NP complétude

Montaigne 2023-2024

– mpi23@arrtes.net –

Classe NP

Pour certains problèmes algorithmiques, il est possible d'éviter le recours à la **brute-force** en trouvant des algorithmes de complexité temporelle polynomiale.

Pour d'autres problèmes, un tel résultat n'a pas, à ce jour, été obtenu. Aucun algorithme de complexité temporelle polynomiale n'a été trouvé.

Classe NP

À la question de savoir **pourquoi** aucun algorithme efficace n'a été trouvé, on ne sait répondre pour deux raisons essentielles.

- Peut-être qu'une **solution efficace non encore révélée** pourrait être trouvée.
- Peut-être que le problème **ne peut pas** être résolu par un algorithme polynomial ; le problème est **intrinsèquement** difficile.

Vérifier vs trouver

Considérons un problème pour lequel aucun algorithme polynomial ne permette de trouver une solution.

Supposons toutefois qu'une solution ait été découverte (par chance ou par mise en œuvre de la brute-force).

Il est alors facile de convaincre qui que ce soit qu'il s'agit effectivement d'une solution. Il suffit de le **vérifier**.

Et généralement, **vérifier** l'existence d'une solution est plus facile que **déterminer** son existence.

Problème HamPath

Illustrons notre propos avec le problème du chemin hamiltonien. Si G est un graphe orienté, un **chemin hamiltonien** dans G est une chemin orienté ne passant qu'une seule fois par chacun de ses sommets.

Considérons le problème suivant.

$$\textsc{HamPath} = \left\{ \langle G, s, t \rangle \mid \begin{array}{l} G \text{ est un graphe orienté avec} \\ \text{un chemin hamiltonien entre} \\ \text{les sommets } s \text{ et } t \end{array} \right\}$$

À ce jour, il n'existe aucun algorithme polynomial pour résoudre $\textsc{HamPath}$.

En revanche, si on trouve un chemin hamiltonien, il est aisé de vérifier que c'est une solution. Et cette vérification peut se faire en temps polynomial en la taille du chemin.

Problème Composite

Un entier naturel peut être décomposé en facteurs premiers. On dit qu'il est composé. Considérons le problème suivant.

$$\textsc{Composite} = \{x \in \mathbb{N}, x > 1 \mid x = pq \text{ avec } p, q, > 1\}$$

On ne connaît pas d'algorithme polynomial pour trouver un entier diviseur de x.

En revanche, un algorithme peut vérifier en temps polynomial qu'un entier divise x.

Vérificateur et certificat

Les observations précédentes ont mené à introduire les notions de **vérificateur** et de **certificat**.

Un **vérificateur** V pour un problème donné est un algorithme qui reçoit deux informations :

- une **instance positive** du problème ;
- un **certificat**.

Un vérificateur est dit polynomial s'il s'exécute en temps polynomial en la taille de l'instance. Ce qui n'est possible que si le certificat est lui-même de longueur polynomiale en la taille de cette même instance.

Exemples

Pour HAMPATH, un certificat pour une chaîne $\langle G, s, t \rangle$ est simplement un chemin hamiltonien reliant s à t.

Pour COMPOSITE, un certificat pour un entier x est simplement l'un de ses diviseurs.

Dans les deux cas, un vérificateur vérifie en temps polynomial que la donnée d'entrée appartient au langage, étant donné le certificat.

Classe NP

NP est la classe des problèmes de décision qui admettent des vérificateurs polynomiaux.

Par définition, $P \subseteq NP$. À ce jour, on ne sait si l'inclusion inverse est vraie.

Classe NP

On peut reformuler la définition d'un **problème de classe NP**.

Définition 1

Un problème de décision défini par une fonction $f : E \to \mathbb{B}$ est dans la **classe NP** si :

- il existe un ensemble C de **certificats** et une fonction $g : E \times C \to \mathbb{B}$ telle que $f(e) = \mathrm{V}$ si et seulement s'il existe un $c \in C$ de taille polynomiale en la taille de e tel que $g(e, c) = \mathrm{V}$;
- le problème de décision défini par g, appelé problème de **vérification** d'un certificat, est dans la classe P.

