Université Paris Cité

Introduction à la théorie de complexité

CM nº11 — Algorithmique (AL5)

Matěj Stehlík

8/12/2023

Introduction à la théorie de la complexité

- Let but est de classifier les problèmes entre ceux qui sont « durs » et ceux qui sont « faciles », où « facile » veut dire « il existe un algorithme pour les résoudre en temps polynomial ».
- Théorie de la complexité : domaine théorique qui s'intéresse à classifier les problèmes selon leurs difficulté (il y a de nombreux degrés de difficulté).

Problème de décision vs problème d'optimisation

Problème de décision

Instance : un graphe $G = (V, E)$

Question : G est-il biparti ?

Réponses possibles : oui ou non

Problème d'optimisation

Instance : un graphe $G = (V, E)$

Question : Quelle est la taille de la plus grande clique dans G ?

Réponses possibles : n'importe quel entier entre 0 et n, où n est le nombre de sommets.

Transformation en problème de décision

Définition

- Un problème de décision est un problème dont la réponse est oui ou non.
- Un problème d'optimisation est un problème où l'on cherche à minimiser ou maximiser une certaine quantité.

- En théorie de la complexité, on transforme toujours un problème d'optimisation en problème de décision (plus facile ou de difficulté égale), la plupart du temps en ajoutant un "seuil" et en se demandant si la valeur optimale est :
  - supérieure ou égale au seuil (pour un problème de maximisation) ou
  - inférieure ou égale au seuil (pour un problème de minimisation).

Exemple

Problème d'optimisation

Instance : un graphe $G = (V, E)$

Question : Quelle est la taille de la plus grande clique dans G ?

Réponses possibles : n'importe quel entier entre 0 et n, où n est le nombre de sommets.

Problème de décision

Instance : un graphe $G = (V, E)$

Question : G admet-il une clique de taille au moins k ?

Réponses possibles : oui ou non

Taille de l'instance : $N + \log k$, où N est la taille de stockage du graphe

Remarque sur la taille des instances

- Il faut compter le nombre N de bits nécessaires pour stocker l'entrée.
- Pour un graphe (V, E) à n sommets, il faut $O(n \log n)$ bits pour décrire $V = \{1, \ldots, n\}$.
- Pour décrire E par une matrice d'adjacence, il faut $O(n^2)$ bits
- Pour décrire E par une liste d'adjacence, il faut $2|E| \log n \leq 2n^2 \log n$ bits.
- Dans tous les cas, pour un graphe à n sommets, $N = \text{poly}(n)$.
- Du coup, dire « polynomial en N » ou « polynomial en n » revient au même (car la composée de deux polynômes est un polynôme).
- Par simplicité, on comptera désormais la complexité d'un problème de graphes en fonction du nombre n de sommets.

Algorithmes de complexité polynomiale

Question

Existe-t-il un algorithme de complexité *polynomiale* pour résoudre le problème sur une instance de taille N, le nombre d'opérations dans le pire cas est poly(N) ? (Par exemple, $O(N^2), O(N^{32}), O(\sqrt{N}), \ldots$)

Remarque

On n'a pas de définition exacte de « algorithme » ni de « opération élémentaire » (pour calculer la complexité). Il existe une vraie definition des deux (voir machines de Turing).

La classe $\mathcal{P}$

Définition

Un problème est dans $\mathcal{P}$ (pour *polynomial*) s'il existe un algorithme répondant correctement oui ou non sur n'importe quelle instance, et dont la complexité dans le pire cas est polynomiale en N (taille de l'instance).

Exemples de problèmes dans $\mathcal{P}$

Instance : un graphe $G = (V, E)$.

Question : G est-il biparti ?

Question : G admet-il un arbre couvrant ?

Question : G admet-il un cycle eulérien ?

Algorithmes vérifieurs

Définition

Un *algorithme vérifieur* pour un problème A est un algorithme qui prend en entrée une instance du problème A (par exemple, un graphe) et une information supplémentaire appelée « certificat » (ou, informellement, indice), et qui répond « oui » ou « l'indice ne permet pas de décider ». Un indice pour lequel l'algorithme répond « oui » est un *certificat*.

Retour a l'exemple

Problème de décision

Entrée : Graphe G et entier k

Question : Existe-t-il une clique de taille au moins k ?

Indice : Ensemble W de sommets de G

- L'algorithme vérifie que pour tout $u, v \in W$, uv est bien une arête (donc W est une clique) puis que $|W| \geq k$.
- Si ces conditions sont vérifiées, l'algorithme vérifieur répond « Oui, G admet une clique de taille au moins k ».
- Sinon, il répond « L'indice ne permet pas de décider ».

La classe $\mathcal{NP}$

Définition

Le problème A est dans $\mathcal{NP}$ (pour *non-deterministic polynomial*) s'il existe un algorithme vérifieur pour A vérifiant les conditions suivantes :

- l'algorithme vérifieur fonctionne en temps polynomial en la taille de l'entrée
- si la véritable réponse au problème est « oui », alors il existe un indice de taille polynomial sur lequel l'algorithme vérifieur répond « oui » (un certificat polynomial)
- si la veritable réponse est non, alors il n'existe aucun indice sur lequel l'algorithme vérifieur répond « oui » (pas de certificat).

