928 : Problèmes NP-complets - Exemples de réductions

Introduction. Comment mesurer la difficulté d'un problème ?

- Problème du voyageur de commerce :

PVC : Entrée : Un graphe complet pondéré
Sortie : Un cycle hamiltonien de poids minimal

I. NP-complétude [SIP]

1) Problèmes de décision.

On s'intéresse ici aux problèmes de décision dans lesquels on répond à une question par OUI ou NON.

Ex : PAIR : Entrée : Un entier n
Sortie : Oui ssi n est pair.

→ On peut transformer un problème d'optimisation (trouver la meilleure solution) en un problème de décision (est-ce qu'il existe une solution assez bonne ?)

EX : Voyageur de commerce, version décisionnelle :

PVC : Entrée : Graphe G complet pondéré, k une valeur.
Sortie : Oui ssi il existe un cycle hamiltonien de poids inférieur ou égal à k.

2) Classes de complexité

Def : Le temps d'exécution d'une machine de Turing M est la fonct° $t : \mathbb{N} \to \mathbb{N}$ telle que $\boxed{t(n) = \max_{|w|=n}}$ (Nb d'étapes de calcul de M sur une branche qcq de son exécution sur w

FAUX
max min pour sur

→ On définit pour $f : \mathbb{N} \to \mathbb{N}$, les classes de complexité :

TIME(f) = Ensemble de problèmes décidés par une machine de Turing déterministe de temps d'exec t tel que $t(n) = O(f(n))$

NTIME(f) = ——— non-déterministe ———

Non déterministe : "ACCEPTER" plutôt que décider.

Def : $P = \bigcup_{k \in \mathbb{N}} TIME(n \mapsto n^k)$ et $NP = \bigcup_{k \in \mathbb{N}} NTIME(n \mapsto n^k)$

prop : $P \subseteq NP$

3) Réductions

: On formalise le fait qu'un problème soit plus dur qu'un autre problème

Def : A se réduit à B ssi il existe une machine M déterministe qui calcule f telle que $w \in A$ ssi $f(w) \in B$

Def (Réduction polynomiale) La réduction est polynomiale ssi $\exists k \in \mathbb{N}$ tel que $t_M = O(n^k)$. On note $A \leq_P B$

→ On identifie un problème de décision A à l'ensemble des encodages de ses instances positives. (Le codage est souvent défini implicitement)

Exemple : Chercher un ensemble de sommets stable dans un graphe G est équivalent à chercher une clique dans son complémentaire.

4) Classe NP

Def : (Vérificateur) Un vérificateur d'un pb A est une machine terminante M tel que $w \in A$ ssi $\exists c$, tel que M accepte $\langle w, c \rangle$. c est appelé "certificat."

→ le vérificateur est polynomial ssi M est déterministe polynomiale et c est polynomiale

prop : $A \in NP$ ssi A admet un vérificateur polynomial

Ex : On définit SAT : Entrée : Une formule φ sous CNF (calcul prop.)
Sortie : Oui ssi φ est satisfiable.

prop : SAT $\in$ NP. Étant donné une valuation v, on peut dire en temps polynomiale si $\varphi[v] = 1$ → Intérêt pratique pour dire si $A \in NP$

Def (NP-dur) : Un problème est NP-dur ssi tout problème de la classe NP se réduit polynomialement à lui.

Def : A est NP-complet ssi $A \in NP$ et A est NP-dur

Rem : Pour montrer qu'un problème est NP-complet, on montre qu'il est dans NP et qu'un problème NP-complet se réduit (polynomialement) à lui

→ Il faut donc déterminer un premier problème NP-complet.

Ecrire : Réduc $A \leq_P B$

- Si $A \in$ NP-complet alors B est NP-dur
- $B \in NP$ (ou P) alors $A \in NP$ (ou P).

⚠ CNF-SAT ou SAT ? (En fait, c'est pareil)

Thm [Cook] : SAT est NP-complet [DVLT]

Corollaire : 3-SAT est NP-complet. → On réduit SAT à 3-SAT.

II. Stratégies de réduction [GAR]

On cherche à prouver que B est NP-complet en réduisant un problème NP-complet A à B. On peut distinguer trois types principaux de stratégies de réduction, que l'on peut éventuellement mixer.

1) Restriction du problème.

On est dans le cas où A est un sous-problème NP-complet de B.