Exemple

Le problème SATappartient bien à la classe NP. On le justifie avec les éléments suivants.

- On prend comme ensemble de certificats l'ensemble des valuations et on définit la fonction g telle que $g(e, c) = \mathrm{V}$ si et seulement si la valuation c rend vraie la formule e.
- Si une formule e est effectivement satisfiable alors il existe des valuations c telles que $g(e, c) = \mathrm{V}$. En particulier il en existe au moins une qui ne mentionne que les variables effectivement présentes dans e, dont la taille est donc majorée par celle de e.
- Étant données une formule et une valuation, on peut déterminer si la formule est vraie en un temps linéaire.

Exemple

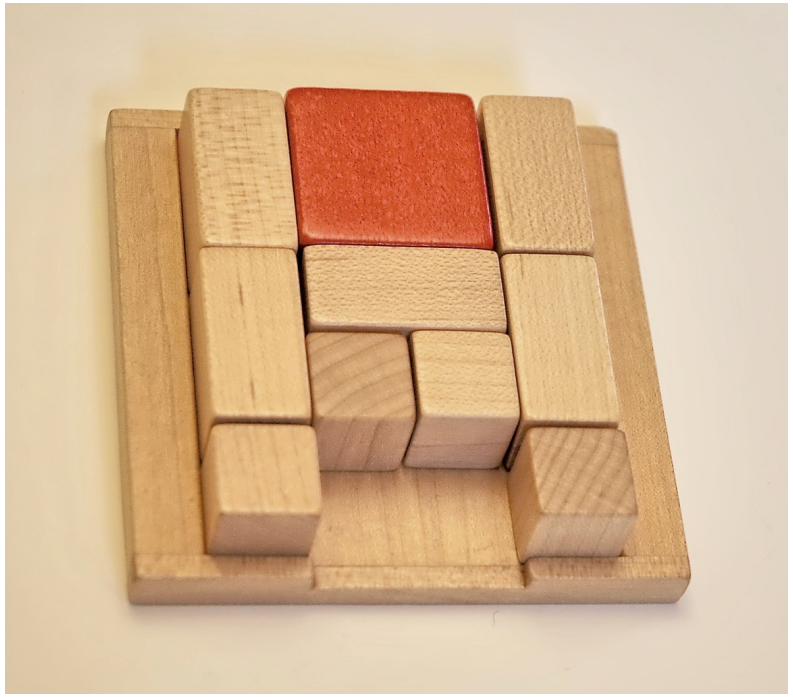
Dans le jeu de Sudoku généralisé d'ordre n, on a une grille de côté n^2 subdivisée en n^2 carrés de taille $n \times n$, à remplir avec n^2 symboles. Chaque symbole doit apparaître exactement une fois dans chaque ligne, chaque colonne, et chaque carré $n \times n$. Le Sudoku usuel est celui d'ordre 3.
Ce jeu de Sudoku est un problème de recherche : on veut trouver, s'il en existe, une manière valide de compléter une grille partiellement remplie fournie en entrée. Le problème associé d'existence d'une solution est dans la classe NP, car :

- une solution peut être donnée sous la forme d'une grille complétée, dont la taille est comparable à celle de l'entrée ;
- étant donnée une solution potentielle c à une grille e, on peut effectivement vérifier en temps qu'une polynomial d'une part que c est valide, et d'autre part qu'elle complète bien e.

Un contre-exemple

On pourrait tenter de justifier que le problème de l'existence d'une solution à partir d'une configuration de l'âne rouge généralisé est également dans la classe NP, en prenant comme certificats les séquences de coups permettant de résoudre ce jeu. En effet, étant donnée une liste de coups, on peut facilement les reproduire et vérifier qu'ils mènent à une position gagnante. Cependant, on ne sait pas établir que de tels certificats respectent la contrainte de taille polynomiale !