La classe $\mathcal{NP}$ contient la classe $\mathcal{P}$

- Si un problème A est dans $\mathcal{P}$, alors is est dans $\mathcal{NP}$.
- En effet, l'algorithme vérifieur de A est tout simplement l'algorithme polynomial pour A, et le certificat est vide.
- En particulier, arbre couvrant de poids minimum, biparti, degré max sont tous dans $\mathcal{NP}$.

Quelques problèmes dans $\mathcal{NP}$ (1/2)

STABLE MAX

Entrée : Graphe G et entier k

Question : Existe-t-il un stable de taille au moins k ?

Indice : Ensemble W de sommets de G

k-COLOR

Entrée : Graphe G et entier k

Question : Le graphe G est-il k-coloriable ?

Indice : une affectation de couleurs $c : V \to \{1, \ldots, k\}$

- L'algorithme vérifieur vérifie que $c(u) \neq c(v)$ pour toute arête $uv \in E(G)$.
- Si tout vérifie, l'algorithme vérifieur répond « oui ».
- Il existe bien un certificat ssi le graphe est k-coloriable.

Quelques problèmes dans $\mathcal{NP}$ (2/2)

VERTEX COVER

Instance : Graphe G et entier k

Question : Existe-t-il un transversal de taille au plus k ?

= vertex cover ou hitting set

TSP

Instance : Graphe complet G, poids sur les arêtes, nombre k.

Question : Existe-t-il un cycle hamiltonien de poids au plus k ?

Indice : Un ordre $v_1, v_2, \ldots, v_n$ sur les sommets.

La question à 1 million

Question

Existe-t-il des problèmes qui sont dans $\mathcal{NP}$ mais pas dans $\mathcal{P}$?

Conjecture

$\mathcal{P} \neq \mathcal{NP}$

- La conjecture semble naturelle, car il est plus facile d'écrire un algorithme vérifieur plutôt qu'un algorithme qui décide oui ou non sur toute instance.
- Le Clay Mathematical Institute offre 1 million de dollars pour une preuve ou contre-exemple.

Réduction et définition de $\mathcal{NP}$-difficile (1/3)

Problème A

Instance Un entier x

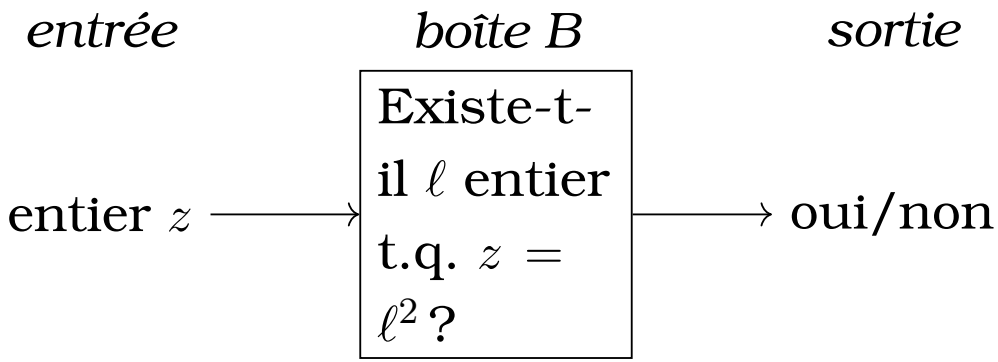
Question Existe-t-il y entier tel que $x = y^2 + 1$?

Problème B

Instance Un entier z

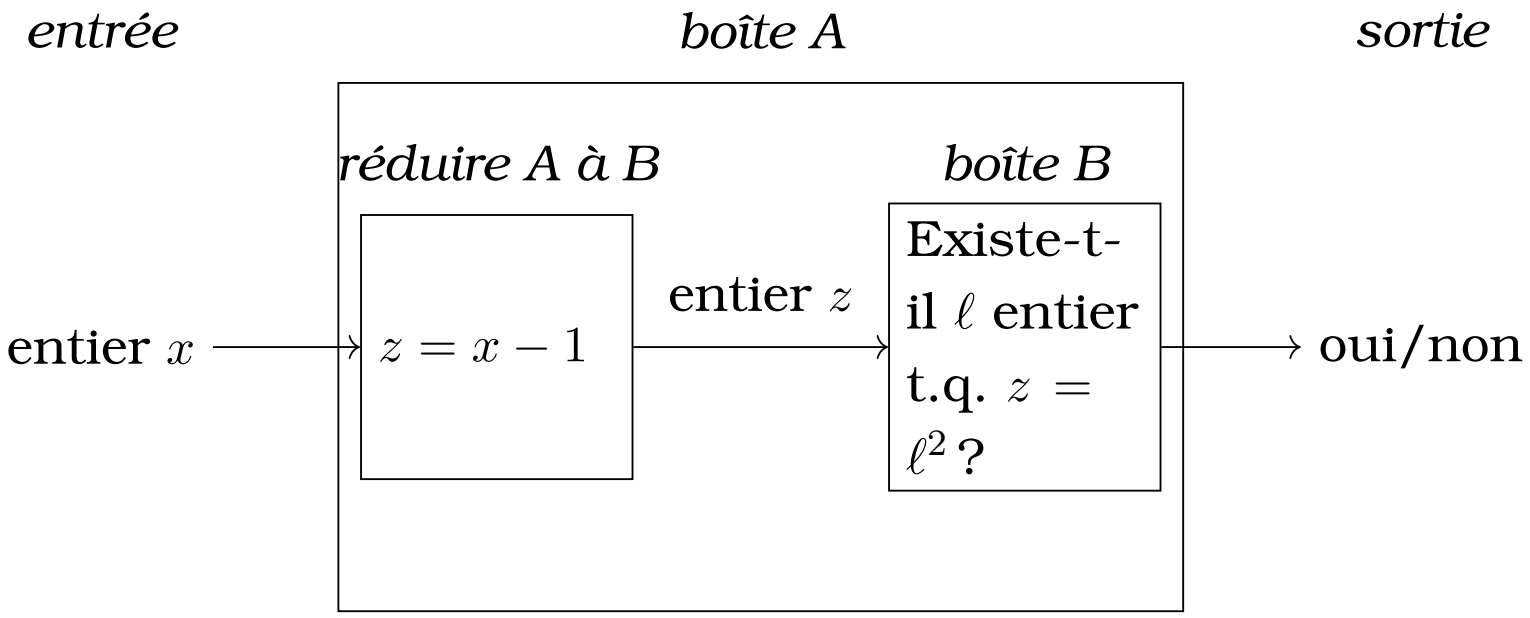
Question Existe-t-il ℓ entier tel que $z = \ell^2$?

