La NP-complétude

Johanne Cohen

LRI-CNRS, Orsay, France.

Références

1. *Algorithm Design*, Jon Kleinberg, Eva Tardos, Addison-Wesley, 2006.

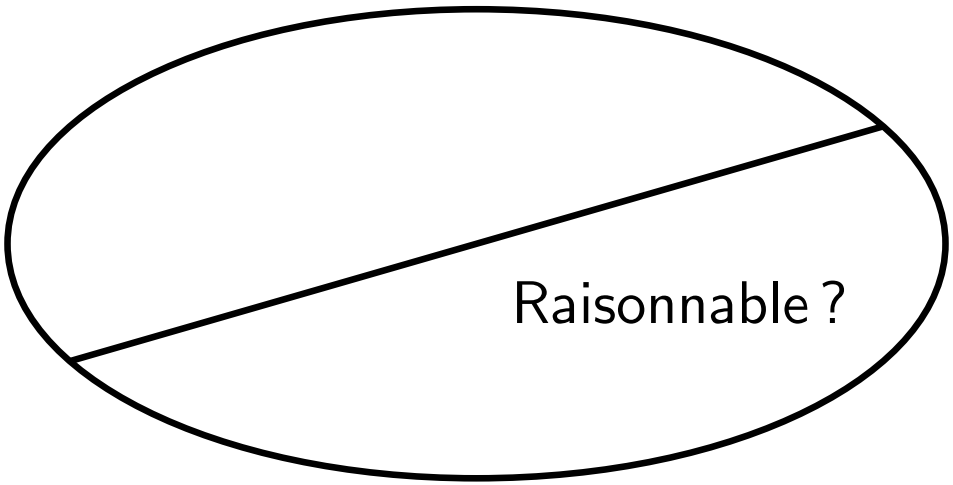
2. *Computers and Intractability : A Guide to the Theory of NP-Completeness*, M. R. Garey, D. S. Johnson, 1976.

3. Transparents inspirés du cours INF 423, Ecole Polytechnique *"Fondements de l'informatique : Logique, Modèles, Calculs"* .

Objectif de ce cours

Distinguer ce qui est raisonnable de ce qui n'est pas raisonnable en terme de temps de calcul.

Problèmes décidables.



Complexité d'un algorithme, d'un problème

- $Complexite(\mathcal{A}, d)$ est le nombre d'opérations effectuées par algorithme $\mathcal{A}$ ayant d comme une entrée.

  On cherche souvent à évaluer cette complexité en fonction de la taille des entrées $taille(d)$.

- On peut alors parler de la complexité (au pire cas)
  - d'un algorithme (on fait varier seulement les entrées)

$$Complexite_{\mathcal{A}}(n) = \max_{d\,/\,taille(d)=n} Complexite(\mathcal{A}, d).$$

  - d'un problème (on fait varier l'algorithme, et les entrées)

$$\inf_{\mathcal{A} \text{ correct}} Complexite_{\mathcal{A}}(n).$$

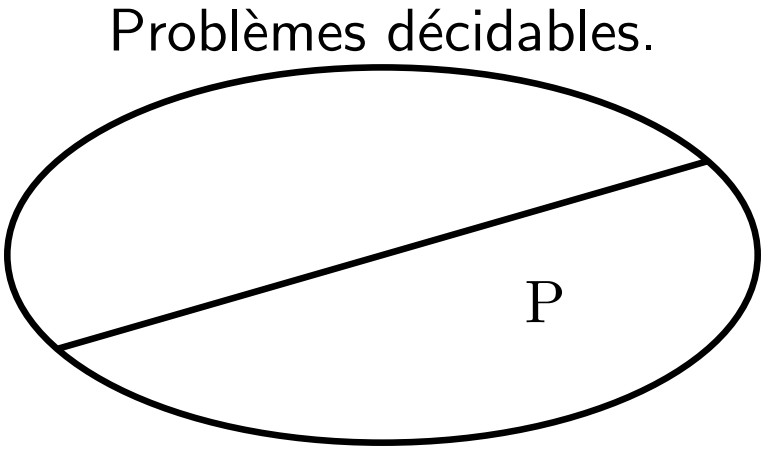
Plus précisément

Retour sur l'épisode précédent

- Une convention
- Pourquoi cette convention ?

