

NP-complétude

Définition et notation

Un problème de décision est un problème dont la réponse est OUI ou NON.
Pour un problème P, notons $O(P)$ l'ensemble des instances de P dont la réponse est OUI.

Propriété

Soient A et B deux problèmes de décision.
Si $A \leq B$, alors il existe une fonction f calculable en temps polynomial telle que $p \in O(A) \Leftrightarrow f(p) \in O(B)$

Définition

\mathcal{P} est l'ensemble de tous les problèmes de décision pouvant être résolu en temps polynomial.

Théorème 1

Soit A et B deux problèmes de décision tels que $A \leq^p B$ et $B \in \mathcal{P}$.
Alors $A \in \mathcal{P}$

Preuve

Soit $p_B(n)$ le temps nécessaire pour résoudre une instance de B de taille n, et soit d_B le degré du polynôme p_B .
Soit $p_A(n)$ le temps nécessaire pour résoudre une instance de A de taille n, en comptant chaque résolution de B comme une opération élémentaire. Soit d_A le degré du polynôme p_A .
On peut résoudre A en un temps $\in O((1+p_B(p_A(n)))p_A(n)) = O(n^{d_A+d_A d_B})$.

Définition

\mathcal{NP} est l'ensemble de tous les problèmes de décision P tel que pour toute instance $p \in O(P)$, il existe une preuve vérifiable en temps polynomial que $p \in O(P)$.

Définition

Soit P un problème de décision. Le problème complémentaire de P, noté \bar{P} est le problème de décision tel que $p \in O(P) \Leftrightarrow p \notin O(\bar{P})$

Définition

$\text{co-}\mathcal{NP}$ est l'ensemble de tous les problèmes de décision P tel que $\bar{P} \in \mathcal{NP}$

Exemple

$\text{HAMD} \in \mathcal{NP}$ car si un graphe G a un cycle hamiltonien, on peut le prouver en exhibant un tel cycle. Pour vérifier la preuve, il suffit de s'assurer que le cycle passe bien une et une seule fois par chaque sommet du graphe, en empruntant uniquement des arêtes de G.

Remarque

La définition de \mathcal{NP} est asymétrique. On ne demande rien pour les instances $p \notin O(P)$. Remarquons ainsi par exemple qu'il n'est pas évident comment on peut fournir une preuve vérifiable en temps polynomial qu'un graphe n'est pas hamiltonien.

Ainsi, on ne sait pas si $\overline{\text{HAMD}} \in \mathcal{NP}$. Par contre, par définition, on a $\overline{\text{HAMD}} \in \text{co-}\mathcal{NP}$

Théorème 2 $\mathcal{P} \subseteq \mathcal{NP}$ et $\mathcal{P} \subseteq \text{co-}\mathcal{NP}$

Preuve

Soit $P \in \mathcal{P}$ et soit A un algorithme polynomial permettant de résoudre chaque instance de P. Pour une instance $p \in O(P)$, on peut vérifier que $p \in O(P)$ en appliquant A, qui donne la réponse OUI en un temps polynomial. Ceci prouve que $\mathcal{P} \subseteq \mathcal{NP}$. La preuve que $\mathcal{P} \subseteq \text{co-}\mathcal{NP}$ est similaire.

Question $\mathcal{P} = \mathcal{NP}?$

Beaucoup pensent qu'il est peu probable que $\mathcal{P} = \mathcal{NP}$, mais personne n'a démontré le contraire.

Définition

\mathcal{NP} -complet est l'ensemble de tous les problèmes de décision P tel que

- $P \in \mathcal{NP}$

- $P' \leq P \quad \forall P' \in \mathcal{NP}$

Théorème 3

Soient A et B deux problèmes de décision tels que $A \in \mathcal{NP}$ -complet, $B \in \mathcal{NP}$ et $A \leq B$.

Alors $B \in \mathcal{NP}$ -complet

Preuve

Pour tout $P \in \mathcal{NP}$, on a $P \leq A \leq B$, ce qui implique $P \leq B$.

Du théorème 3, on déduit que pour montrer qu'un problème P est \mathcal{NP} -complet, il faut d'abord s'assurer que $P \in \mathcal{NP}$, et il suffit ensuite de déterminer un problème $P' \in \mathcal{NP}$ -complet tel que $P' \leq P$. En d'autres termes, si P était facile à résoudre, P' le serait également car on est capable de transformer polynomialement P' en P.

