Chapitre 4 : Réductions et complétude

ENSIIE - Théorie de la complexité

Dimitri Watel (dimitri.watel@ensiie.fr)

2017

Réduction polynomiale et complétude

Idée de base

Une manière de résoudre un problème Π_1 est de le transformer en un autre Π_2 qu'on sait résoudre.

Si la transformation est rapide et que le problème Π_2 peut être résolu rapidement, alors Π_1 peut être résolu rapidement.

Si la transformation est rapide et que le problème Π_1 ne peut être résolu rapidement, alors Π_2 ne peut être résolu rapidement.

Réduction polynomiale de Karp

Définition (informelle)

Soit deux problèmes de décision Π_1 et Π_2, une réduction polynomiale de Π_1 vers Π_2 transforme en temps polynomial toute instance positive (resp. negative) de Π_1 en une instance positive (resp. negative) de Π_2.

Si Π_1 se réduit à Π_2, alors on dit que Π_2 est *plus difficile* que Π_1.

Réduction polynomiale de Karp

Définition

Soit deux problèmes de décision Π_1 et Π_2, une réduction polynomiale de $\Pi_1 = (\mathcal{L}^1, \mathcal{L}^1_Y, \mathcal{L}^1_N)$ vers $\Pi_2 = (\mathcal{L}^2, \mathcal{L}^2_Y, \mathcal{L}^2_N)$ est un algorithme $\mathcal{R}$ tel que

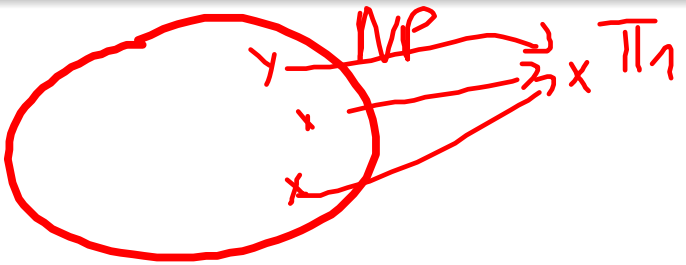
- $\mathcal{R}$ a une complexité polynomiale
- $\mathcal{R}$ prend en entrée une instance $\mathcal{I}$ de Π_1 et renvoie en sortie une instance $\mathcal{J}$ de Π_2
- $\mathcal{I} \in \mathcal{L}^1_Y \Leftrightarrow \mathcal{J} \in \mathcal{L}^2_Y$

On note $\Pi_1 \preccurlyeq \Pi_2$.

Difficulté

Problème $\mathcal{C}$-difficile

Soit $\mathcal{C}$ une classe de complexité (NP, EXPTIME, ...) et Π_1 un problème de décision. On dit que Π_1 est $\mathcal{C}$-Difficile si, pour tout problème Π_2 de $\mathcal{C}$, $\Pi_2 \preccurlyeq \Pi_1$.



Complétude

Problème $\mathcal{C}$-complet

Soit $\mathcal{C}$ une classe de complexité (NP, EXPTIME, ...) et Π un problème de décision. On dit que Π est $\mathcal{C}$-Complet si $\Pi \in \mathcal{C}$ et s'il est $\mathcal{C}$-difficile.

Π_1 le pb le plus dur de NP.

Résultats importants

Démontrer la difficulté

- Si Π_1 est $\mathcal{C}$-Difficile et si $\Pi_1 \preccurlyeq \Pi_2$ alors Π_2 est $\mathcal{C}$-Difficile.

$\Pi_1 \leq \Pi_2$ et $\Pi_2 \leq \Pi_3$ alors $\Pi_1 \leq \Pi_3$

P et NP

- Si $\Pi_2 \in$ P et $\Pi_1 \preccurlyeq \Pi_2$, alors $\Pi_1 \in$ P.
- Si $\Pi_2 \in$ NP et $\Pi_1 \preccurlyeq \Pi_2$, alors $\Pi_1 \in$ NP.
- P $\neq$ NP $\Leftrightarrow \forall$ problème Π NP-Complet, $\Pi \notin P$.
- 3-SAT est NP-Complet.