En réalité, le problème de l'âne rouge généralisé est fortement soupçonné de ne pas appartenir à la classe NP.



Réduction

Soient A et B deux problèmes de décision. Un **réduction** de A vers B est une fonction f calculable en **temps polynomial** telle que :

$$w \in \text{Oui}(A) \iff f(w) \in \text{Oui}(B)$$

On note :

$$A \leqslant_p B$$

Cette notation indique de A est « plus facile » que B.

La relation $\leqslant_p$ est

- réflexive : un problème est aussi facile (ou difficile) que lui-même ;
- transitive : la composée de deux fonctions calculables en temps polynomial est calculable.

Réduction

Les deux résultats suivants sont importants.

- Si $A \leqslant_p B$ et si B est dans P alors A est dans P.
- Si $A \leqslant_p B$ et si A n'est pas dans P alors B n'est pas dans P.

NP complétude

Parmi les problèmes de la classe NP, en existe-t-il de plus difficiles que d'autres ?

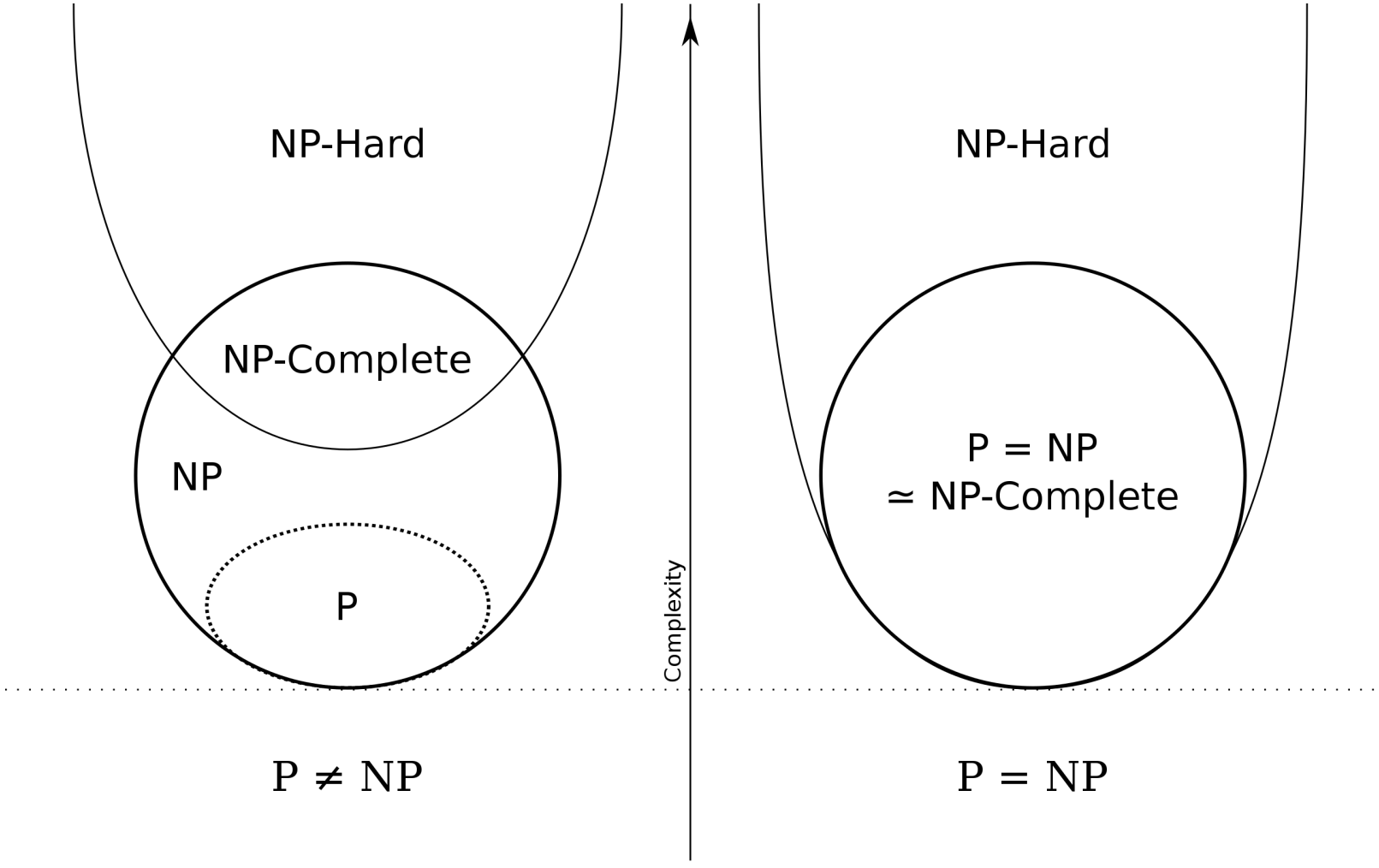
- Un problème A est dit **NP-difficile** si tout autre problème B de NP vérifie $B \leqslant_p A$.