Exs : 3-SAT $\leq_p$ SAT et HAMV $\leq_p$ HAM$_{orienté}$ ← non orienté / de général est orienté

· Cas où la restriction est moins explicite. PARTITION est une restriction de SAD

PARTITION : Entrée : Ensemble A fini, $s : A \to$
Sortie : Oui ssi il existe $A' \subseteq A$ /
$$\sum_{A'} s(a) = \sum_{A \setminus A'} s(a)$$

SAD : Entrée : Ensemble A fini, $s, v : A \to \mathbb{N}^+$, $K, B \in \mathbb{N}^+$
Sortie : Oui ssi $\exists A' \subseteq A$ /
$$\sum_{A'} s(a) \leq B \text{ et } \sum_{A'} v(a) \geq K$$

[Papa]

→ PARTITION est une restriction de SAD : On prend $s = v$ et $K = B = \left(\sum_A s(a)\right) \cdot \frac{1}{2}$ } donc PARTITION $\leq_p$ SAD

2) Remplacement local

→ On modifie "localement" une instance du problème A pour obtenir (après un nombre polynomial de modifications) une instance de B sans avoir modifié la propriété d'être une instance positive de A.

Ex1 : On réduit SAT à 3-SAT : A une clause de longueur k $(l_1 \vee \dots \vee l_k)$ on associe deux clauses, une de longueur 3 et une autre de longueur k-1, en rajoutant une variable $y \in (l_1, \dots, l_k)$:

$$l_1 \vee \dots \vee l_k \equiv (l_1 \vee l_2 \vee y) \wedge (\bar{y} \vee l_3 \vee \dots \vee l_k)$$

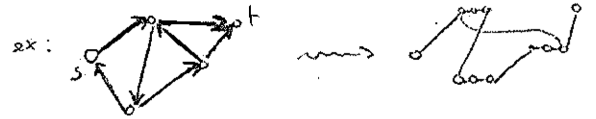
Ex2 : On fait une réduction de HAM$_{orienté}$ à HAM.

HAM$_{orienté}$: Entrée : $G = (S, A)$: graphe orienté, $s, t \in S$
Sortie : Oui ssi $\exists$ un chemin hamiltonien allant de s à t

HAM : Pareil pour un graphe non-orienté

Idée de la construction : On construit $G' = (S', A')$ non orienté tq $\forall u \in S \setminus \{s, t\}$, $u_{in}, u_m, u_{out} \in S'$, $s_{out} \in S'$ et $t_{in} \in S'$, puis : $(u_{out}, v_{in}) \in A'$ ssi $(u, v) \in A$ et u_{in} — u_m — u_{out}

ex :



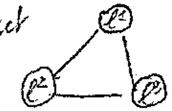
3) Création de gadgets : On veut réduire A à B.

→ On prend une instance w_A de A et on identifie ses "briques élémentaires" (une clause pour SAT par ex.) et on construit une brique élémentaire dans le vocabulaire du problème B. On construit alors une instance de B (w_B) (polynomialement) telle que w_A positive ssi w_B positive.

Ex1 : 3-SAT $\leq_p$ VERTEX-COVER _ Soit φ une formule 3-SAT (n variables, m clauses)

VERTEX-COVER : Entrée : Graphe $G = (S, A)$, $k \in \mathbb{N}$
Sortie : Oui ssi, $\exists S' \subseteq S$ tel que $|S'| \leq k$ et $\forall (x, y) \in A$, $x \in S'$ ou $y \in S'$

Idée de la réduction : A chaque variable x_i ($i \in \{1 \dots n\}$) on associe un gadget x_i — $\bar{x_i}$ et à chaque clause $(l_j^1 \vee l_j^2 \vee l_j^3)$ ($j \in \{1 \dots m\}$) on associe le gadget



. On met une arête entre x_i et l_j^r si ils sont égaux

→ le graphe obtenu valide VERTEX-COVER pour $k = n + 2m$ ssi φ est satisfiable.

Ex2 : Réduction de 3-SAT à HAM$_{orienté}$.

4) Schéma des réductions

SAT $\xrightarrow{2}$ 3SAT $\xrightarrow{3}$ HAM$_{orienté}$ $\xrightarrow{1}$ HAM $\xrightarrow{2}$ HAM$_{cycle}$ $\xrightarrow{1}$ PVC

3SAT $\longrightarrow$ VERTEX_COVER

3SAT $\xrightarrow{3}$ PARTITION $\xrightarrow{1}$ SAD

Rem : Comme tous ces problèmes sont NP-complet, n'importe quel diagramme de ce type est valide en théorie.