Imaginons qu'il existe une boîte noire qui répond correctement au problème B :



Réduction et définition de $\mathcal{NP}$-difficile (2/3)

En utilisant cette boîte, on peut construire une autre boîte pour résoudre le problème A :



Donc, le problème B est aussi dur que le problème A (voire plus).

Réduction et définition de $\mathcal{NP}$-difficile (3/3)

- Soient P_1 et P_2 deux problèmes de décision.

Définition

Un algorithme $\mathcal{A}$ *réduit* P_1 *à* P_2 si, en prenant en entrée une instance I_1 de P_1, l'algorithme construit une instance I_2 de P_2 *équivalente* : c'est-à-dire, la vraie réponse à I_1 sur P_1 est OUI *si et seulement si* la (vraie) réponse à I_2 sur P_2 est OUI.

Définition

Un problème P_1 est $\mathcal{NP}$*-difficile* si n'importe quel problème P_2 appartenant à $\mathcal{NP}$ peut se réduire polynomialement à P_1.

- Autrement dit, si je sais résoudre P_1 qui est $\mathcal{NP}$-difficile, alors je sais résoudre *tous* les problèmes dans $\mathcal{NP}$.

La classe $\mathcal{NP}$-complet

Définition

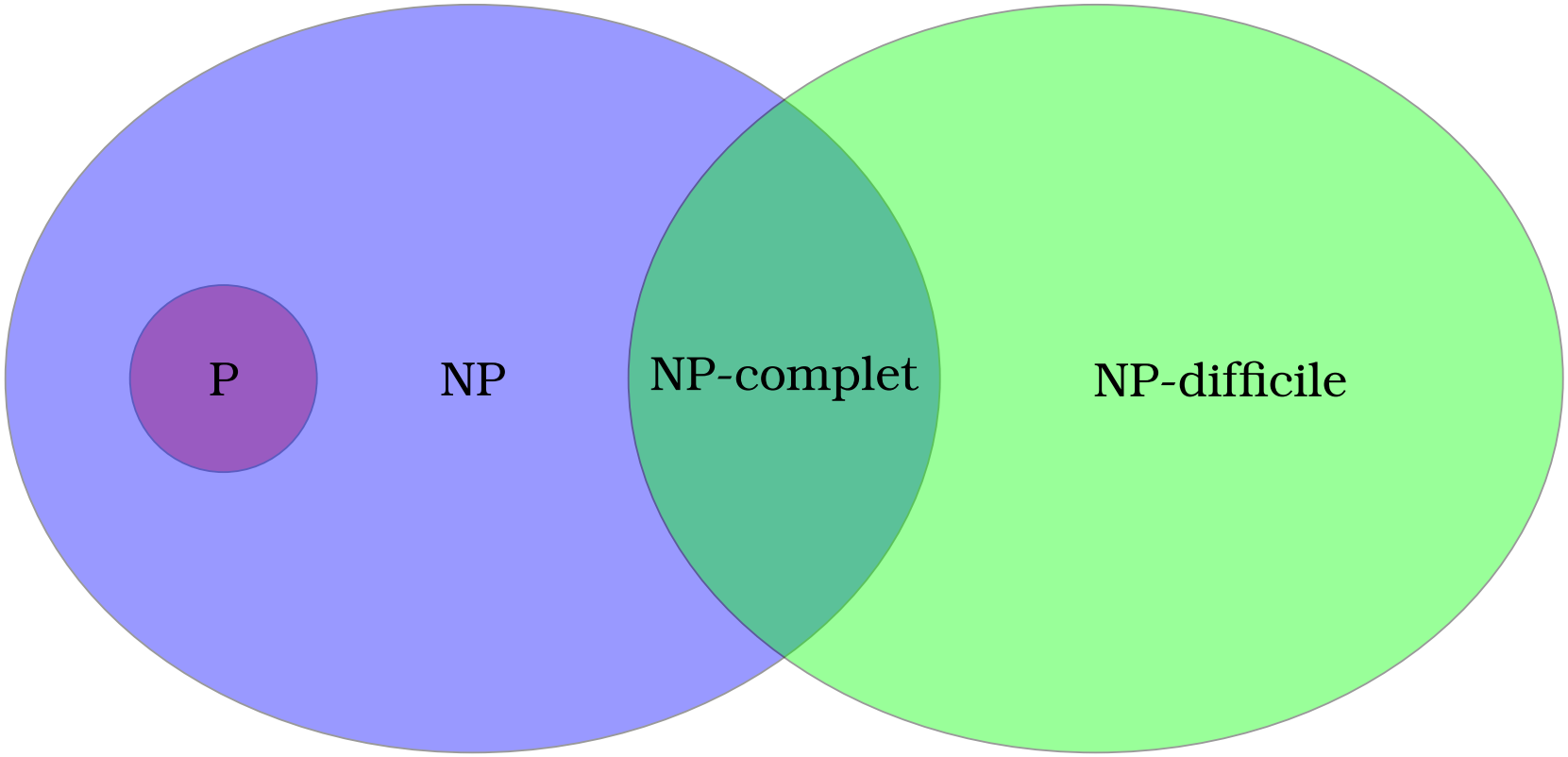
Un problème est *$\mathcal{NP}$-complet* si il est à la fois dans $\mathcal{NP}$ (il existe un algorithme vérifieur) et $\mathcal{NP}$-difficile.