Une convention

- La convention suivante s'est imposée en informatique :
  - **CONVENTION :** Temps raisonnable $=$ temps polynomial,
    - c'est-à-dire en $\mathcal{O}(n^k)$ pour un entier k.
- Graphiquement :



- Exemples :
  - Décider si un graphe est eulerien est dans P.
  - On ne sait pas si le problème du cycle hamiltonien est dans P.

Plus précisément

Retour sur l'épisode précédent

- Une convention
- Pourquoi cette convention ?

Raison 1 : s'affranchir du modèle de calcul

- Thèse de Church :
  - Les modèles suivants se simulent deux à deux :
    - Les machines de Turing à un ruban.
    - Les machines de Turing à deux rubans.
    - Les machines à $k \geq 2$ piles
    - Les machines RAM
    - Les programmes JAVA, C, CAML, ...
  - de telle sorte que : t instructions de l'un sont simulées par un nombre d'instructions polynomial en t par l'autre.

Raison 1 : s'affranchir du modèle de calcul

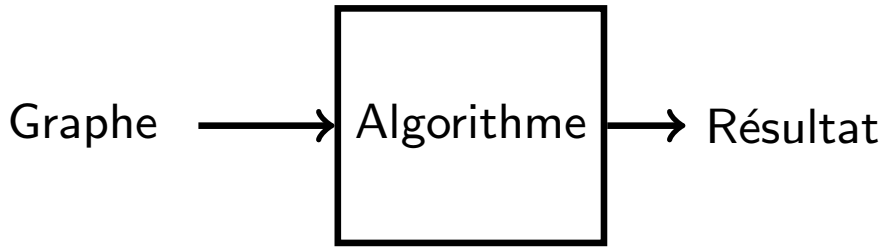
Nombre d'instructions : T

- On peut donc parler d'algorithme **efficace** sans avoir à préciser dans quel langage / avec quel modèle l'algorithme est implémenté.

- Le temps correspond au nombre d'instructions dans chacun de ces modèles.

(à la composition par un polynôme près)

Raison 2 : s'affranchir de problèmes du codage



Temps : Polynomial/Non polynomial

La classe P

Rappel : Un problème de décision Π est un ensemble d'instances I_Π et un sous-ensemble $Oui(\Pi)$ d'instances positives.

Exemple : Graphe eulerien
Données : un graphe non-orienté $G = (V, E)$.
Question : G a-il un cycle eulerien ?

Définition

La classe P est la classe des problèmes de décision qui admettent un algorithme de complexité polynomiale.

La classe NP

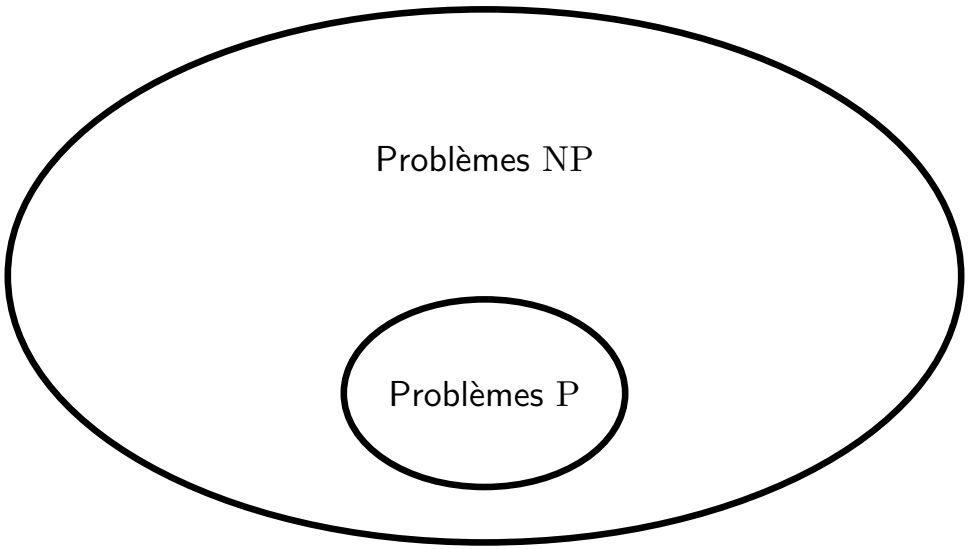
Définition

La classe NP est formée des problèmes de décision Π qui possèdent un vérificateur polynomial.
Un vérificateur V est un algorithme qui prend une information en plus (certificat) pour vérifier qu'une instance est positive.

