Calculabilité et Complexité: CM 8

Florian Bridoux

Polytech Nice Sophia

2024-2025

Table des matières

Table des matières

Temps non déterministe polynomial et exponentiel

Définition

La classe NP est l'ensemble des langages reconnus par une machine non déterministe en temps polynomial, c'est-à-dire

$$NP = \mathbf{NTIME}(n^{O(1)}) = \bigcup_{k \in \mathbb{N}} \mathbf{NTIME}(n^k).$$

La classe NEXP (ou NEXPTIME) est l'ensemble des langages reconnus par une machine non déterministe en temps exponentiel, c'est-à-dire

$$NEXP = \mathbf{NTIME}(2^{n^{O(1)}}) = \bigcup_{k \in \mathbb{N}} \mathbf{NTIME}(2^{n^k}).$$

Attention, une erreur souvent commise est de croire que NP signifie "non polynomial" alors que c'est une abréviation pour Nondeterministic Polynomial time.

Exemple problèmes dans NP et NEXP

CLIQUE:
Entrée: un graphe non orienté G et un entier k ;
Question: G a-t-il une clique de taille k ?

Exemple problèmes dans NP et NEXP

CLIQUE:
Entrée: un graphe non orienté G et un entier k ;
Question: G a-t-il une clique de taille k ?

Une clique de taille k est un ensemble de k sommets tous reliés les uns aux autres. Ce problème est dans NP : une machine non déterministe polynomiale pour le reconnaître peut deviner un ensemble de k sommets puis vérifier que ces k sommets sont tous reliés entre eux. Nous verrons que ce problème est " NP- complet ".

Exemple problèmes dans NP et NEXP

ARRÊT NEXP :
Entrée: *le code d'une machine non déterministe N et un entier k en binaire ;*
Question: est-ce que $N(\epsilon)$ s'arrête en $\leq k$ étapes ?

Exemple problèmes dans NP et NEXP

ARRÊT NEXP :
Entrée: le code d'une machine non déterministe N et un entier k en binaire ;
Question: est-ce que $N(\epsilon)$ s'arrête en $\leq k$ étapes ?

Ce problème est dans NEXP car on peut simuler $N(\epsilon)$ pendant k étapes grâce à une machine universelle non déterministe : puisque la taille de l'entrée k est $log(k)$, cela prend un temps exponentiel. Nous n'avons pas encore vu cette notion, mais mentionnons au passage que ce problème est NEXP-complet.

Temps non déterministe polynomial et exponentiel

Remarque:

Si P est la classe des problèmes pour lesquels on peut trouver une solution efficacement, NP est la classe des problèmes pour lesquels on peut vérifier une solution efficacement. En effet, on nous donne une preuve et on doit vérifier en temps polynomial que cette preuve est correcte. En d'autres termes, on nous donne une solution potentielle (un chemin acceptant) et on doit vérifier qu'il s'agit en effet d'une solution. On retrouve cela dans l'exemple précédent de problèmes NP : pour CLIQUE, on choisissait (de manière non déterministe) l'ensemble formant la clique et il fallait vérifier qu'il formait bien une clique. Cet exemple illustre un phénomène général : NP est la classe des problèmes "faciles à vérifier".

Temps non déterministe polynomial et exponentiel

Corollaire

$P \subseteq NP \subseteq EXP \subseteq NEXP$

Pour rappel, le théorème de la hiérarchie en temps nous donnait le résultat suivant:

Théorème

$P \subset EXP$

Un équivalent non déterministe de cet argument nous donne:

Théorème

$NP \subset NEXP$

Cependant, c'est une question ouverte de savoir si chacune des inclusions $P \subseteq NP$, $NP \subseteq EXP$ et $EXP \subseteq NEXP$ est stricte.

Table des matières

co-NP

On va maintenant définir la classe co-NP: l'ensemble des langages dont le complémentaire est dans NP.

Définition

La classe co-NP est l'ensemble des langages A tels qu'il existe une machine de Turing non déterministe N fonctionnant en temps polynomial et satisfaisant : $x \in A$ ssi tous les chemins de calcul de $N(x)$ sont acceptants.

Remarque

Comme on l'a vu auparavant, on ne sait pas décider dans NP le complémentaire de tout langage de NP : cela se traduit par le fait que la question " NP = coNP ? " est ouverte.

Exemple de problème dans co-NP

Le "complémentaire" de n'importe quel problème dans NP:

co-CLIQUE:
Entrée: un graphe non orienté G et un entier k ;
Question: G n'a aucune clique de taille k ?

Si la réponse est non, c'est facile à prouver: il suffit de deviner un ensemble de k sommets et de vérifier que c'est une clique. Si la réponse est oui, on ne sait pas...

Table des matières

Les problèmes du prix du millénaire

Les problèmes du prix du millénaire sont un ensemble de sept défis mathématiques réputés insurmontables, posés par l'Institut de mathématiques Clay en 2000.
La résolution de chacun des problèmes est dotée d'un prix d'un million de dollars américains offert par l'institut Clay. En 2022, six des sept problèmes demeurent non résolus.
Liste des problèmes:

- Hypothèse de Riemann
- Conjecture de Poincaré - résolue par Grigori Perelman
- Problème ouvert $P = NP$
- Conjecture de Hodge
- Conjecture de Birch et Swinnerton-Dyer
- Équations de Navier-Stokes
- Équations de Yang-Mills

$P = NP$?