- Si l'on dispose d'un premier problème de référence $\mathcal{NP}$-difficile P_{ref}, on peut prouver qu'un autre problème P_1 est $\mathcal{NP}$-difficile en réduisant P_{ref} à P_1.
- N'importe quel problème $P_2 \in \mathcal{NP}$ se réduit à P_{ref} qui lui-même se réduit à P_1.
- Donc, P_2 se réduit à P_1.
- Autrement dit, si je sais résoudre P_1, alors je sais résoudre P_{ref}, qui est $\mathcal{NP}$-difficile, donc je sais résoudre n'importe quel problème dans $\mathcal{NP}$.

Synthèse des classes de complexité $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{NP}$-difficile et $\mathcal{NP}$-complet

- Problèmes dans $\mathcal{P}$: il existe un algorithme polynomial (faciles à résoudre)
- Problèmes dans $\mathcal{NP}$: il existe un certificat polynomial (faciles à vérifier)
- Si Q se réduit polynomialement à P, alors P est au moins aussi dur que Q
- Problème $\mathcal{NP}$-difficile : problème vers lequel tout problème dans $\mathcal{NP}$ se réduit polynomialement
- Problème $\mathcal{NP}$-complet : problème $\mathcal{NP}$-difficile qui est dans $\mathcal{NP}$

Diagramme des classes de complexité



La classe co-NP

Définition

Le problème A est dans co-$\mathcal{NP}$ s'il existe un algorithme vérifieur pour A vérifiant les conditions suivantes :

- l'algorithme vérifieur fonctionne en temps polynomial en la taille de l'entrée
- si la veritable réponse au problème est non, alors il existe un indice de taille polynomiale sur lequel l'algorithme vérifieur répond « non » (un certificat polynomial)
- si la veritable réponse est oui, alors il n'existe aucun indice sur lequel l'algorithme vérifieur répond « non » (pas de certificat).

Exemples de problèmes dans co-NP

Le problème 2-COL

Instance Un graphe G

Question G est-il biparti ?

