euclidien : on effectue une fermeture bitonique

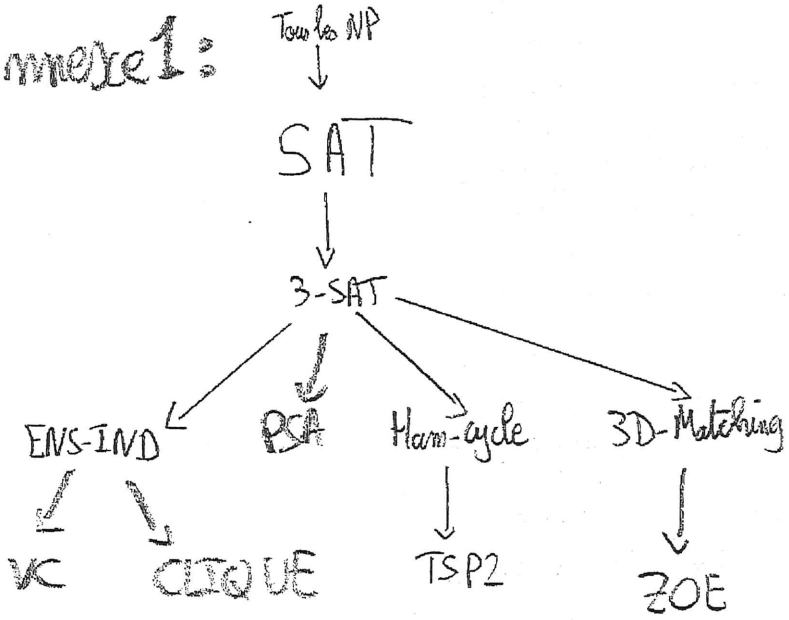
4) Par des algorithmes d'approximation

On s'intéresse à des problème d'optimisation dans cette partie

Prop 13 : Le problème Scheduling : → entrée : m machines et n travaux de temps t_i.
→ sortie : trouver un ordonnancement de tâche de temps minimale

peut s'approximer par un algorithme glouton polynomial tel que la solution T vérifie $T \leq \frac{3}{2} T^*$ où T^* solution optimale

Annexe 1 :



Annexe 2 :

1) Satisfaction de contraintes : SAT, 3-SAT
2) Couverture : VC, TSP, ENS-IND, Clique
3) Partitionnement : 3D-Matching
4) Séquencement : Ham-cycle, Scheduling
5) Problèmes numériques : ZOE, knapsack

Refs :

- [G-J] : Garey-Johnson
- [Car] : Carton
- [F-B] : Floyd-Bagel
- [K-T] : Kleinberg-Tardos
- [Pap] : Papadimitriou : Algorithms