"Si quelqu'un prouve P = NP, la première chose qu'il devrait faire, c'est voler les 200 milliards de dollars qui existent en bitcoin. La deuxième, c'est résoudre tous les autres Problèmes du Prix du Millénaire" -Scott Aaronson

"Si P=NP, le monde serait très différent de celui que l'on pense qu'il est. Il n'y aurait [...] aucun écart fondamental entre résoudre un problème et reconnaître la solution une fois qu'elle est trouvée. Tous ceux capables d'apprécier une symphonie seraient Mozart; tous ceux capables de suivre un argument point par point seraient Gauss; tous ceux capables de reconnaître une bonne stratégie d'investissement seraient Warren Buffet" -Scott Aaronson

$P = NP$?

La question de savoir si $P = NP$ est ouverte et centrale en complexité. Pour reprendre la caractérisation précédente, cela revient à savoir si trouver une solution est "aussi simple" que de vérifier une solution. Par exemple:

- Est-il aussi facile de trouver une démonstration d'un énoncé mathématique que de vérifier qu'une démonstration donnée est correcte ?
- Est-il aussi facile de remplir une grille de Sudoku que de vérifier qu'un remplissage donné est bien une solution ?

On pourrait bien sûr multiplier les exemples. Notre expérience semble indiquer qu'il est plus difficile de trouver une solution, car il faut faire preuve d'imagination ou d'intuition : c'est une des raisons pour lesquelles de nombreuses personnes pensent que $P \neq NP$.

Table des matières

Réduction polynomiales

Définition (réductions many-one polynomiales)

Une réduction many-one en temps polynomial d'un problème A (sur l'alphabet Σ_A) vers un problème B (sur l'alphabet Σ_B) est une fonction $f : \Sigma_A^* \to \Sigma_B^*$ *calculable en temps polynomial* telle que :

$$\forall x \in \Sigma_A^*, x \in A \Leftrightarrow f(x) \in B.$$

Si une telle fonction f existe alors on dit que A se réduit à B et on note $A \leq_m^P B$.

Réduction polynomiales

Proposition:

Supponsons que $A \leq_m^P B$ et que $B \in \mathscr{C}$ avec $\mathscr{C} \in \{P, NP, coNP, EXP, NEXP, coNEXP\}$. Alors $A \in \mathscr{C}$.

Idée de la démonstration: Pour décider si $x \in A$, il suffit de calculer $f(x)$ (ce qui se fait en temps polynomial de manière déterministe), puis de lancer M_B qui reconnaît B sur $f(x)$.

Remarque:

Pour prouver que A est dans $\mathscr{C}$ (est "facile"), il suffit de donner un algorithme qui reconnaît A dans la bonne classe de complexité. On utilisera surtout les réductions dans l'autre sens: pour prouver que B est "difficile".

Réduction polynomiales

Proposition:

La relation $\leq_m^P$ est réflexive et transitive.

- Réflexivité: Soit A un langage, alors $A \leq_m^P A$ via l'identité.
- Transitivité: soit A, B et C des langages tels que $A \leq_m^P B$ via f et $B \leq_m^P C$ via g. Alors $A \leq_m^P C$ via $g \circ f$.

Table des matières

Difficulté et complétude

Définition (difficulté et complétude):

Soit $\leq$ une notion de réduction entre problèmes. Soit A un problème et $\mathscr{C}$ une classe de complexité.

- A est dit $\mathscr{C}$-difficile (ou $\mathscr{C}$-dur) pour les réductions $\leq$ si pour tout $B \in \mathscr{C}$, on a $B \leq A$.
- A est dit $\mathscr{C}$-complet pour les réductions $\leq$ s'il est $\mathscr{C}$-difficile pour les réductions $\leq$ et si $A \in \mathscr{C}$.

Cette notion dépend du type de réductions $\leq$. Par défaut, si l'on ne précise pas le type de réductions, il s'agira des réductions polynomiales $\leq_m^P$.

Difficulté et complétude

Remarque:

- La condition est très forte : il faut que tous les problèmes de $\mathscr{C}$ se réduisent à A. À priori, rien ne dit qu'il existe des problèmes $\mathscr{C}$-difficiles ni (a fortiori) $\mathscr{C}$-complets. Cependant, il existe de nombreux problèmes naturels complets pour les classes vues jusqu'à présent, notamment NP.
- Attention, si un langage A est hors de $\mathscr{C}$, cela n'implique pas qu'il est C-difficile ! Ce sera cela-dit le cas de tous les problèmes que nous verrons pendant ce cours.

Difficulté et complétude

Montrons que cette notion est inutile pour la classe P.

Proposition:

Tout problème B non trivial (c'est-à-dire $B \neq \emptyset$ et $B \neq \Sigma^*$) est P-difficile pour les réductions many-one en temps polynomial.

Proof.