  Intuitivement, A est plus difficile que tous les problèmes dans la classe.

- Un problème A est dit **NP-complet** si de plus $A \in$ NP.

  A est un élément maximum pour $\leqslant_p$.

Emboitement des classes



Problème SAT

SAT est le premier problème pour lequel l'appartenance à la classe des problèmes NP-complets a été prouvée. Ce résultat constitue le **théorème de Cook-Levin** (admis).

Le problème **SAT** est NP-complet.

Conséquences fondamentales

Théorème 2

$P = NP$ si et seulement si $SAT \in P$.

Démonstration

- Si P = NP alors puisque SAT est dans NP, alors SAT est aussi dans P.
- Réciproquement, si SAT est dans P, puisque SAT est complet, pour tout problème $B \in$ NP, $B \leqslant_p$ SAT. Par conséquent B est dans P.

Ce résultat se généralise à n'importe quel problème NP-complet.

Si A est un problème NP-complet,
alors P = NP si et seulement $A \in$ P.

Ce qui motive l'intérêt de produire des problèmes NP-complets.

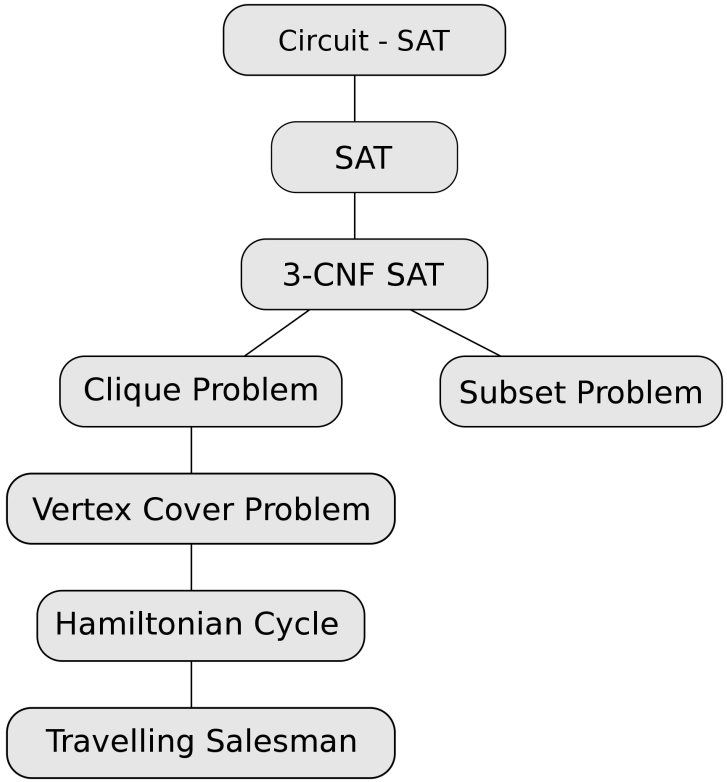
Stratégie

Pour **établir la NP-complétude** d'un problème A, on procède en deux temps.