Théorème 4 $(\mathcal{NP}\text{-complet} \cap \mathcal{P} \neq \emptyset) \Leftrightarrow (\mathcal{P} = \mathcal{NP})$ **Preuve**

(\Leftarrow) Supposons $\mathcal{P} = \mathcal{NP}$. On a alors $\mathcal{P} = \mathcal{NP}$ -complet car \mathcal{NP} -complet $\subseteq \mathcal{NP} = \mathcal{P}$ et si $P \in \mathcal{P}$, on a $P \in \mathcal{NP}$ et $P' \leq P$ pour tout $P' \in \mathcal{NP} = \mathcal{P}$ (puisque l'on peut résoudre P' en faisant un nombre polynomial d'opérations élémentaires + un appel inutile à la boîte noire qui résout P). Comme $\mathcal{P} \neq \emptyset$, on a donc $\mathcal{NP}\text{-complet} \cap \mathcal{P} = \mathcal{P} \neq \emptyset$.

(\Rightarrow) On sait déjà que $\mathcal{P} \subseteq \mathcal{NP}$. Soit $A \in \mathcal{NP}$. Il suffit de montrer que $A \in \mathcal{P}$.

Choisissons B dans $\mathcal{NP}\text{-complet} \cap \mathcal{P}$. On a alors $A \leq B$ car B est \mathcal{NP} -complet. Mais on sait par le théorème 1 que $A \leq B$ et $B \in \mathcal{P}$ implique $A \in \mathcal{P}$.

Définitions

Une formule booléenne est dite satisfaisable s'il existe au moins une affectation de valeurs à ses variables de telle sorte que cette formule soit vraie. On note **SAT** le problème consistant à déterminer si une formule booléenne donnée est satisfaisable.

Exemple

$(p \vee q) \wedge \neg(p \wedge q)$ est une formule booléenne satisfaisable car elle est vraie si on donne la valeur VRAI à p et FAUX à q. Par contre $(\neg p) \wedge (p \vee q) \wedge (\neg q)$ n'est pas satisfaisable.

Définitions

Un **littéral** est une variable booléenne ou sa négation. Une **clause** est un littéral ou une disjonction de littéraux. Une formule booléenne est sous **forme normale conjonctive** si elle est une clause ou une conjonction de clauses. Le problème **SAT-CNF** est la restriction de SAT aux formules booléennes données sous forme normale conjonctive. Le problème **SAT-k-CNF** est la restriction de SAT-CNF au cas où toutes les clauses ont au plus k littéraux.

Théorème 5 (Cook, 1971)

$\text{SAT-CNF} \in \mathcal{NP}\text{-complet}$

Cook a en fait réussi à démontrer que $\mathcal{NP}\text{-complet} \neq \emptyset$. En utilisant le théorème 3, il devenait alors plus facile de déterminer d'autres problèmes \mathcal{NP} -complets. On en connaît aujourd'hui plusieurs milliers. Voici quelques exemples.

Théorème 6

$\text{SAT} \in \mathcal{NP}\text{-complet}$

Preuve

Il est facile de vérifier que $\text{SAT} \in \mathcal{NP}$. La propriété découle alors du fait que $\text{SAT-CNF} \leq \text{SAT}$.

Théorème 7SAT-3-CNF $\in \mathcal{NP}$ -complet**Preuve**

Il est facile de vérifier que SAT-3-CNF $\in \mathcal{NP}$. On va montrer que SAT-CNF \approx SAT-3-CNF.

Soit f une formule booléenne sous forme normale conjonctive. On a donc $f = C_1 \wedge C_2 \wedge \dots \wedge C_m$, où chaque C_i est une disjonction de littéraux. Chaque C_i peut être remplacé par une conjonction de clauses à au plus 3 littéraux de la manière suivante. Soit $C_i = \ell_1 \vee \dots \vee \ell_r$:

- si $r \leq 3$ alors posons $C'_i = C_i$
- sinon, posons $C'_i = (\ell_1 \vee \ell_2 \vee x_1) \wedge (\neg x_1 \vee \ell_3 \vee x_2) \wedge \dots \wedge (\neg x_{r-3} \vee \ell_{r-1} \vee \ell_r)$

Soit $f' = C'_1 \wedge C'_2 \wedge \dots \wedge C'_m$. On va vérifier que f est satisfaisable si et seulement si f' l'est.