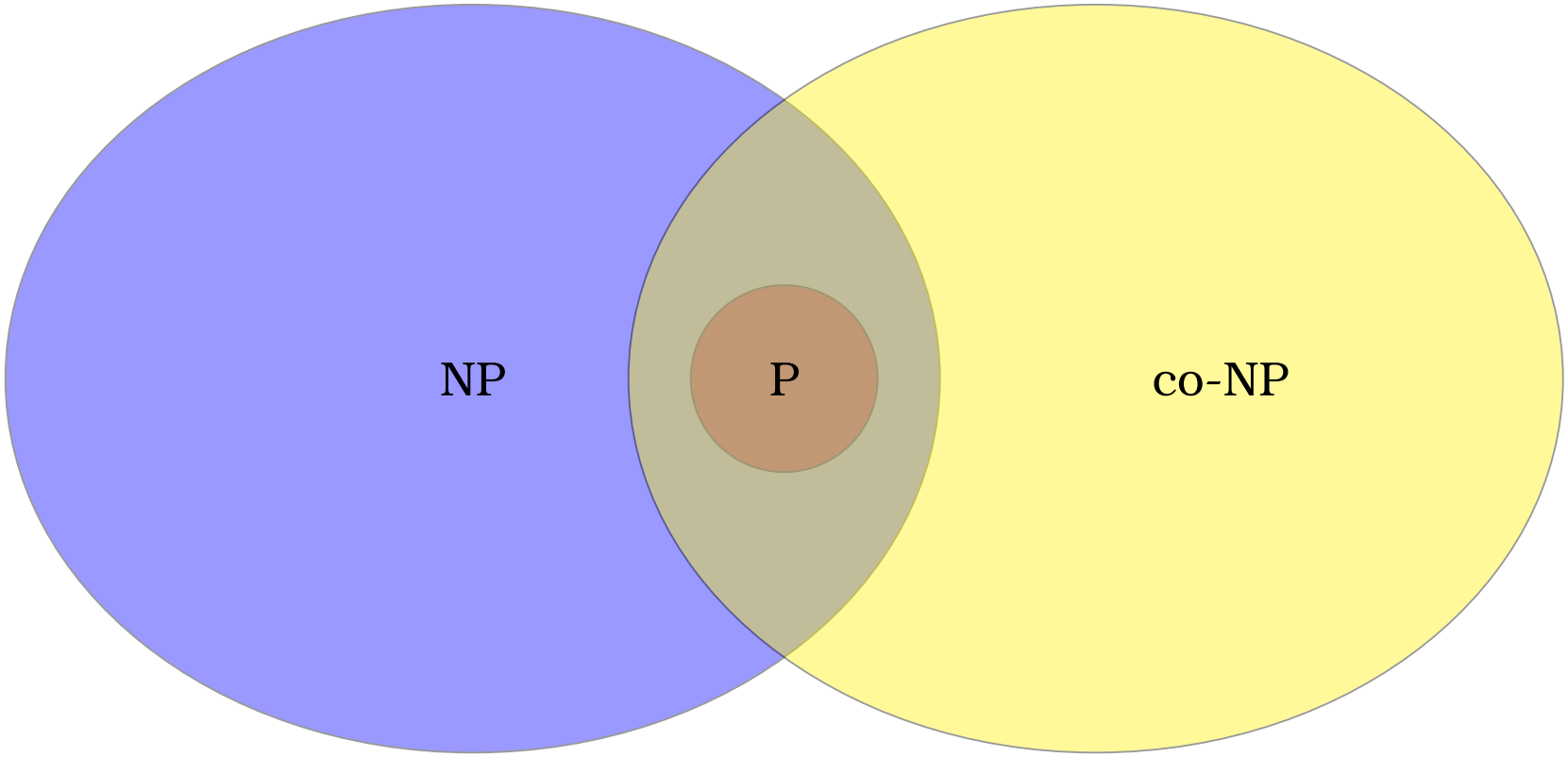
Lemme

2-COL est dans NP $\cap$ co-NP

Démonstration

- Certificat polynomial pour une réponse positive : une 2-coloration de G.
- Certificat polynomial pour une réponse négative : un cycle impair dans G.

Relation entre P, NP et co-NP



Quelques questions ouvertes

Conjecture

$\mathrm{P} \neq \mathrm{NP}$

Conjecture

$\mathrm{NP} \neq \text{co-NP}$

Conjecture

$\mathrm{NP} \cap \text{co-NP} = \mathrm{P}$

Formules de la logique propositionnelle

- Les formules de la logique propositionnelle sont construites à partir de variables propositionnelles et des opérateurs booléens :
  - $\wedge$ conjonction (et)
  - $\vee$ disjonction (ou)
  - $\neg$ négation (non)
- Une formule est *satisfaisable* s'il existe une assignation des variables propositionnelles qui rend la formule logiquement vraie.

Exemple

- La formule $(x \wedge y) \vee \neg x$ est satisfaisable car si x prend la valeur faux, la formule est évaluée à vrai
- La formule $x \wedge \neg x$ n'est pas satisfaisable car aucune valeur de x ne peut rendre la formule vraie

La forme normale conjonctive (CNF)

Définition

- Un *littéral* ℓ est :
  - une variable propositionnelle v_j (littéral positif) ou
  - la négation d'une variable propositionnelle $\neg v_j$ (littéral négatif).
- Une *clause* est une disjonction de littéraux de la forme $\ell_1 \vee \ell_2 \vee \ldots \vee \ell_n$.
- Une expression logique est en CNF ssi elle est une conjonction d'une ou plusieurs disjonction(s) d'un ou plusieurs littéraux.

Exemple

- $f = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_2 \vee \neg x_1)$

Le problème de satifaisabilité (SAT)

Le problème SAT

Instance Une formule boolénne en forme normale conjonctive (CNF)

Question Existe-t-il une assignation des variables qui rende la formule vraie ?

Exemple

- Soit $f = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_2 \vee \neg x_1)$
- L'assignation $x_1 =$ faux, $x_2 =$ vrai, $x_3 =$ vrai rend la formule f vraie

Théorème de Cook

Théorème
Le problème SAT est $\mathcal{NP}$-complet.

- Facile à voir que SAT est dans $\mathcal{NP}$.
- Moins facile à prouver que SAT est $\mathcal{NP}$-difficile ; nous admettrons ce résultat sans preuve.
- Le problème SAT est le problème $\mathcal{NP}$-complet de référence.
- Nous allons utiliser le théorème de Cook pour montrer que d'autres problèmes sont $\mathcal{NP}$-difficiles.
- Si SAT se réduit polynomialement à P, alors P est $\mathcal{NP}$-difficile.

Le problème 3-SAT

Le problème 3-SAT

Instance : Une formule booléenne en forme normale conjonctive (CNF) où les clauses contiennent *au plus* 3 littéraux

Question : Existe-t-il une assignation des variables qui rende la formule vraie ?

Exemple

- $f = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$

3-SAT est dans NP

Lemme

3-SAT est dans NP.

Démonstration

- L'algorithme vérifieur prend en entrée l'instance (formule booléenne f) et comme indice, une assignation des variables.
- Il vérifie si l'assignation rende chaque clause de la formule vraie.
- Si toutes les vérifications réussissent, il renvoie OUI, sinon il renvoie NE PERMET PAS DE CONCLURE.
- On vérifie qu'il existe un certificat si et seulement si f a une assignation de variables qui rend la formule vraie.
- L'algorithme vérifieur fonctionne bien en temps polynomial en n, le nombre de variables de f.