- Prouver qu'il admet un **vérificateur polynomial**. Ce point garantit que $A \in \text{NP}$.
- Prouver que pour un problème NP-complet B, on a $B \leqslant_p A$.

Stratégie

En pratique, on s'appuie sur la connaissance préalable de problèmes NP-complets pour établir, par réduction, celle d'autres problèmes.



Gian Luca RuggeroActam
Public domain, via Wikimedia Commons

Exercice

Soient P_1 et P_2 deux problèmes de décision. Supposons $P_1 \leqslant_p P_2$. Répondre brièvement aux questions suivantes.

- **Question 1.** Si $P_1 \in \mathrm{P}$, a-t-on $P_2 \in \mathrm{P}$?
- **Question 2.** Si $P_2 \in P$, a-t-on $P_1 \in \mathrm{P}$?
- **Question 3.** Si P_1 est NP-complet, P_2 est-il NP-complet ?
- **Question 4.** Si P_2 est NP-complet, P_1 est-il NP-complet ?
- **Question 5.** Si $P_2 \leqslant_p P_1$, P_1 et P_2 sont-ils NP-complets ?
- **Question 6.** Si P_1 et P_2 sont NP-complets, existe-t-il une transformation polynomiale de P_2 en P_1 ?
- **Question 7.** Si $P_1 \in \mathrm{NP}$, P_2 est-il NP-complet ?

Exercice

On considère le problème PARTITION suivant.

Entrée :	un multi-ensemble fini S d'entiers positifs
Sortie :	peut-on partitionner $S = S_1 \cup S_2$ en deux multi-ensembles S_1 et S_2 tels que $\sum_{x \in S_1} x = \sum_{x \in S_2} x$?

Montrer que PARTITION $\in$ NP.

Exercice (solution)

Considérons l'ensemble de certificats suivant :

- certificats : les multi-ensembles d'entiers ;
- vérification : pour S une entrée du problème et C_1 un certificat (qui joue le rôle de S_1) :
  - on vérifie que $C_1 \subseteq S$ (en temps polynomial) ;
  - on calcule $S_2 = S \setminus C_1$ (en temps polynomial) ;
  - on vérifie que $\sum_{x \in C_1} x = \sum_{x \in S_2} x$ (en temps polynomial).

Ainsi S possède une solution au problème de PARTITION si et seulement s'il existe bien un certificat C_1 de taille polynomiale en $|S|$ qui va convenir (le $C_1 \subseteq S$ de la solution).

Exercice

On considère le problème SubsetSum suivant.

Entrée :	un multi-ensemble fini S d'entiers positifs, un entier naturel t
Sortie :	existe-t-il $S_1 \subseteq S$ tel que $\sum_{x \in S_1} x = t$?

Montrer que SubsetSum $\in NP$.

Exercice (solution)

Considérons l'ensemble des certificats suivant :

- certificats : les multi-ensembles d'entiers ;
- vérification : pour (S, t) une entrée du problème et S_1 un certificat :
  - on vérifie que $S_1 \subseteq S$ (en temps polynomial) ;
  - on vérifie que $\sum_{x \in S_1} x = t$ (en temps polynomial).

On a bien que (S, t) possède une solution au problème de SubsetSum si et seulement s'il existe bien un certificat S_1 de taille polynomiale en $|S|$ qui va convenir (le $S_1 \subseteq S$ de la solution).

Exercice

Selon que $\sum_{x \in S} x$ est pair ou impair, montrer que :

$$\textsc{Partition} \leqslant_p \textsc{SubsetSum}$$

Exercice (solution)

Soit S une instance de Partition. Construisons en temps polynomial une instance (S', t') de SubsetSum telle que S possède une solution si et seulement si (S', t') en a une.

