

Calculabilité et complexité

Cours n° 4

Nicolas (Miki) Hermann

LIX, École Polytechnique

`hermann@lix.polytechnique.fr`

Les classes P, NP, PSPACE, EXP, ... contiennent une infinité de langages.

Question : Comment classifier des langages dans une classe ?

Réponse : Par leur difficulté de solution.

Question : Comment dire que le langage/problème B est au moins aussi difficile que le langage/problème A ?

Réponse : Par le concept de **réduction**.

Réductions

A se réduit à B s'il existe une **transformation R** , telle que pour chaque entrée $x \in A$ elle calcule l'entrée équivalente $R(x) \in B$.
La transformation R ne peut pas être trop coûteuse !

A se réduit à B s'il existe une **transformation R** , telle que pour chaque entrée $x \in A$ elle calcule l'entrée équivalente $R(x) \in B$.
La transformation R ne peut pas être trop coûteuse !

Définition de la réduction de Karp (many-one)

Soit A et B deux langages/problèmes A est **polynomialement (logarithmiquement)** réductible à B s'il existe une fonction $R: \Sigma^* \rightarrow \Sigma^*$ calculable par une machine de Turing déterministe en **temps polynomial** (en **espace $\log n$**), telle que pour chaque mot x la condition suivante est satisfaite :

$$x \in A \text{ si et seulement si } R(x) \in B$$

R s'appelle une **réduction** polynomiale (logarithmique) de A à B

Les réductions se composent.

Théorème

Si R est la réduction de A à B et R' la réduction de B à C , alors la composition $R \cdot R'$ est une réduction de A à C .

Les réductions se composent.

Théorème

Si R est la réduction de A à B et R' la réduction de B à C , alors la composition $R \cdot R'$ est une réduction de A à C .

L'idée de la preuve :

$$\begin{aligned}x \in A &\equiv R(x) \in B \\y \in B &\equiv R'(y) \in C \\x \in A &\equiv R'(R(x)) \in C\end{aligned}$$

Il faut faire attention aux réductions logarithmiques. □

Convention : réduction = réduction polynomiale

Notation : $A \propto B$ ou $A \preceq B$ si A se réduit à B . Vous pouvez aussi voir $A \propto_m^p B$ ou $A \preceq_m^p B$ exprimant qu'il s'agit d'une réduction *many-one* polynomiale.

Complétude

Soit \mathcal{C} une classe de complexité. Le langage/problème A est dit **\mathcal{C} -difficile** si pour chaque langage/problème $B \in \mathcal{C}$, B se réduit à A .

Si le langage/problème A est \mathcal{C} -difficile et $A \in \mathcal{C}$, on dit que A est **\mathcal{C} -complet**.

Problèmes \mathcal{C} -complets = les problèmes les plus difficiles à résoudre dans la classe \mathcal{C}

On peut prendre pour \mathcal{C} les classes **NP**, **PSPACE**, **EXP**, ... et aussi **P** si on utilise les réductions logarithmiques.

Une classe \mathcal{C} est **close par réductions** si $A \preceq B$ et $B \in \mathcal{C}$ implique $A \in \mathcal{C}$.

Une classe \mathcal{C} est **close par réductions** si $A \preceq B$ et $B \in \mathcal{C}$ implique $A \in \mathcal{C}$.

Proposition

Les classes NP, coNP, PSPACE, EXP, ... sont closes par réductions.
Les classes P, L, NL sont closes par réductions logarithmiques.

Preuve :

Exercice de style.

Une classe \mathcal{C} est **close par réductions** si $A \preceq B$ et $B \in \mathcal{C}$ implique $A \in \mathcal{C}$.

Proposition

Les classes NP, coNP, PSPACE, EXP, ... sont closes par réductions.
Les classes P, L, NL sont closes par réductions logarithmiques.

Proposition

Si les classes \mathcal{C} et \mathcal{C}' sont closes par réductions et A est \mathcal{C} -complet ainsi que \mathcal{C}' -complet, alors $\mathcal{C} = \mathcal{C}'$.

Soit $R \subseteq \Sigma^* \times \Sigma^*$ une relation binaire sur les mots. La relation (prédicat) R est **polynomialement décidable** s'il existe une machine de Turing déterministe qui décide le langage $\{(x, y) \mid (x, y) \in R\}$.