3-SAT est NP-difficile

Lemme

SAT se réduit à 3-SAT

Démonstration

- Étant donné une instance du problème SAT (une formule), il faut trouver une instance de 3-SAT équivalente.
- On remplace chaque clause $C = x \vee y \vee z \vee t \vee u \vee \ldots$ par une conjonction de clauses à 3 littéraux connectées par de nouvelles variables :
- $R(C) = (x \vee y \vee a) \wedge (\bar{a} \vee z \vee b) \wedge (b \vee t \vee c) \wedge (\bar{c} \vee u \vee d) \wedge \ldots$
- La formule $f = C_1 \wedge C_2 \wedge \ldots$ est satisfaisable ssi la formule $R(F) = R(C_1) \wedge R(C_2) \wedge \ldots$ l'est.
- On construit $R(F)$ en temps polynomial.

Le problème CLIQUE

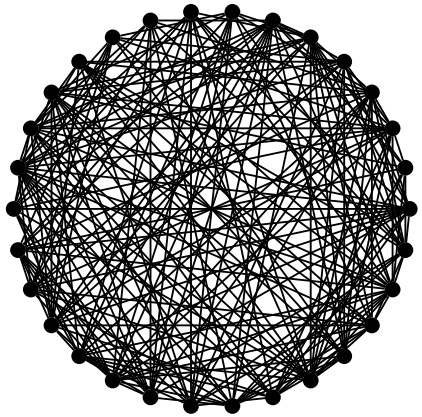
Problème CLIQUE

Instance : Un graphe G et un entier k

Question : Existe-t-il une clique de taille au moins k dans G ?

Exemple

Existe-t-il une clique de taille 10 dans le graphe suivant ?



CLIQUE est dans NP

Lemme

CLIQUE est dans NP.

Démonstration

- L'algorithme vérifieur prend en entrée l'instance (G, k) et comme indice, un ensemble de sommets W.
- Il vérifie si W contient au moins k sommets et si $\forall u, v \in W$, $uv \in E$.
- Si toutes les vérifications réussissent, il renvoie OUI, sinon il renvoie NE PERMET PAS DE CONCLURE.
- On vérifie qu'il existe un certificat ssi G a une clique de taille $\geq k$.
- L'algorithme vérifieur fonctionne bien en temps polynomial en n, le nombre de sommets de G.

CLIQUE est NP-difficile

Lemme

3-SAT se réduit à CLIQUE (3-SAT $\leq_p$ CLIQUE).

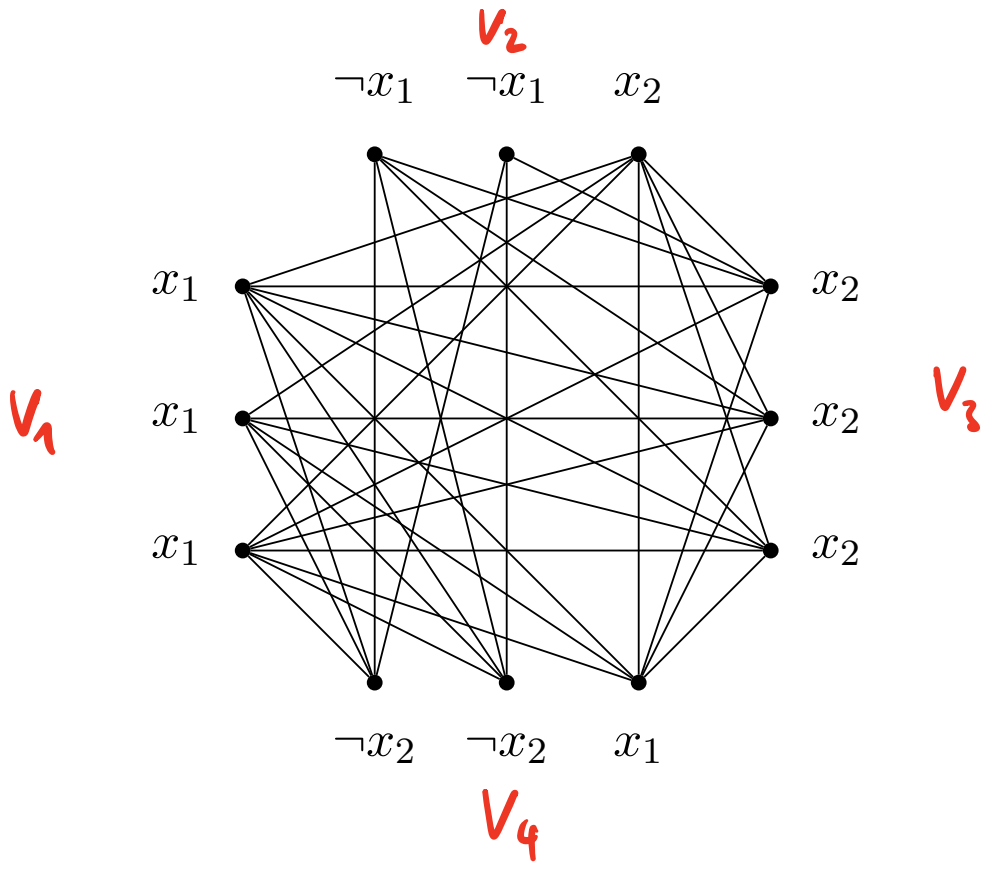
Démonstration

- Soit f une instance de 3-SAT : une formule f avec n clauses $C_1, C_2, \ldots, C_n$ à au plus 3 littéraux.
- On construit un graphe G avec au plus $3n$ sommets, un pour chaque occurence de variable dans une clause.
- L'ensemble des sommets de G consiste de n parties $V_1, V_2, \ldots, V_n$, correspondants aux clauses $C_1, C_2, \ldots, C_n$.
- Les sommets de V_i sont étiquetés par les littéraux de la clause C_i.
- Le graphe G contient toutes les arêtes entre différentes parties, *sauf* entre des paires de la forme $\{x, \neg x\}$.