- Si $\sum_{x \in S} x$ est impair, S n'a pas de solution au problème de Partition car on travaille avec des entiers. On construit alors une instance (S', t') n'ayant pas de solution pour SubsetSum. Par exemple : $S' = S$ et $t' = 1 + \sum_{x \in S} x$; cela se fait bien en temps polynomial.
- Si $\sum_{x \in S} x$ est pair, posons $S' = S$ et $t' = \left(\sum_{x \in S} x\right)/2$. Cela se fait bien en temps polynomial. S possède une solution pour le problème de Partition si et seulement si (S', t') possède une solution au problème SubsetSum (même S_1 dans les deux cas, et S_2 son complémentaire).

Exercice

Soit (S, t) une instance de SubsetSum, et soit $t' = \sum_{x \in S} x - t$. Justifier que (S, t) a une solution pour le problème SubsetSum si et seulement si (S, t') en a une.

Exercice (solution)

(S, t) possède une solution pour le problème SubsetSum si et seulement si (S, t') en possède une car il suffit de passer de S_1 à $S \setminus S_1$ pour passer d'une solution d'une instance à une solution de l'autre instance.

Exercice

Montrer que SubsetSum $\leqslant_p$ Partition.

Indication : si (S, t) est une instance de SubsetSum, on pourra construire une instance S' de Partition en rajoutant à S une valeur dépendant de t et de $\sum_{x \in S} x$ (distinguer trois cas).

Exercice (solution)

Montrons que $\textsc{SubsetSum} \leq_{\mathrm{P}} \textsc{Partition}$. Soit (S, t) une instance de $\textsc{SubsetSum}$. Construisons en temps polynomial une instance S' de $\textsc{Partition}$ telle que (S, t) possède une solution si et seulement si S' en a une.

- Si $2t = \sum_{x \in S} x$, alors on pose $S' = S$ et dans ce cas les deux problèmes sont équivalents.
- Si $2t > \sum_{x \in S} x$, on pose $S' = S \uplus \{x'\}$ avec $x' = 2t - \sum_{x \in S} x$.

Exercice (solution)

Remarquons que S' vérifie $\sum_{x \in S'} x = 2t$.

- S'il existe $S_1 \subseteq S$ tel que $\sum_{x \in S_1} x = t$, alors en posant $S_2 = S' \backslash S_1$, on a bien : $\sum_{x \in S_1} x = t = \sum_{x \in S_2} x$.
- Réciproquement, s'il existe $S' = S_1 \uplus S_2$ tels que $\sum_{x \in S_1} x = \sum_{x \in S_2} x$, alors ces deux sommes valent t, et l'une des deux parties S_1 ou S_2 ne contient pas x' : c'est donc une solution au problème SubsetSum. - si $2t < \sum_{x \in S} x$, on pose $S' = S \uplus \{x'\}$ avec $x' = \sum_{x \in S} x - 2t$. Posons également $t' = \sum_{x \in S} x - t$. Remarquons que S' vérifie $\sum_{x \in S'} x = 2 \cdot \sum_{x \in S'} x - 2t = 2t'$.

Exercice (solution)

De plus, d'après la question précédente, (S, t) possède une solution pour le problème SubsetSum si et seulement si (S, t') possède une solution pour le problème SubsetSum. Le raisonnement est alors analogue au cas précédent, en raisonnant cette fois-ci sur t' au lieu de t.

Cette construction se fait bien en temps polynomial dans tous les cas. Donc on a bien une réduction qui prouve que :

$$\textsc{SubsetSum} \leqslant_p \textsc{Partition}$$

Exercice

En déduire que Partition est NP-complet si et seulement si SubsetSum est NP-complet.

Exercice (solution)

- Si Partition est NP-complet (donc NP-difficile), puisque Partition $\leq_P$ Subset-Sum alors Subset-Sum est NP-difficile. De plus, Subset-Sum $\in$ NP. Donc Subset-Sum est NP-complet.
- Si Subset-Sum est NP-complet (donc NP-difficile), puisque Subset-Sum $\leq_P$ Partition (question [20b), alors Partition est NP-difficile. De plus, Partition $\in$ NP. Donc Partition est NP-complet.