Théorème

Soit $L \subseteq \Sigma^*$ un langage. Nous avons $L \in \text{NP}$ si et seulement si il existe une relation binaire R et un polynôme p où R est **polynomialement décidable** et pour chaque $(x, y) \in R$ nous avons $|y| \leq p(|x|)$, tel que $L = \{x \mid \exists y (x, y) \in R\}$.

Théorème

Soit $L \subseteq \Sigma^*$ un langage. Nous avons $L \in \text{NP}$ si et seulement si il existe une relation binaire R et un polynôme p où R est **polynomialement décidable** et pour chaque $(x, y) \in R$ nous avons $|y| \leq p(|x|)$, tel que $L = \{x \mid \exists y (x, y) \in R\}$.

Preuve :

Supposons qu'un tel R existe. Alors L est décidé par la machine non-déterministe M suivante. Sur le mot x la machine M **choisie** un mot y de longueur au plus $p(|x|)$ et utilise l'algorithme polynomial pour tester $(x, y) \in R$.

Preuve (cont)

Supposons que $L \in \text{NP}$, i.e., il existe une machine de Turing non-déterministe N qui décide L en temps $p(|x|)$. Soit R la relation suivante : $(x, y) \in R$ si et seulement si y est le codage d'un calcul acceptant de N sur x . Alors nous avons $|y| \leq p(|x|)$, R est décidable en temps polynomial et $L = \{x \mid \exists y (x, y) \in R\}$. □

Méthode de preuve qu'un problème A est NP-complet.

- 1 Prouver l'appartenance $A \in \text{NP}$ par la méthode de **choix** d'une solution parmi un **ensemble fini** de possibilités, suivi par un **test** en temps **polynomial** si le choix est vraiment une solution de l'instance du problème.
- 2 Prouver que A est NP-difficile par une **réduction** à partir d'un autre problème B connu d'être NP-complet.

Intermezzo : Forme normale conjonctive

Littéral : Une variable x ou sa négation $\neg x$

Clause : Disjonction de littéraux $(l_1 \vee \dots \vee l_k)$

Formule CNF : Conjonction de clauses $\varphi = c_1 \wedge \dots \wedge c_n$

Problème: SAT

Entrée: Une formule propositionnelle $\varphi(x_1, \dots, x_n)$ en forme normale conjonctive avec les variables $X = \{x_1, \dots, x_n\}$.

Question: La formule φ est-elle satisfaisable, existe-t-il un **valuation** $I: X \rightarrow \{0, 1\}$ telle que $I(\varphi) = 1$?

Théorème (Cook)

SAT est NP-complet.

Théorème (Cook)

SAT est NP-complet.

Idée de la preuve

SAT \in NP : Choix non-déterministe d'une valuation I et test en temps polynomial si elle satisfait φ .

Borne inférieure : Le travail d'une machine de Turing non-déterministe M travaillant en temps polynomial peut être décrit par une formule propositionnelle en CNF.

Preuve (cont 1)

Trois types de variables :

$Q(i, k)$: En temp i la machine M est dans l'état q_k

$H(i, j)$: En temps i la tête lit le symbole à la position j

$S(i, j, k)$: En temps i le contenu de la position j est le symbole s_k

Preuve (cont 2)

Six types de clauses : Pour chaque temps $i = 1, \dots, p(n)$,

- G_1 : M est exactement dans un **seule état**
- G_2 : M lit exactement une **seule position** du ruban
- G_3 : chaque position du ruban contient exactement un **seule symbol** de Γ
- G_4 : Au départ (temps $i = 0$) M se trouve dans la **configuration initiale**
- G_5 : En temps $p(n)$, M est arrivé dans l'état YES et a **accepté** le mot x
- G_6 : Pour $i < p(n)$, la configuration de M en temps $i + 1$ suit par la **transition** Δ à partir de la configuration i (non-déterminisme = **disjunction**) □

Problème: 3SAT

Entrée: Une formule propositionnelle $\varphi(x_1, \dots, x_n)$ en CNF avec au plus 3 littéraux par clause.

Question: La formule φ est-elle satisfaisable ?

Théorème

3SAT est NP-complet.

Théorème

3SAT est NP-complet.

Idée de la preuve

3SAT \in NP : 3SAT est un cas spécial de SAT.