Pour montrer qu'un pb Π_2 est NP-Complet :

- choisir un autre pb Π_1 NP Complet
- $\Pi_1 \in NP$
- $\Pi_1 \leq \Pi_2$

Réduction polynomiale de Turing

Définition : Réduction polynomiale de Turing

Soient deux problèmes Π_1 et Π_2, une réduction polynomiale de Turing de Π_1 vers Π_2 est un algorithme $\mathcal{R}$ tel que

- $\mathcal{R}$ peut appeler un algorithme $\mathcal{R}'$ (imaginaire), appelé *oracle*, résolvant Π_2 en temps constant ;
- $\mathcal{R}$ résoud Π_1 en temps polynomial.

On note $\Pi_1 \preccurlyeq_T \Pi_2$.

Π_1 et Π_2 ne sont pas nécessairement des problèmes de décision.

Autres réductions

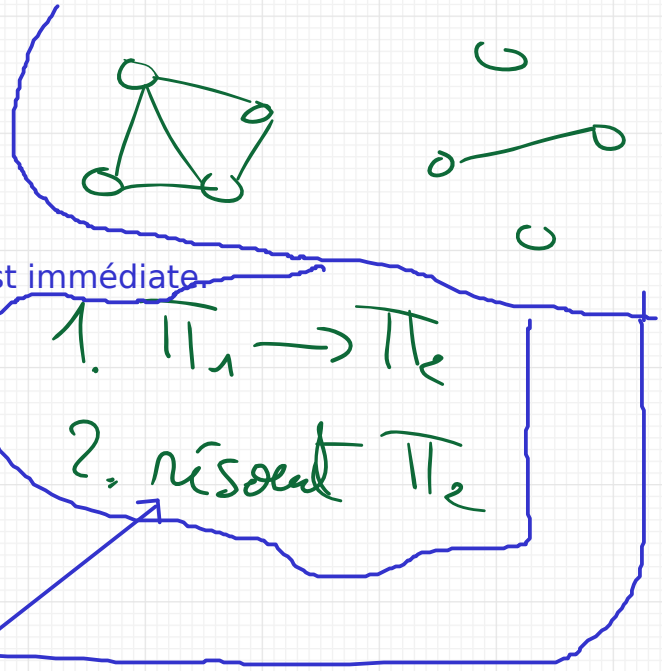
- Réduction exponentielle
- Réduction en espace polynomial
- Réduction probabiliste
- ...

C'est ce qu'il faudrait montrer si on voulait montrer qu'un problème n'appartient pas à P.

$\Pi \notin P \iff \forall$ M machine det en temps poly, M ne résout pas Π.

Π_1 (3SAT) $\xrightarrow[\text{cas particulier}]{\text{immédiat}}$ Π_2 (SAT)

3SAT est un sous pb de SAT, donc la transformation de 3SAT en SAT est immédiate.

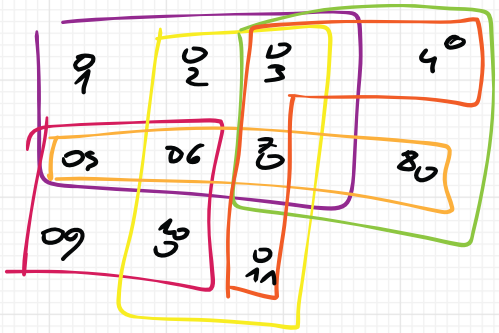


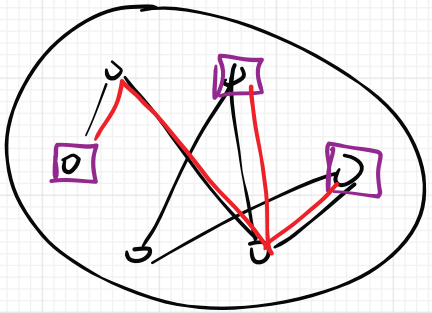
(CLIQUE) $\xrightarrow[\text{du graphe}]{\text{complémentaire}}$ (WINS)

(WINS = stable max)

Ci dessus vous avez 2 exemples de transformations qui vous montre qu'il est possible, si vous savez résoudre le pb de droite rapidement, de résoudre le pb de gauche en utilisant cette technique . Donc on en déduit que le pb de gauche est plus facile à résoudre que celui de droite puisqu'il suffit de sdavoir résoudre celui de droite pour résoudre celui de gauche.