III Contourner la NP-complétude

1) P=NP

→ A ce jour, on ne sait pas dire si P ≠ NP ou pas.
→ Si on prouve qu'un problème NP-complet peut être résolu en temps polynomial, alors P=NP
. Si P≠NP, a priori, tout algorithme déterministe décidant un problème NP-complet serait ~~exponentiel~~ sur-polynomial. (Rq NP ⊆ EXPTIME mais ⊊ ? ON SAIT PAS)

Comment réduire cette complexité exponentielle ?

$\boxed{1}$ Se restreindre à des instances sur lesquelles on connaît un algorithme polynomial.
Ex: 3-COLOR. Entrée: Graphe G, Sortie: Oui ssi il existe un coloriage à 3 couleurs.
→ 3-COLOR est NP-complet mais si on se restreint aux instances des graphes d'intervalle, on connaît un algorithme glouton (donc polynomial) décidant le problème.
→ 2-SAT est décidé en temps polynomial
→ HAMcycle si le degré du graphe est ≤ 2

$\boxed{2}$ On tente d'améliorer le temps de calcul sans perte de généralités.
→ heuristiques
→ Backtracking
→ Methodes gloutonnes
→ Algorithmes génétiques
→ Optimisation locale

Insister (i.e. sans sens d'approcher)

$\boxed{3}$ Algorithmes d'approximation :

Pas au programme.

Def: Soit A un problème $\boxed{\text{d'optimisation}}$ et $\rho > 1$. On note $v(A)$ la valeur optimale (minimale) de A. On dit que ϕ est un algorithme de ρ-approximation de A si pour toute instance de A, ϕ retourne une solution de valeur inférieure à $\rho v(A)$.

c'est bon qd ϕ polynômial.

ouverture/conclu ?

Exs: Il existe un algorithme de ρ-approximation pour
- VERTEX-COVER : Algorithme glouton. avec $\rho = 2$
- SAD pour tout $\rho > 1$:

On dispose d'un algo (Prog Dyn) pseudo-polynomial en $O(n \times V)$ [Papa] avec $V = \sum v_i$ ⇝ On "tronque" les v_i pour ne garder que l'info la plus utile : $\hat{V}_i = \left\lfloor V_i \times \frac{n}{(\rho-1)V_{max}} \right\rfloor$: on a $\hat{V}_i \leq n$ d'où une complexité en $O(n^3)$ tout en gardant une bonne approx de la solution

Ex : Algo de 2-approx de PVC où les poids du graphe vérifient l'inégalité triangulaire (PVCE)

DEV2

Prop : S'il existe $\rho > 1$ et un algo de ρ-approx de PVC, alors P = NP

→ Polynom

Références :

[SIP] : M. Sipser "Introduction to the Theory of Computation"
[GAR] : R. Garey / D.S. Johnson "Computers and Intractability"
[Papa] : Papadimitriou

. Pb en théorie des langages ? (Séparation par automate)

• Plus courte sur-séquence commune sur un ENSEMBLE de mots ? (Pr 2 c'est polynômial).

• — Longue

Dvpts : - Séparaᵒ par automate
- HAM

2. Approximation pour le problème du voyageur de commerce euclidien en temps polynomial

2PVCE

<u>Entrée</u> : $G = (S, S^2)$: graphe complet (non orienté)

$\omega : S^2 \longrightarrow \mathbb{N}$ tq $\forall u, v, w$

- $\omega(u,v) = \omega(v,u)$
- $\omega(u,v) + \omega(v,w) \leqslant \omega(u,w)$

<u>Sortie</u> : Un cycle hamiltonien C_S tq $\omega(C_S) \leqslant 2\,\omega(C_{Min})$ où C_{Min} est un cycle hamiltonien de poids minimal

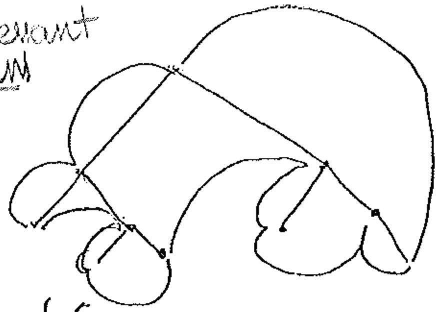
$AC_{Min} \longleftarrow$ Prim (G, ω) : arbre couvrant de poids minimal

$C_A \longleftarrow$ Chemin obtenu en effectuant un parcours en profondeur de AC_{Min}

$C_S \longleftarrow$ Chemin obtenu à partir de C_A en enlevant les sommets déjà parcourus.