Borne inférieure : Chaque clause $c = l_1 \vee \dots \vee l_k$ peut être transformée en

$$(x \vee l_3 \vee \dots \vee l_k) \wedge (x \equiv l_1 \vee l_2)$$

où x est une nouvelle variable. La formule $(x \equiv l_1 \vee l_2)$ est équivalente à

$$(\neg x \vee l_1) \wedge (\neg x \vee l_2) \wedge (\neg l_1 \vee \neg l_2 \vee x)$$



Problème: 2SAT

Entrée: Une formule propositionnelle $\varphi(x_1, \dots, x_n)$ en CNF avec au plus 2 littéraux par clause.

Question: La formule φ est-elle satisfaisable ?

Problèmes NP-complets ?

Théorème

2SAT est dans P.

Problèmes NP-complets ?

Théorème

2SAT est dans P.

Preuve

Pour chaque formule $\varphi(x_1, \dots, x_n)$ construire un graphe orienté G_φ avec $2n$ sommets où chaque sommet représente x_i ou $\neg x_i$. Pour chaque clause $c_j = (l_{j1} \vee l_{j2})$ construire deux arcs $\neg l_{j1} \rightarrow l_{j2}$ et $\neg l_{j2} \rightarrow l_{j1}$ qui représentent les implications équivalentes à c_j . La formule φ n'est pas satisfaisable si et seulement si il existe une variable x et un chemin $x \rightarrow^* \neg x$ et $\neg x \rightarrow^* x$ dans G_φ (REACHABILITY). □

Problèmes NP-complets ?

Théorème

2SAT est dans P.

Preuve

Pour chaque formule $\varphi(x_1, \dots, x_n)$ construire un graphe orienté G_φ avec $2n$ sommets où chaque sommet représente x_i ou $\neg x_i$. Pour chaque clause $c_j = (l_{j1} \vee l_{j2})$ construire deux arcs $\neg l_{j1} \rightarrow l_{j2}$ et $\neg l_{j2} \rightarrow l_{j1}$ qui représentent les implications équivalentes à c_j . La formule φ n'est pas satisfaisable si et seulement s'il existe une variable x et un chemin $x \rightarrow^* \neg x$ et $\neg x \rightarrow^* x$ dans G_φ (REACHABILITY). □

Corollaire

2SAT est dans NL.

Problème: MAX2SAT

Entrée: Une formule propositionnelle $\varphi(x_1, \dots, x_n)$ en CNF avec au plus 2 littéraux par clause et une borne $K \in \mathbb{N}$.

Question: Existe-t-il une valuation I qui satisfait au moins K clauses de φ ?

Théorème

MAX2SAT est NP-complet

Théorème

MAX2SAT est NP-complet

Preuve

MAX2SAT \in NP : choix et test

Borne inférieure : réduction à partir de 3SAT

Soit $c = (x \vee y \vee z)$ une clause de 3SAT. Prenons les clauses

$$\begin{aligned} & (x)(y)(z)(w) \\ & (\neg x \vee \neg y)(\neg y \vee \neg z)(\neg z \vee \neg x) \\ & (x \vee \neg w)(y \vee \neg w)(z \vee \neg w) \end{aligned}$$

Si c est satisfaisable, on peut satisfaire 7 clauses précédentes. Si c est insatisfaisable ($x = y = z = 0$), seulement 6 clauses sont satisfaites.

Pour $\varphi = c_1 \wedge \dots \wedge c_k$ on pose $K = 7k$. □

Problème: 1-IN-3SAT

Entrée: Une formule propositionnelle φ en CNF avec au plus 3 littéraux par clause.

Question: Existe-t-il une valuation qui dans chaque clause évalue un littéral à 1 et les deux autres à 0 ?

Théorème

1-IN-3SAT est NP-complet.

Théorème

1-IN-3SAT est NP-complet.

Preuve

1-IN-3SAT \in NP : choix et test.

Borne inférieure : réduction de 3SAT

La clause $c = (l_1 \vee l_2 \vee l_3)$ est remplacée par la formule

$$(l_1 \vee z_1 \vee z_2) \wedge (\neg l_2 \vee z_1 \vee z_3) \wedge (\neg l_3 \vee z_2 \vee z_4) \wedge (z_2 \vee z_3 \vee z_5)$$

où z_1, \dots, z_5 sont des nouvelles variables. □

Cette réduction est **parsimonieuse**.

Problème: NOT-ALL-EQUAL SAT (NAE SAT)

Entrée: Formule propositionnelle φ en CNF.

Question: Existe-t-il une valuation I qui évalue dans chaque clause au moins un littéral à 1 et au moins un littéral à 0 ?

Théorème

NAE SAT est NP-complet.

Théorème

NAE SAT est NP-complet.