Soit B non trivial, $x^0 \notin B$ et $x^1 \in B$. Soit $A \in P$: montrons que $A \leq_m^P B$. La réduction f de A vers B est alors définie comme suit :

$$f(x) = \begin{cases} x^1 \text{ si } x \in A \\ x^0 \text{ sinon} \end{cases}$$

Puisque $A \in P$, cette fonction f est calculable en temps polynomial. Par ailleurs, si $x \in A$ alors $f(x) = x^1 \in B$ et si $x \notin A$ alors $f(x) = x^0 \notin B$: ainsi, $x \in A$ ssi $f(x) \in B$, donc f est une réduction de A à B. □

Difficulté et complétude

Explication

Ce résultat vient du fait que la réduction est trop puissante pour la classe P : pour avoir une pertinence, il faudra donc réduire la puissance des réductions. Pour la classe P on utilise généralement les réductions en espace logarithmique. Il s'agit d'un phénomène général : plus la classe est petite, plus il faut utiliser des réductions faibles pour comparer les problèmes de la classe.

Table des matières

Difficulté et complétude

Définition: formule CNF (conjunctive normal form)

Une formule normale conjonctive (ou CNF) sur un ensemble de *variables* $\lambda = \{\lambda_1, \ldots, \lambda_n\}$ est une formule de la forme:

$$\phi = \mu_1 \wedge \mu_2 \wedge \cdots \wedge \mu_m.$$

Chaque formule μ_j (appelé *clause*) est elle même une disjonction de *littéraux*:

$$\mu_j = \mu_{j,1} \vee \mu_{j,2} \vee \cdots \vee \mu_{j,m_j}.$$

Chaque littéral $\mu_{j,k}$ correspond lui même à une variable λ_i et peut-être de polarité positive (on a alors $\mu_{j,k} = \lambda_i$) ou négative (on a alors $\mu_{j,k} = \overline{\lambda_i}$)

Difficulté et complétude

Définition:

Une *valuation* est une fonction $v : \lambda \to \{0, 1\}$ (ou $\{\bot, \top\}$).
Une formule CNF μ est satisfiable s'il existe une valuation de ses variables qui la rend vraie.

Exemples:

- La formule CNF $\mu = \lambda_1 \wedge \overline{\lambda_1}$ n'est pas satisfiable.
- Pour la formule CNF $\mu = (\lambda_1 \vee \lambda_2) \wedge (\overline{\lambda_1} \vee \overline{\lambda_2})$ est satisfiable:
  - les valuations v et v' avec $v(\lambda_1) = v(\lambda_2) = 1$ et $v'(\lambda_1) = v'(\lambda_2) = 0$ ne satisfont pas la formule.
  - les valuations u et u' avec $u(\lambda_1) = 1, u(\lambda_2) = 0$ et $u'(\lambda_1) = 0, u'(\lambda_2) = 1$ satisfont la formule.

Difficulté et complétude

Problème SAT:
Entrée: Une formule CNF ϕ.
Question: ϕ est-elle satisfiable?

Théorème de Cook

Le problème SAT est NP-complet.

Ce théorème a été démontré en 1971 par Stephen Cook.

Difficulté et complétude

Le problème SAT est dans NP car la machine de Turing non déterministe suivante le décide en temps polynomial :

- Lire l'expression booléenne ϕ.
- Pour toute variable λ_i apparaissant dans μ, choisir de manière non déterministe une valeur $v(\lambda_i)$ dans $\{\bot, \top\}$.
- Accepter si la valuation v satisfait ϕ et refuser sinon.

Difficulté et complétude

Idée de preuve que SAT est NP-dur:
Soit A un problème dans NP. Il existe donc une machine de Turing non déterministe M qui le décide en temps polynomial : pour toute instance x de A, x est une instance positive de A si et seulement s'il existe une exécution acceptante de M avec x sur le ruban d'entrée de longueur polynomiale en $|x|$. L'idée est donc de construire effectivement en temps polynomial une formule booléenne $\mu^{M,x}$ qui dit intuitivement " il existe une exécution acceptante de M avec x sur le ruban d'entrée de longueur polynomiale en $|x|$ ". Ainsi, x est une instance positive de A si et seulement si $\mu^{M,x}$ est satisfiable. Ainsi, on a une réduction polynomiale de A dans SAT : SAT est donc NP-dur.

Difficulté et complétude

Le théorème de Cook est la première preuve de NP-complétude. Les autres preuves de NP-complétude se font généralement par réduction polynomiale d'un problème déjà classé comme NP-complet. En effet, pour montrer la NP-difficulté d'un problème A, il suffit maintenant de prouver que SAT $\leq_m^P A$.

Table des matières

3-SAT

Une formule 3-CNF est une formule CNF ou chaque clause est composé de 3 littéral ou moins.
Exemple:

$$\mu = (\lambda_1 \vee \lambda_2 \vee \overline{\lambda_3}) \wedge (\overline{\lambda_1} \vee \lambda_2 \vee \overline{\lambda_3}) \wedge (\lambda_1 \vee \overline{\lambda_2} \vee \overline{\lambda_3}) \wedge (\lambda_3)$$