Retourner C_S

↳ Pq on montre ça ? Préciser que c'est interessant dans le plan



<u>Fait</u> : Il existe une implementation polynomiale de 2PVCE

<u>Preuve de correction</u> : • C_S est un cycle hamiltonien :

C_S passe par tous les sommets de AC_{Min}, donc tous les sommets de G, et une unique fois par sommet. (Et, G étant complet, les arrêtes de C_S sont valides)

• $\omega(C_S) \leqslant 2\,\omega(C_{Min})$

$$\left.\begin{array}{l}(i)\ \omega(AC_{Min}) \leqslant \omega(C_{Min}) \\ (ii)\ \omega(C_A) = 2\,\omega(AC_{Min}) \\ (iii)\ \omega(C_S) \leqslant \omega(C_A)\end{array}\right\} \Rightarrow \omega(C_S) \leqslant 2\,\omega(C_{Min})$$

(i) $\forall a \in C_{Min}$, $C_{Min} \setminus \{a\}$ peut être vu comme un arbre couvrant

d'où $\omega(C_{Min} \setminus \{a\}) \geqslant \omega(\mathcal{A}C_{Min})$ et $\omega(C_{Min}) \geqslant \omega(\mathcal{A}C_{Min})$ puisque $\omega(a) \geqslant 0$

(ii) Par construction, C_1 possède exactement deux fois chaque arête de C_{Min}

(iii) On écrit $C_1 = \alpha_1 \beta_1 \dots \alpha_p \beta_p \alpha_{p+1}$ tq

α_i, β_i sous chemins avec $|\alpha_i| \geqslant 0$, $|\beta_i| \geqslant 2$ et les sommets internes de β_k ont été parcourus par l'un des α_i, $i \leqslant k$

Pour $\beta_i = (u_i, u_i'), \dots, (v_i', v_i)$ on a alors :

$$C_S = \alpha_1 (u_1, v_1) \dots \alpha_p (u_p, v_p) \alpha_{p+1}$$

Par l'inégalité triangulaire, on a $\omega(u_i, v_i) \leqslant \omega(\beta_i)$

et $\omega(C_S) \leqslant \omega(C_1)$.

<u>Théorème</u> : Si $\exists \rho > 1$ et ρ^{PVC} un algo de ρ-Approx de PVC polynomial, alors $P = NP$.

<u>P.HAM.PATH</u> : <u>Entrée</u> $(G = S, A)$: graphe non orienté

<u>Sortie</u> Oui ssi G possède un cycle hamiltonien

$G' \longleftarrow (S, S^2)$

$$\omega \longleftarrow \begin{pmatrix} S^2 \longrightarrow \mathbb{N} \\ (u,v) \longmapsto 1 \text{ si } (u,v) \in A \\ \rho(|S|+1) \text{ sinon} \end{pmatrix}$$

$C_{Approx} \longleftarrow \rho\,PVC(G', \omega)$

Si $\omega(C_{Approx}) = |S|$ alors Oui

Sinon Non

Ici, plutôt que ça, on pourrait faire une <u>réduction</u> (P.HAM à ρ-PVC).

Fait : Il existe une implémentation polynomiale de P.HAM.PATH.

Lemme 1 : P. HAM. PATH est correct, i.e.

$\omega(C_{Approx}) = |S|$ ssi G possède un cycle hamiltonien

Rq : $\forall C$ cycle hamiltonien de G', C possède $|S|$ arêtes de poids ≥ 1, donc $\omega(C) \geq |S|$

$\boxed{\Rightarrow}$ Les arêtes de C_{Approx} sont toutes de poids 1, elles sont donc dans A et C_{Approx} est un cycle hamiltonien de G.

$\boxed{\Leftarrow}$ Soit C_H un cycle hamiltonien de G, on a : $\omega(C_H) = |S|$: poids minimal

d'où $\omega(C_{Approx}) \leq \rho|S|$: Toutes les arêtes de C_{Approx} sont de poids 1, et $\omega(C_{Approx}) = |S|$.

Lemme 2 : HAM. PATH est NP. complet

$\longrightarrow$ Pour tout problème D de NP, il existe un algorithme polynomial qui décide D.