Preuve

NAE SAT \in NP : choix et test.

Borne inférieure : réduction à partir de SAT.

Pour chaque clause $c = (l_1, \vee \dots \vee l_k)$ construire une nouvelle clause $c' = (l_1, \vee \dots \vee l_k \vee z)$ où z est une nouvelle variable.

Explorer la complémentarité : (b_1, \dots, b_n) est une valuation satisfaisante de la formule NAE SAT φ si et seulement si $(1 - b_1, \dots, 1 - b_n)$ est, elle aussi, une valuation satisfaisante. \square

Problème: NAE k SAT

Entrée: Formule propositionnelle φ en CNF avec au plus k littéraux par clause.

Question: Existe-t-il une valuation I qui évalue dans chaque clause au moins un littéral à 1 et au moins un littéral à 0 ?

Devoir maison :

- 1 Prouver que NAE 3SAT est NP-complet. Attention, la réduction précédente à partir de 3SAT est la preuve pour NAE 4SAT.
- 2 Prouver que NAE 2SAT est dans P.

Soit $G = (V, E)$ un graphe et $S \subseteq V$. L'ensemble S est un **ensemble indépendant** (**independent set**) si pour chaque pair de sommets $u, v \in V$ nous avons

$$(u \in S) \wedge (v \in S) \Rightarrow (u, v) \notin E$$

Problème: INDEPENDENT SET

Entrée: Graphe $G = (V, E)$ et une constante $K \in \mathbb{N}$.

Question: Existe-t-il un ensemble indépendant S en G avec $|S| \geq K$?

Théorème

INDEPENDENT SET est NP-complet.

Théorème

INDEPENDENT SET est NP-complet.

Preuve :

INDEPENDENT SET \in NP : choix et test.

Borne inférieure : réduction à partir de 1-IN-3SAT (3SAT).

Soit $\varphi = c_1 \wedge \dots \wedge c_k$ avec $c_i = l_1^i \vee l_2^i \vee l_3^i$. Nous allons construire un graphe $G_\varphi = (V, E)$ qui correspond à φ .

$$V = \{l_j^i \mid i = 1, \dots, k; j = 1, 2, 3\}$$

$$E = \{(l_1^i, l_2^i), (l_2^i, l_3^i), (l_3^i, l_1^i) \mid i = 1, \dots, k\} \quad \text{triangles}$$

$$\cup \{(l_i^a, l_j^b) \mid l_i^a = \neg l_j^b\} \quad \text{lianes}$$

et choisissons $K = k$. Utiliser le principe de Dirichlet. □

Problème: k -DEGREE INDEPENDENT SET

Entrée: Graphe $G = (V, E)$, où chaque sommet $v \in V$ a le degré $\deg(v) \leq k$, et une constante $K \in \mathbb{N}$.

Question: Existe-t-il un ensemble indépendant S en G avec $|S| \geq K$?

Devoir maison :

- 1 Prouver que 4-DEGREE INDEPENDENT SET est NP-complet.
- 2 Quid de 3-DEGREE INDEPENDENT SET ?
- 3 Quid de 2-DEGREE INDEPENDENT SET ?
- 4 Quid de INDEPENDENT SET pour un graphe biparti ?

Soit $G = (V, E)$ un graphe et $S \subseteq V$. L'ensemble S est une **clique** si pour chaque pair de sommets $u, v \in S$, $u \neq v$, nous avons $(u, v) \in E$.

Problème: CLIQUE

Entrée: Graphe $G = (V, E)$ et une constante $K \in \mathbb{N}$.

Question: Existe-t-il dans G une clique $S \subseteq V$ avec $|S| \geq K$?

Théorème

CLIQUE est NP-complet.

Théorème

CLIQUE est NP-complet.

Preuve :

CLIQUE est une reformulation de INDEPENDENT SET.

Soit $G = (V, E)$ un graphe, son **complément** est $\bar{G} = (V, V \times V \setminus E)$. Il existe une clique avec au moins K sommets dans G si et seulement si il existe un ensemble indépendant de cardinalité au moins K dans \bar{G} . □

Soit $G = (V, E)$ un graphe et $S \subseteq V$. L'ensemble S est un **recouvrement par sommets** (**vertex cover**) si pour chaque arête $(u, v) \in E$ nous avons $u \in S$ ou $v \in S$.

Problème: VERTEX COVER

Entrée: Graphe $G = (V, E)$ et un budget $B \in \mathbb{N}$.