Exemple

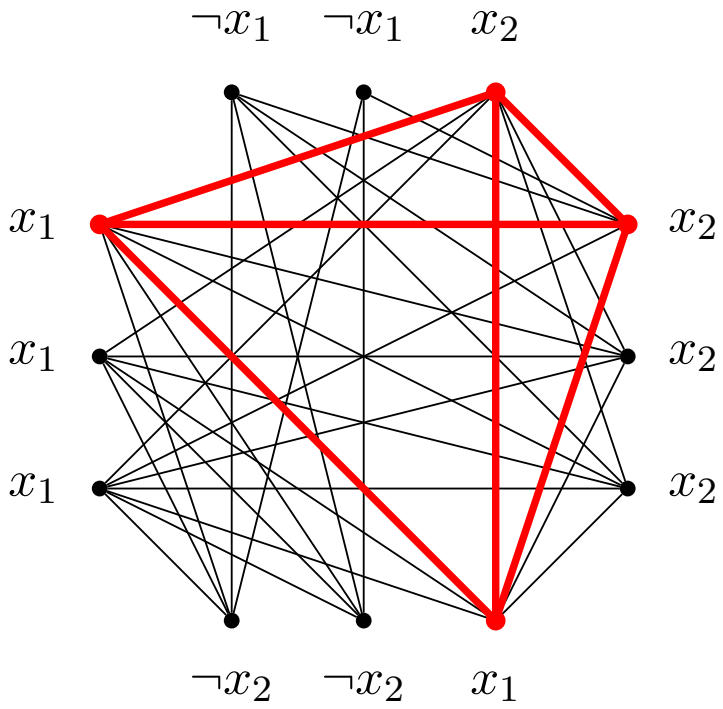
C_1 C_2 C_3 C_4

$$f = (x_1 \vee x_1 \vee x_1) \wedge (\neg x_1 \vee \neg x_1 \vee x_2) \wedge (x_2 \vee x_2 \vee x_2) \wedge (\neg x_2 \vee \neg x_2 \vee x_1)$$



Exemple

$$f = (x_1 \vee x_1 \vee x_1) \wedge (\neg x_1 \vee \neg x_1 \vee x_2) \wedge (x_2 \vee x_2 \vee x_2) \wedge (\neg x_2 \vee \neg x_2 \vee x_1)$$



Lemme

La formule f est satisfaisable ssi G contient une clique de taille au moins n

Démonstration

- Soit A une assignation de f.
- Dans chaque clause C_i, au moins un littéral est évalué à vrai.
- Pour chaque i, soit v_i le sommet de V_i étiqueté par le premier littéral évalué VRAI dans A.
- Les sommets v_i forment une clique de taille n dans G :
  - Supposons par l'absurde que $v_i v_j \notin E(G)$
  - Par la définition de G, $\{v_i, v_j\} = \{x, \neg x\}$, où x est une variable de f.
  - Or, A ne peut pas satisfaire à la fois x et $\neg x$; contradiction.

Démonstration

- Soit W l'ensemble des sommets d'une clique de taille n dans G.
- Nous allons construire une assignation A qui satisfait f
- W contient exactement un sommet de chaque V_i.
- Pour chaque variable x de f, mettre $x = \text{VRAI}$ ssi il existe un sommet étiqueté x dans W.
- Pour chaque i, un sommet de V_i est dans W.
- Donc, A satisfait au moins un littéral de chaque clause.
- On conclut que A satisfait f.

CLIQUE est NP-complet

- Nous avons montré que f est satisfaisable ssi G a un stable de taille n.
- La construction de (G, n) se fait en temps polynomial à partir de f.
- Donc, 3-SAT $\leq_p$ CLIQUE
- Comme 3-SAT est NP-difficile, CLIQUE est NP-difficile aussi.
- De plus, nous avons montré que CLIQUE est dans NP.
- Nous avons ainsi démontré le théorème suivant :

Théorème

Le problème CLIQUE est NP-complet.

Le problème STABLE

Problème STABLE

Instance : Un graphe G et un entier k

Question : Existe-t-il un stable de taille au moins k dans G ?

Lemme

STABLE est dans NP

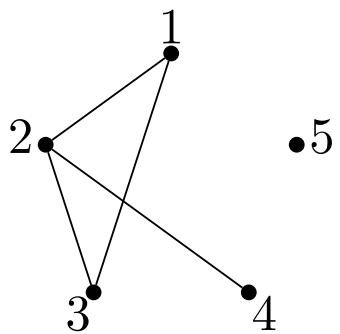
Démonstration

Même preuve que pour CLIQUE, sauf qu'il faut vérifier que $\forall u, v \in W$, $uv \notin E$.