Si f est satisfaisable, considérons une affectation de valeurs aux variables qui rendent chaque C_i vrai, et soit $C_i = \ell_1 \vee \dots \vee \ell_r$ une clause quelconque de f . Si $r \leq 3$ alors C'_i est également vraie. Sinon, il existe un indice s tel que ℓ_s est vrai., et en posant $x_1 = x_2 = \dots = x_{s-2} = \text{vrai}$ et $x_{s-1} = x_s = \dots = x_{r-3} = \text{faux}$, et en gardant les mêmes valeurs pour ℓ_1, \dots, ℓ_r on a que C'_i est également vraie. Il est donc également possible de satisfaire f' .

Si f' est satisfaisable, considérons une affectation de valeurs aux variables qui rendent chaque C'_i vrai, et soit $C'_i = (\ell_1 \vee \ell_2 \vee x_1) \wedge (\neg x_1 \vee \ell_3 \vee x_2) \wedge \dots \wedge (\neg x_{r-3} \vee \ell_{r-1} \vee \ell_r)$ une clause quelconque de f' . Il est facile de vérifier que si $\ell_1 = \dots = \ell_{r-2} = \text{faux}$ alors $x_1 = \dots = x_{r-3} = \text{vrai}$, ce qui implique $\ell_{r-1} = \text{vrai}$ ou $\ell_r = \text{vrai}$. C_i est donc également vraie.

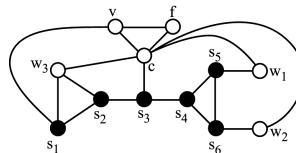
Définition

k-COL est le problème de décision consistant à déterminer s'il est possible de colorer les sommets d'un graphe en utilisant au plus k couleurs, de telle sorte qu'aucune arête n'ait ses deux extrémités de même couleur.

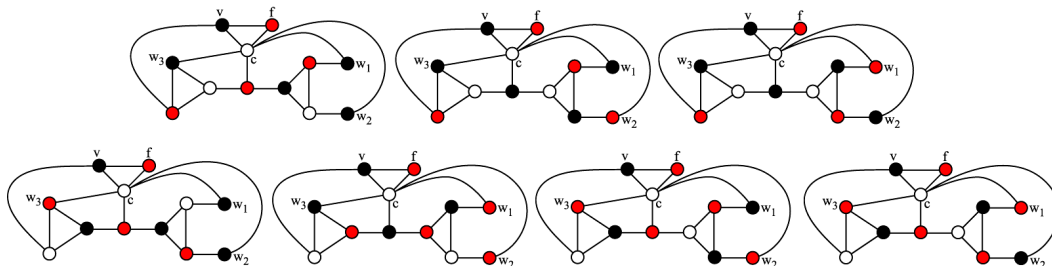
Théorème 8k-COL $\in \mathcal{NP}$ -complet pour tout $k \geq 3$.**Preuve**

Il est facile de vérifier que k-COL $\in \mathcal{NP}$. Comme on a trivialement 3-COL \approx k-COL, nous allons démontrer SAT-3-CNF \approx 3-COL.

Soit $f = C_1 \wedge C_2 \wedge \dots \wedge C_m$ une formule booléenne sous forme normale conjonctive, dans laquelle chaque clause a au plus 3 littéraux. Sans perte de généralité, on peut supposer que chaque clause a exactement 3 littéraux car $\ell = \ell \vee \ell \vee \ell$, et $\ell \vee \ell' = \ell \vee \ell \vee \ell'$. Soit v_1, \dots, v_k les variables booléennes apparaissant dans f . Construisons un graphe G_f comme suit. On crée 3 sommets notés v, f, c (pour vrai, faux, contrôle) qu'on relie 2 à 2. On crée ensuite 2 sommets x_i et y_i par variable v_i , et on relie chaque x_i au y_i et chaque x_i et chaque y_i au sommet c . Finalement, pour chaque clause $C_i = \ell_1 \vee \ell_2 \vee \ell_3$ on crée 6 sommets supplémentaires s_1, \dots, s_6 reliés comme représenté ci-dessous. Le sommet w_t ($t=1,2,3$) est égal à x_u si $\ell_t = v_u$ et à y_u si $\ell_t = \neg v_u$.