Exemple 1 : Graphe Hamiltonien
Données : un graphe non-orienté $G = (V, E)$.
Question : G a-il un cycle hamiltonien ?

- Le certificat correspond à une suite S de sommets
- V vérifie que S est un cycle et qu'il transverse chaque sommet une unique fois.
- V fonctionne bien en temps polynomial.

Comparaison entre les deux classes P et NP.



Par définition $\mathrm{P} \subset \mathrm{NP}$.

Plus précisément

Les classes P et NP

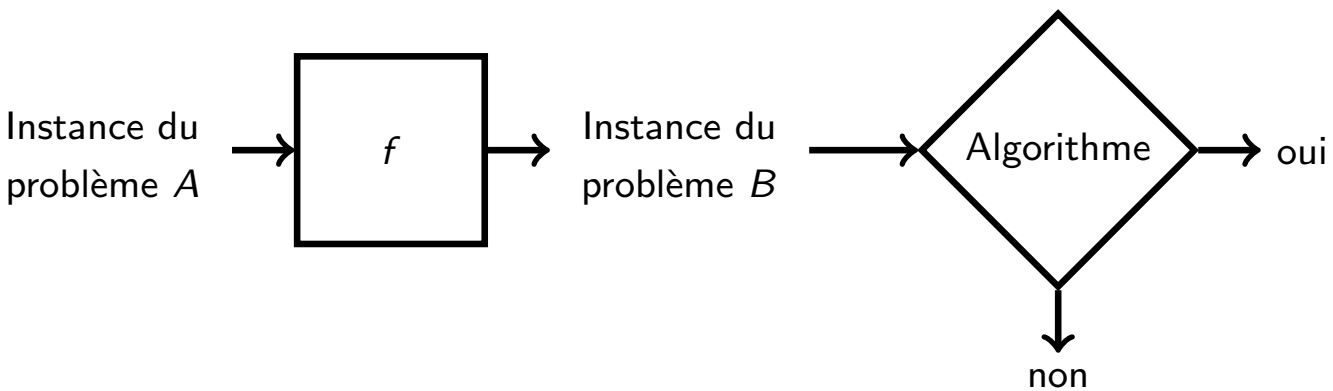
La notion de réduction

Comment comparer les problèmes

- Soient A et B deux problèmes de décision.
  Une **réduction de A vers B** est une fonction $f : I_A \to I_B$ calculable **en temps polynomial** telle que

$$w \in Oui(A) \text{ ssi } f(w) \in Oui(B).$$

- On note $A \leq B$ lorsque A se réduit à B.
  - intuitivement : $A \leq B$ signifie que A est plus facile que B.



Principales propriétés

Théorème

$\leq$ est un préordre ($=$ est reflexive, transitive) :

1. $L \leq L$;
2. $L_1 \leq L_2$, $L_2 \leq L_3$ impliquent $L_1 \leq L_3$.

Preuve:

- Intuitivement : un problème est aussi facile (et difficile) que lui-même

  il suffit de considérer la fonction identité pour f

- Intuitivement : la relation "être plus facile que" est transitive.

$L_1 \leq L_2$ via la réduction f

$\Longrightarrow x \in OUI(I_{L_1})$ ssi $g(f(x)) \in OUI(I_{L_2})$.

$L_2 \leq L_3$ via la réduction g

La composée de deux fonctions calculable **en temps polynomial** est calculable. □