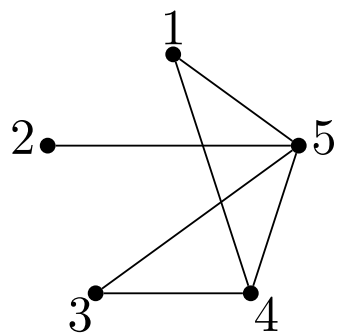
STABLE est NP-difficile

Lemme

CLIQUE se réduit à STABLE.



G

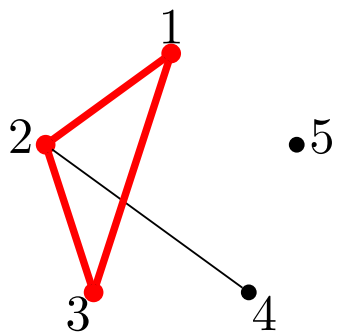


$\overline{G}$

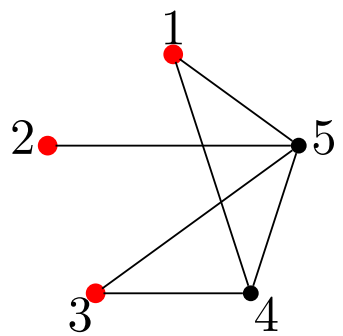
STABLE est NP-difficile

Lemme

CLIQUE se réduit à STABLE.



G



$\overline{G}$

Réduction de CLIQUE à STABLE

Démonstration

- Soit (G, k) une instance de CLIQUE.
- Considérons l'instance $(\overline{G}, k)$ de STABLE.
- Il est clair que $\overline{G}$ a un stable de taille au moins k ssi G a une clique de taille au moins k.
- La construction de $(\overline{G}, k)$ se fait en temps polynomial à partir de (G, k).

Théorème

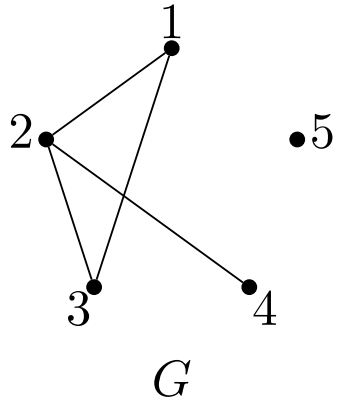
Le problème STABLE est NP-complet.

Le problème VERTEX-COVER

Problème VERTEX-COVER

Instance : Un graphe G et un entier k

Question : Existe-t-il une ensemble d'au plus k sommets couvrant toutes les arêtes (au moins une des 2 extrémités est couverte) ?



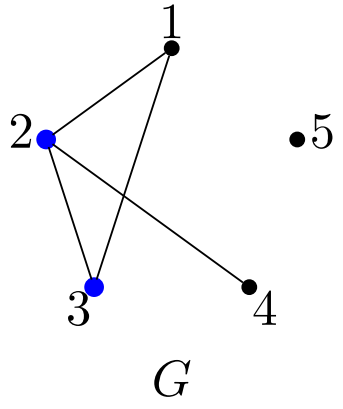
G

Le problème VERTEX-COVER

Problème VERTEX-COVER

Instance : Un graphe G et un entier k

Question : Existe-t-il une ensemble d'au plus k sommets couvrant toutes les arêtes (au moins une des 2 extrémités est couverte) ?



G

VERTEX-COVER est dans NP

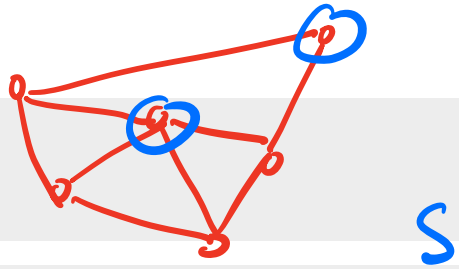
Lemme

VERTEX-COVER est dans NP.

Démonstration

- L'algorithme vérifieur prend en entrée l'instance (G, k) et comme indice, un ensemble de sommets W.
- Il vérifie si W contient au plus k sommets et si $\forall e \in E(G)$, $W \cap e \neq \emptyset$.
- Si toutes les vérifications réussissent, il renvoie OUI, sinon il renvoie NE PERMET PAS DE CONCLURE.
- On vérifie qu'il existe un certificat ssi G a un recouvrement de taille $\leq k$.
- L'algorithme vérifieur fonctionne bien en temps polynomial en n, le nombre de sommets de G.

VERTEX-COVER est NP-difficile



Lemme

STABLE se réduit à VERTEX-COVER

Démonstration

- Soit (G, k) une instance de STABLE.
- Considérons l'instance $(G, n - k)$ de VERTEX-COVER, ou n est le nombre de sommets de G.
- S est un stable de G ssi $V(G) \setminus S$ couvre toute les arêtes de G.
- G a un recouvrement d'au plus $n - k$ sommets ssi G a un stable de taille au moins k.
- La construction de $(G, n - k)$ se fait en temps polynomial à partir de (G, k).

VERTEX-COVER est NP-complet

- Nous avons montré que VERTEX-COVER est à la fois dans NP est NP-difficile.
- Donc, on a le théorème suivant :

Théorème

Le problème VERTEX-COVER est NP-complet.