Si on veut colorer G_f en 3 couleurs, au moins l'un des 3 sommets w_1, w_2 et w_3 doit avoir la même couleur que v . En effet, étant donné que w_1, w_2 et w_3 sont tous les 3 reliés à c , ils ne peuvent être colorés que comme v ou f . Si w_1 et w_2 ont la même couleur que f , alors s_4 aussi; s_3 a alors la même couleur que v ; donc ni s_1 ni s_2 n'a la même couleur que v , ce qui oblige w_3 à avoir la même couleur que v . Toutes les 7 possibilités de colorer w_1, w_2 et w_3 avec les couleurs de v et f sont représentées ci-dessous.



Ainsi, G_f est colorable en 3 couleurs si et seulement si f est satisfaisable

Définition

STABLE est le problème de décision consistant à déterminer si un graphe donné contient k sommets 2 à 2 non-adjacents.

Théorème 9

STABLE \in \mathcal{NP} -complet

Il est facile de vérifier que STABLE \in \mathcal{NP} . On va donner 2 démonstrations que STABLE \in \mathcal{NP} -complet. En fait, nous allons choisir un problème \mathcal{NP} -complet P et montrer que P \approx STABLE. Dans la 1^{ère} démonstration, nous choisirons P=k-COL alors que dans la 2^{ème}, nous choisirons P=SAT-CNF.

Montrons que k-COL \approx STABLE. Soit $G=(V,E)$ un graphe. Construisons un graphe H_G en faisant k copies de G et en reliant chaque sommet à chacune de ses copies. Il est facile de vérifier que G est colorable en au plus k couleurs si et seulement si il existe dans H_G un ensemble de |V| sommets 2 à 2 non-adjacents.

Montrons que SAT-CNF \approx STABLE. Soit $f=C_1 \wedge C_2 \wedge \dots \wedge C_m$ une formule booléenne sous forme normale conjonctive. Construisons un graphe $G_f=(V,E)$ avec $V=\{(\ell,i) \text{ tel que } \ell \text{ est un littéral de } C_i\}$ et en reliant deux sommets (ℓ,i) et (ℓ',j) par une arête si et seulement si $i=j$ ou $\ell=\neg\ell'$. Il est alors facile de vérifier que f est satisfaisable si et seulement si il existe dans G_f un ensemble de m sommets 2 à 2 non-adjacents.

Définition

Soient n entiers v_1, \dots, v_n . Le problème **PARTITION** consiste à déterminer s'il existe un sous-ensemble S de $\{1, \dots, n\}$ tel que $\sum_{i \in S} v_i = \sum_{i \notin S} v_i$

Théorème 10

PARTITION \in \mathcal{NP} -complet

Définition

Soient n entiers v_1, \dots, v_n , ainsi que 2 entiers k et L. Le problème **k^{ème} Plus Lourd Sous-Ensemble (KPLS)** consiste à déterminer s'il existe k sous-ensembles S_1, \dots, S_k de $\{1, \dots, n\}$ tel que $\sum_{i \in S_j} v_i \leq L$ ($j=1, \dots, k$).

Théorème 11

PARTITION \approx KPLS

Preuve

Soient n entiers v_1, \dots, v_n . Définissons $L = \frac{1}{2} \sum_{i=1}^n v_i$ et $k=2^{n-1} + 1$

Soit $A_<$ (resp. $A_ =$ et $A_>$) l'ensemble des sous-ensembles S de $\{1, \dots, n\}$ tel que $\sum_{i \in S} v_i < L$ (resp. $=L$ et $>L$).

On a $|A_<| + |A_ =| + |A_>| = 2^n$.

De plus, $|A_<| = |A_>|$ car $S \in A_< \Leftrightarrow (\{1, \dots, n\} - S) \in A_>$.

On déduit que $2|A_<| + |A_ =| = 2^n \Rightarrow 2(|A_<| + |A_ =|) = |A_ =| + 2^n$.