Question: Existe-t-il dans G un recouvrement par sommets S avec $|S| \leq B$?

Théorème

VERTEX COVER est NP-complet.

Théorème

VERTEX COVER est NP-complet.

Idée de la preuve

VERTEX COVER est encore une reformulation de INDEPENDENT SET. L'ensemble S est un INDEPENDENT SET dans le graphe $G = (V, E)$ si et seulement si $V \setminus S$ est un VERTEX COVER de G . \square

Soit $G = (V, E)$ un graphe. Un **chemin Hamiltonien** dans G est un chemin qui visite chaque sommet V exactement une fois.

Problème: HAMILTONIAN

Entrée: Graphe $G = (V, E)$.

Question: Existe-t-il un chemin Hamiltonian dans G ?

Théorème

HAMILTONIAN est NP-complet.

Théorème

HAMILTONIAN est NP-complet.

Idée de la preuve

HAMILTONIAN \in NP : choix et test.

Borne inférieure : réduction à partir de 3SAT

Beaucoup de dessins avec trois gadgets :

- gadget de variables
- gadget de clauses
- gadget XOR

Tour regrouper par une grande clique. □

Soit $K_n = (V, V \times V)$ le graphe complet avec $|V| = n$ sommets. Chaque arrête $(u, v) \in V \times V$ sera étiquetée par une **distance** $d: V \times V \rightarrow \mathbb{N}$, où $d(u, v)$ est la distance entre les sommets u et v .

Problème du **commi voyageur (traveling salesperson)** : trouver un chemin le plus court qui parcourt tous les sommets

Problème: TSP(D)

Entrée: Graphe complet $K_n = (V, V \times V)$ avec $|V| = n$ et les distances $d: V \times V \rightarrow \mathbb{N}$, plus un budget $B \in \mathbb{N}$.

Question: Existe-t-il un chemin dans K_n qui parcourt tous les sommets et dont la longueur est inférieure ou égale à B ?

Théorème

TSP(D) est NP-complet.

Théorème

TSP(D) est NP-complet.

Preuve

TSP(D) \in NP : choix et test.

Borne inférieure : réduction à partir de HAMILTONIAN.

Compléter le graph $G = (V, E)$ en graphe complet K_n . Construire les distances

$$d(u, v) = \begin{cases} 1 & \text{si } (u, v) \in E \\ 2 & \text{si } (u, v) \notin E \end{cases}$$

Budget $B = n - 1$. □

Problème: k COLORING

Entrée: Graphe $G = (V, E)$ et k couleurs $\{0, \dots, k - 1\}$.

Question: Existe-t-il un bon coloriage du graphe G , i.e., existe-t-il une fonction $f: V \rightarrow \{0, \dots, k - 1\}$ telle que $(u, v) \in E$ implique $f(u) \neq f(v)$?

Théorème

3COLORING est NP-complet.

Théorème

3COLORING est NP-complet.

Preuve :

3COLORING \in NP : choix et test.

Borne inférieure : réduction à partir de 1-IN-3SAT (3SAT suffit).

Pour chaque formule $\varphi = c_1 \wedge \dots \wedge c_k$ où $c_i = l_1^i \vee l_2^i \vee l_3^i$ construire un graphe $G_\varphi = (V, E)$.

Sommet général : a

Chaque variable x : sommets x et $\neg x$

Chaque clause c_i : triangle $\{(l_1^i, l_2^i), (l_2^i, l_3^i), (l_3^i, l_1^i)\}$

Preuve (cont)

Pour $\varphi(x_1, \dots, x_n) = c_1 \wedge \dots \wedge c_k$:

$$V = \{a\} \cup \{x_i, \neg x_i \mid i = 1, \dots, n\} \cup \{l_i^j \mid i = 1, 2, 3; j = 1, \dots, k\}$$

$$E = \{(a, x_i), (a, \neg x_i), (x_i, \neg x_i) \mid i = 1, \dots, n\}$$

$$\cup \{(l_1^i, l_2^i), (l_2^i, l_3^i), (l_3^i, l_1^i) \mid i = 1, \dots, k\}$$

$$\cup \{(x_i, l_j^m) \mid \text{si } x_i = l_j^m\} \cup \{(\neg x_i, l_j^m) \mid \text{si } \neg x_i = l_j^m\}$$

Valeurs 0 et 2 = NO, valeur 1 = YES. □

Devoir maison :

- 1 Prouver que 2COLORING est dans P.