928 : Problèmes NP-complets : exemples de réductions

Antoine Dequay Louis Noizet

8 février

I Généralités : Classes de complexité

Definition 1 (Problème). *On distingue deux types de problème :*

- *Les problèmes de décision (PDD), qui attendent une réponse binaire,*
- *Les problèmes d'optimisation (PDO), qui attendent une solution optimale à un problème donné.*

Definition 2 (Problème P). *Un problème est dit dans P s'il existe une machine de* Turing *déterministe qui décide ce problème en temps polynomial.*

Exemple 3 (Connexité d'un graphe). *Le PDD de la connexité d'un graphe est dans P.*

Definition 4 (Problème NP). *Un problème est dit dans NP s'il existe une machine de* Turing *non nécessairement déterministe qui décide ce problème en temps polynomial, c'est à dire qu'il existe au moins une exécution de la machine qui termine sur un état final.*

Exemple 5 (SAT). *Le PPD* SAT *:*

$$\begin{cases} \underline{\text{entrée}} : \text{une formule } \varphi \text{ en logique propositionnelle,} \\ \underline{\text{sortie}} : \text{oui si } \varphi \text{ est satisfiable, non sinon.} \end{cases}$$

est dans NP.

Definition 6 (Réduction). *On appelle réduction polynomiale d'un problème A à un problème B, une fonction f calculable en temps polynomial par une machine de* Turing *déterministe telle que : w est une instance positive de $A \iff f(w)$ est une instance positive de B.*

Si une telle réduction existe, on dit que A se réduit à B. ***Annexe.***

Definition 7 (NP-complet). *Un problème A est dit NP-dur si, pour tout problème L dans NP, L se réduit à A.*

Un problème est dit NP-complet s'il est NP-dur et dans NP.

Remarque 8. *On ne sait toujours pas si $P \neq NP$ ou non. On a : $P = NP$ si et seulement si il existe un problème NP-complet dans P.*

Théorème 9 (Cook, Développement). *Le PDD* SAT *est NP-complet.*

Remarque 10. *Tout problème n'est pas P ou NP. Par exemple, le calcul de 2^n pour $n \in \mathbb{N}$ est dans EXPTIME.*

II Problèmes NP-complets

Annexe

A Logique

Théorème 11. *Le PPD* CnfSat *:*
$$\begin{cases}\text{\underline{entrée} : une formule } \varphi \text{ de la logique propositionelle sous forme normale conjonctive (FNC),}\\ \text{\underline{sortie} : oui si } \varphi \text{ est satisfiable, non sinon.}\end{cases}$$
est NP-complet.

Théorème 12. *Le PPD* 3CnfSat *:*
$$\begin{cases}\text{\underline{entrée} : une formule } \varphi \text{ de la logique propositionelle sous FNC avec 3 littéraux par clause,}\\ \text{\underline{sortie} : oui si } \varphi \text{ est satisfiable, non sinon.}\end{cases}$$
est NP-complet.

Théorème 13. *Le PPD* Max2Sat *:*
$$\begin{cases}\text{\underline{entrée} : une formule } \varphi \text{ de la logique propositionelle sous FNC avec 2 littéraux par clause, } k \in \mathbb{N},\\ \text{\underline{sortie} : oui si on peut trouver au moins } k \text{ clauses de } \varphi \text{ qui peuvent être satisfiables simultanément, non sinon.}\end{cases}$$
est NP-complet.

B Graphes

Théorème 14. *Le PPD* Vertex-Cover *:*
$$\begin{cases}\text{\underline{entrée} : un graphe } G = \{S, A\} \text{ non-orienté et } j \leq |S|,\\ \text{\underline{sortie} : oui s'il existe } S' \subset S \text{ avec } |S'| \leq j \text{ telle que pour tout } uv \in A,\ u \in S' \text{ ou } v \in S', \text{ non sinon.}\end{cases}$$
est NP-complet. ***Annexe***

Théorème 15. *Le PPD* Clique *:*
$$\begin{cases}\text{\underline{entrée} : un graphe } G = \{S, A\} \text{ non-orienté et } j \leq |S|,\ j \in \mathbb{N},\\ \text{\underline{sortie} : oui s'il existe une clique de taille } j \text{ dans } G, \text{ non sinon.}\end{cases}$$
est NP-complet. ***Annexe***

Théorème 16. *Le PPD* IndepSet *:*
$$\begin{cases}\text{\underline{entrée} : un graphe } G = \{S, A\} \text{ non-orienté et } j \leq |S|,\ j \in \mathbb{N},\\ \text{\underline{sortie} : oui si } G \text{ contient un ensemble indépendant de taille } j, \text{ non sinon.}\end{cases}$$
est NP-complet. ***Annexe***

Théorème 17. *Le PPD* k-Coloriage *:*
$$\begin{cases}\text{\underline{entrée} : un graphe } G = \{S, A\} \text{ non-orienté,}\\ \text{\underline{sortie} : oui s'il existe une application } c : S \to [\![1, k]\!] \text{ telle que pour tout } uv \in A,\ c(u) \neq c(v), \text{ non sinon.}\end{cases}$$
est NP-complet pour $k \geq 3$.

Théorème 18. *Le PPD* HamCirc *:*
$$\begin{cases}\text{\underline{entrée} : un graphe } G = \{S, A\} \text{ non-orienté,}\\ \text{\underline{sortie} : oui si } G \text{ admet un circuit hamiltonien, non sinon.}\end{cases}$$
est NP-complet. ***Annexe***

Théorème 19. *Le PPD* PVC *:*
$$\begin{cases}\text{\underline{entrée} : un graphe } G = \{S, A\} \text{ non-orienté complet pondéré par des entier relatifs, } j \in \mathbb{N},\\ \text{\underline{sortie} : oui s'il existe un cycle hamiltonien de poids inférieur à } j, \text{ non sinon.}\end{cases}$$
est NP-complet.