(SETCOVER) ⟶ (UST)





Un ensemble X

Un ensemble $S \subset 2^X = \mathcal{P}(X)$

Parties de X

2^X ou P(X) sont les parties de X (donc les sous-ensembles de X)

Un entier k

$\exists ? C \subset S \;/\; |C| \leq k$

$\bigcup_{S \in C} S = X$

Un graphe $G = (V, E)$ non orienté

$Y \subset V$

un entier k

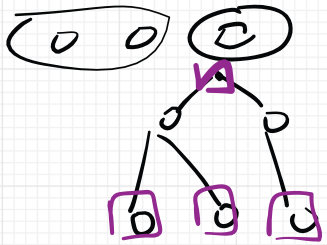
$\exists ?$ un arbre inclus dans G avec k noeuds ou moins couvrant Y ?

(un arbre est un graphe sans cycle, comme c'est le cas en rouge)

Dit autrement
Soit X un ensemble et S des parties de X,
de combien d'ensemble de S au minimum on a besoin
pour couvrir X ?

Pour chaque $s \in S$ créer $V_s \in V$

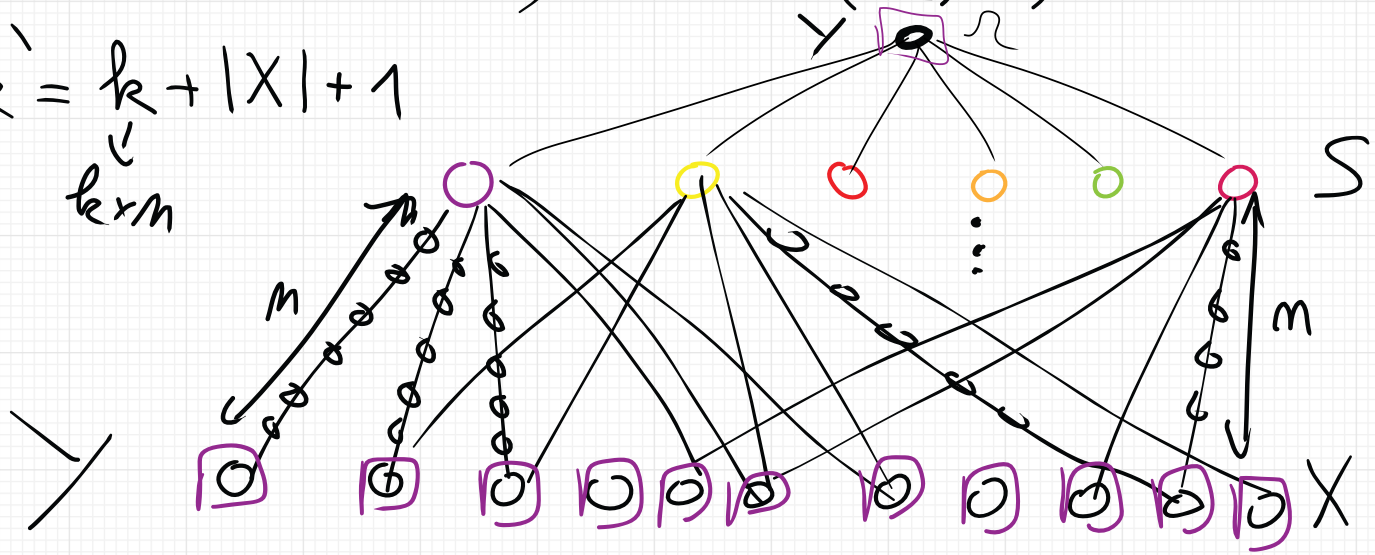
Pour chaque $e \in X$ créer $V_e \in V$



$\forall s \in S$, $\forall e \in s$, créer ~~$(V_e, V_s) \in E$~~ un chaîne de n arêtes reliant Ve et Vs