En conclusion, on a les équivalences suivantes :

$$\exists S \subset \{1, \dots, n\} \text{ tel que } \sum_{i \in S} v_i = \sum_{i \notin S} v_i$$

$$\Leftrightarrow A_ = \neq \emptyset \Leftrightarrow 2(|A_<| + |A_ =|) > 2^n \Leftrightarrow |A_<| + |A_ =| > 2^{n-1} \Leftrightarrow |A_<| + |A_ =| \geq k$$

$$\Leftrightarrow \text{il existe } k \text{ sous-ensembles } S_1, \dots, S_k \text{ de } \{1, \dots, n\} \text{ tel que } \sum_{i \in S_j} v_i \leq L \quad (j=1, \dots, k).$$

Remarque

Nul ne sait si KPLS \in \mathcal{NP} . On ne peut donc pas déduire des théorèmes 10 et 11 que KPLS \in \mathcal{NP} -complet.

Définition

\mathcal{NP} -dur est l'ensemble de tous les problèmes P tel qu'il existe $P' \in \mathcal{NP}$ -complet avec $P' \leq^p P$

Remarque

Un problème \mathcal{NP} -dur n'est pas forcément un problème de décision, ni un problème de \mathcal{NP}

Exemple

Soit COLO le problème consistant à déterminer le plus petit nombre de couleurs nécessaires pour colorer les sommets d'un graphe de telle sorte qu'aucune arête n'ait ses deux extrémités de la même couleur.

Soit COLC le problème consistant à exhiber une telle coloration minimale.

On a bien évidemment $3\text{-COL} \leq^p \text{COLO} \leq^p \text{COLC}$.

\Rightarrow COLO et COLC sont \mathcal{NP} -durs alors que ce ne sont pas des problèmes de décision.

Soit maintenant COLE le problème de décision consistant à déterminer si étant donné un graphe G et un nombre k , il est vrai que G peut être coloré avec k couleurs, mais pas avec $k-1$.

On a $3\text{-COL} \leq^p \text{COLE}$ puisque la réponse à 3-COL est OUI si et seulement si elle est OUI pour COLE avec $k=1, 2$ ou 3 .

\Rightarrow COLE est \mathcal{NP} -dur alors qu'on ne sait pas si $\text{COLE} \in \mathcal{NP}$ car il n'est pas facile de donner une preuve vérifiable en temps polynomial qu'un graphe G n'est pas colorable en $k-1$ couleurs.

Autres classes de complexité

PSPACE est l'ensemble des problèmes de décision qui peuvent être résolus à l'aide d'un algorithme qui requiert une place mémoire de taille $\in O(p(n))$ pour chaque instance de taille n .

LogSPACE est l'ensemble des problèmes de décision qui peuvent être résolus à l'aide d'un algorithme qui requiert une place mémoire de taille $\in O(\log n)$ pour chaque instance de taille n . Pour que cette définition ait du sens, on suppose que la donnée est accessible en lecture seulement, et l'algorithme n'utilisera que $O(\log n)$ bits supplémentaires pour stocker d'autres informations.

On sait que $\text{LogSPACE} \subset \text{PSPACE}$.

On sait aussi que $\text{LogSPACE} \subseteq \mathcal{P} \subseteq \mathcal{NP} \subseteq \text{PSPACE}$. Et donc au moins l'une de ces inclusions est stricte.

En ce qui concerne $\text{co-}\mathcal{NP}$, personne ne sait si $\mathcal{NP} \neq \text{co-}\mathcal{NP}$.

Par contre, il a été démontré que $(\mathcal{NP}\text{-complet} \cap \text{co-}\mathcal{NP} \neq \emptyset) \Leftrightarrow (\mathcal{NP} = \text{co-}\mathcal{NP})$.

La plupart des problèmes connus comme appartenant à la fois à \mathcal{NP} et à $\text{co-}\mathcal{NP}$ sont aussi dans \mathcal{P} . Parmi les problèmes qui sont dans $\mathcal{NP} \cap \text{co-}\mathcal{NP}$ et qu'on a longtemps soupçonné ne pas être dans \mathcal{P} , citons le problème suivant :

Soit n un entier : n est-il un nombre premier ?

Il est facile de donner un certificat vérifiable en temps $\in O(p(\log n))$ que n n'est pas premier. Il est moins facile, mais possible également, de donner un certificat vérifiable en temps $\in O(p(\log n))$ que n est premier. Cependant, ce n'est qu'en août 2002 que Manindra Agrawal, Neeraj Kayal et Nitin Saxena, de l'Indian Institute of Technology, ont réussi à démontrer que ce problème est dans \mathcal{P} .

La situation la plus probable est la suivante :