C Paquets

Théorème 20. *Le PPD* SubsetSum *:*
$$\begin{cases}\text{\underline{entrée} : un ensemble fini } A \subset \mathbb{N},\ t \in \mathbb{N},\\ \text{\underline{sortie} : oui s'il existe } A' \subset A \text{ tel que } \sum_{a \in A'} a = t, \text{ non sinon.}\end{cases}$$
est NP-complet.

Théorème 21. *Le PPD* BinPacking *:*
$$\begin{cases}\text{\underline{entrée} : un ensemble fini } A,\ w : A \to \mathbb{N},\ c, k \in \mathbb{N},\\ \text{\underline{sortie} : oui s'il existe } r : A \to [\![1, k]\!] \text{ telle que } \forall j \in [\![1, k]\!],\ \sum_{a \in r^{-1}(j)} w(a) \leq c, \text{ non sinon.}\end{cases}$$
est NP-complet. ***Annexe***

Théorème 22. *Le PPD* PARTITION *:*

$\begin{cases} \underline{\textit{entrée}} : \textit{un ensemble fini } A,\ w : A \to \mathbb{N}, \\ \underline{\textit{sortie}} : \textit{oui s'il existe } A' \subset A \textit{ tel que } \sum_{a\in A'} w(a) = \sum_{a\notin A'} w(a), \textit{ non sinon.} \end{cases}$

est NP-complet.

Théorème 23. *Le PPD* SAC-À-DOS *:*

$\begin{cases} \underline{\textit{entrée}} : \textit{un ensemble fini } A,\ w : A \to \mathbb{N},\ v : A \to \mathbb{N}, P, V \in \mathbb{N}, \\ \underline{\textit{sortie}} : \textit{oui s'il existe } A' \subset A \textit{ tel que } \sum_{a\in A'} w(a) \leq P \textit{ et } \sum_{a\in A'} w(a) \geq V, \textit{ non sinon.} \end{cases}$

est NP-complet.

D Automates et langages

Théorème 24 (DÉVELOPPEMENT)**.** *Le PPD* SEP *:*

$\begin{cases} \underline{\textit{entrée}} : S \textit{ et } T \textit{ deux langages finis disjoints et } k \in \mathbb{N}, \\ \underline{\textit{sortie}} : \textit{oui si il existe un automate à } k \textit{ états qui accepte les mots de } S \textit{ et rejette ceux de } T, \textit{ non sinon.} \end{cases}$

est NP-complet.

Théorème 25. *Le PPD* ESE *:*

$\begin{cases} \underline{\textit{entrée}} : E \textit{ une expression rationnelle sans étoile, } A^n \textit{ un langage,} \\ \underline{\textit{sortie}} : \textit{oui si le langage de } E \textit{ est différent de } A^n, \textit{ non sinon.} \end{cases}$

est NP-complet.

III Au delà de la NP-complétude

A Simplifier en contraignant

Théorème 26. *Le PPD* 2CNFSAT *:*

$\begin{cases} \underline{\textit{entrée}} : \textit{une formule } \varphi \textit{ de la logique propositionelle sous FNC avec 2 littéraux par clause,} \\ \underline{\textit{sortie}} : \textit{oui si } \varphi \textit{ est satisfiable, non sinon.} \end{cases}$

est dans P.

Théorème 27. *Le PPD* HORNSAT *:*

$\begin{cases} \underline{\textit{entrée}} : \textit{une formule } \varphi \textit{ de } \text{HORN}, \\ \underline{\textit{sortie}} : \textit{oui si } \varphi \textit{ est satisfiable, non sinon.} \end{cases}$

est dans P.

Théorème 28. *Le PPD* 2-COLORIAGE *est dans P.* ***Annexe***

B Méthodes approchées

Definition 29 (f-approximation)**.** *Pour un PDO A donné, on appelle f-approximation de A, pour $f \in \mathbb{R}$, est un algorithme qui, pour une instante x donnée et $C^*(x)$ le coût d'une solution optimale de A pour x, renvoie une solution de coût $C(x)$ telle que*

$$\max\left(\frac{C(x)}{C^*(x)}, \frac{C^*(x)}{C(x)}\right) \leq f.$$

Exemple 30. *L'algorithme :*

> $C \leftarrow \emptyset$
> $A' \leftarrow A$
> ***while*** $A' \neq \emptyset$ ***do***
>   *Soit* (u, v) *une arête de* A'
> $C \leftarrow C \cup \{u, v\}$
>   *Supprimer de* A' *les arêtes contenant* u *ou* v
> ***end while***
> ***return*** C

est une 2-approximation polynomiale de VERTEX-COVER.

Exemple 31. *L'algorithme :*

> *choisir un sommet* $r \in S$
> *construire un arbre* T *couvrant minimal à partir de* r *via l'algorithme de* PRIM
> ***return*** *le cycle hamiltonien qui visite les sommets de* T *dans l'ordre d'un parcours préfixe*

est une 2-approximation polynomiale de PVC *dans le cas où l'inégalité triangulaire est vérifiée.*

Références

— Le Barbenchon,
— Carton,
— Cormen.

Annexe

Cf. messenger pour les dessins : 6, II (rajouter une flèche de 3SAT à ESE, pas besoin de légende ou de couleurs), 14, 15, 16 (rajouter une arête entre le premier et le dernier sommet du chemin et continuer le chemin pour représenter un cycle), 18, 21, 28