$k' = k + |X| + 1$

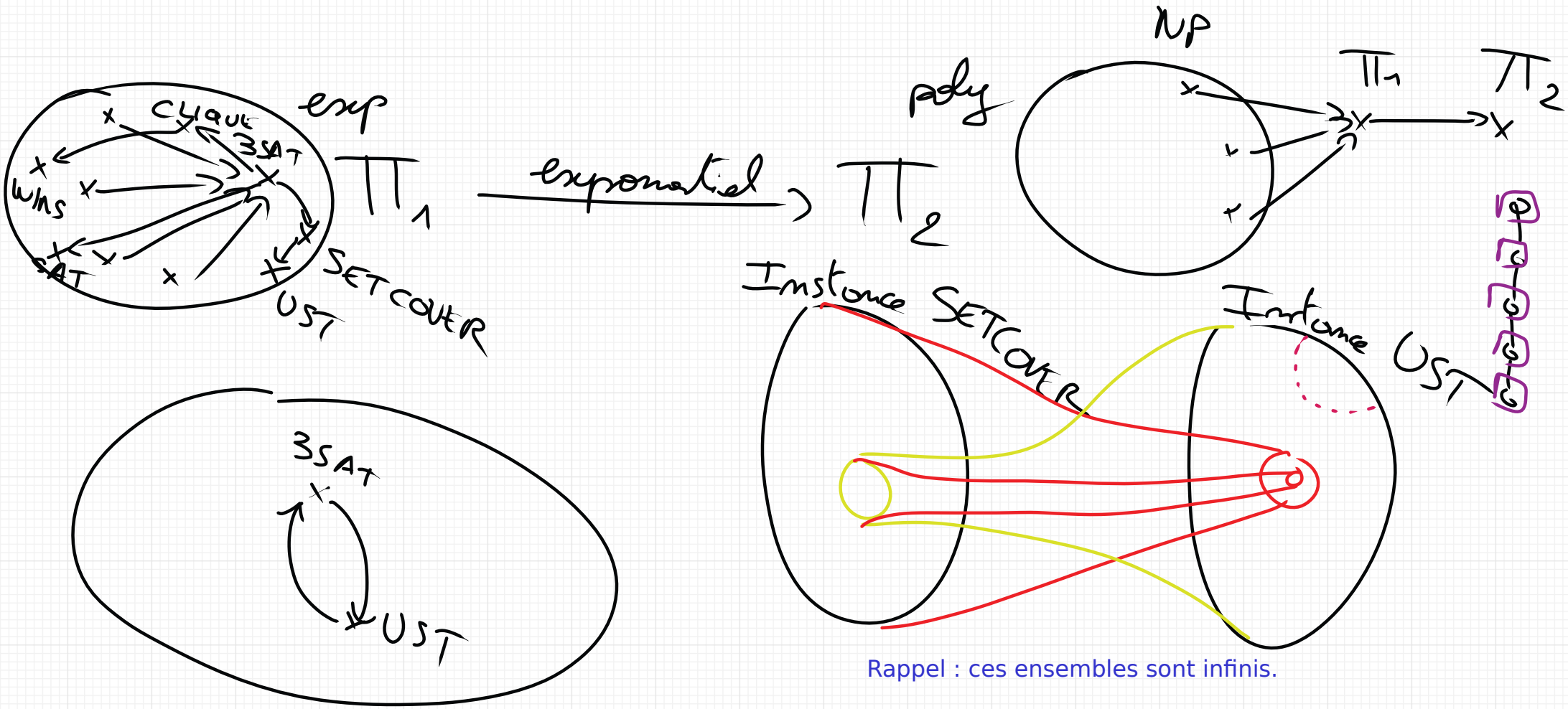
$k \times n$



Ce n noeuds font qu'il est très cher d'utiliser les arêtes du bas pour relier les noeuds de X à la racine et qu'on va donc préférer utiliser les arêtes du haut qui sont moins cher.

Dans tous les cas, pour relier r à X, il faudra utiliser |X| chaînes de taille n (une par noeud de X). L'idée est qu'on ne veut pas en utiliser plus.

Donc pour relier r à X, il faut |X| * n arêtes + 1 arête pour chaque noeud de S relié à r.
Donc on veut minimiser le nombre de noeuds de S reliés à r.
Or chaque noeud de S est en fait à l a base un ensemble de l'instance de Set Cover, donc on veut utiliser le moins d'ensemble de Set Cover possible pour relier r à X.
Or relier r à X signifie qu'on utilise des ensembles dont l'union est X ; donc c'est une solution de Set Cover.
Donc trouver la solution qui utilise le moins de noeuds de S est une solution de Set Cover qui utilise le moins d'ensembles possibles.



Rappel : ces ensembles sont infinis.

Mq $\Pi_2 \in P$, $\Pi_1 \leq \Pi_2 \Rightarrow \Pi_1 \in P$

$\Pi_2 \in P$ ($\exists$ Algo A pour résoudre Π_2 en temps poly $O(n^p)$
où n = taille de l'instance de Π_2

$\Pi_1 \leq \Pi_2$ ($\exists$ algo R qui transforme une entrée de Π_1 en entrée de Π_2
en temps $O(m^q)$ m taille de l'entrée de Π_1

Soit $B = A \circ R$ $\longrightarrow$ Soit x instance de Π_1 de taille m

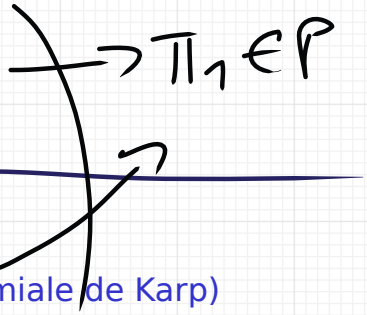
$A \circ R(x) = A(R(x))$
ou
$y \leftarrow R(x)$, renvoyer $A(y)$

$y \leftarrow R(x)$: $O(m^q)$; renvoyer $A(y)$: $O(|y|^p)$

y produit par R en temps $\Theta(m^q)$

Donc $|y| \leq \Theta(m^q)$ sinon on aurait écrit y en moins de temps que sa taille.

$\Rightarrow \Theta(|y|^p) = \Theta(m^{pq})$ polynomial



B est correct car

(cf définition d'une réduction polynomiale de Karp)

R transforme les réponses OUI de Π_1 en réponse OUI de Π_2
NON ... NON ...

$A(y) = \text{OUI} \iff$ la réponse de x est OUI

$\Updownarrow$

$A(R(x)) = \text{OUI} \iff B(x) = \text{OUI}$

Montrer si $\Pi_2 \in NP$ $\Pi_1 \leq \Pi_2$ $\Pi_1 \in NP$

$\Pi_2 \in NP$: $\exists$ algo non det A qui résout Π_2 en temps poly

$\Pi_1 \leq \Pi_2$: $\exists$ réduction (déf) R poly de Karp de Π_1 vers Π_2

Soit $B = A \circ R$

B non det
B temps poly
B correct
(démo précédente)

$\Rightarrow \Pi_1 \in NP$

$P \neq NP \iff \forall \pi$ NP Completo $\pi \notin P$

($\Leftarrow$) (contraposição)

$\exists \pi$ NP completo $\in P$

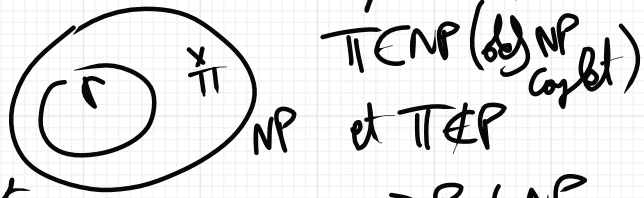
$\Rightarrow \forall \pi_2 \in NP \quad \pi_2 \leq \pi$ (def NP-Completo)

$\Rightarrow \forall \pi_2 \in NP, \pi_2 \in P$

$\Rightarrow NP \subset P$

ou $P \subset NP \Rightarrow P = NP$

($\Rightarrow$) $\forall \pi$ NP Completo $\pi \notin P$



$\pi \in NP$ (def NP Completo) e $\pi \notin P$

$\Rightarrow P \neq NP$

e

$\exists$ um pb NP Completo (3SAT)