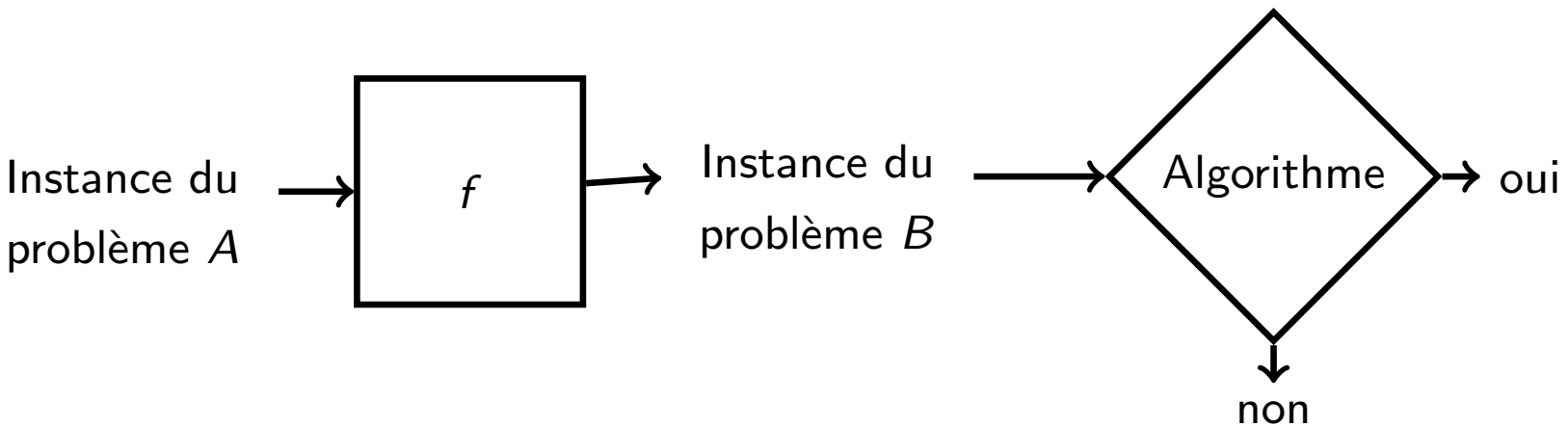
Théorèmes

Théorème

Si $A \leq B$, et si A est dans P, alors B est dans P.

Théorème

Si $A \leq B$, et si A n'est pas dans P, alors B n'est pas dans P.



Motivation

- On souhaite comprendre quels sont les problèmes les plus difficiles dans NP.
- Un problème A est dit NP-difficile si tout autre problème B de NP est tel que $B \leq A$.
  - Intuitivement : il est plus difficile que tous les problèmes dans la classe.
- Un problème A est dit NP-complet si en plus on a $A \in \mathrm{NP}$.
  - Autrement dit : A est NP-complet signifie que A est un élément maximum dans NP pour $\leq$.

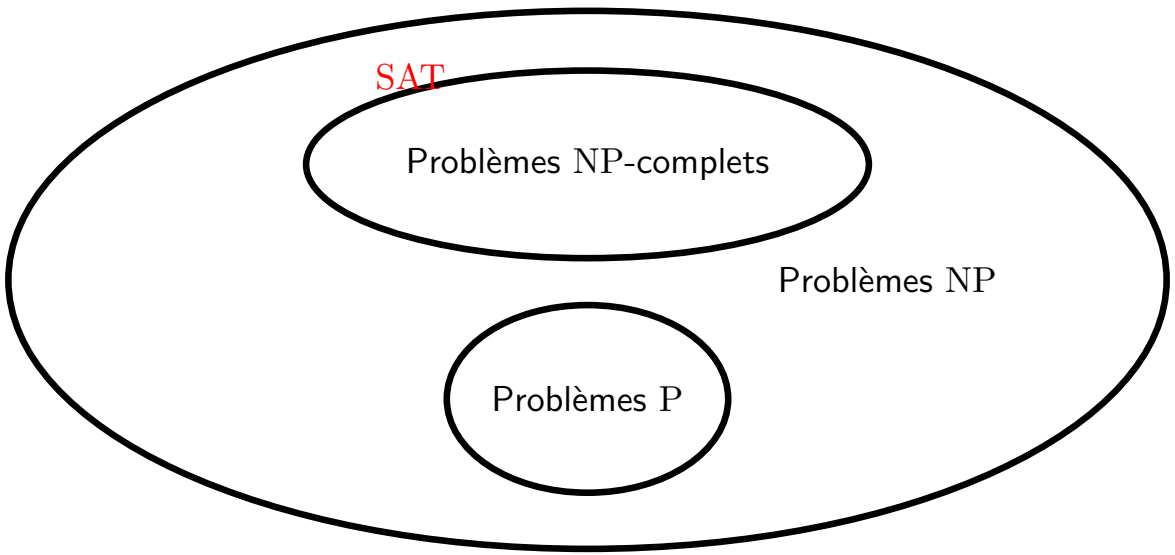
Plus précisément

- NP-complétude
  - Théorème de Cook-Levin
  - Comment pour prouver la NP-complétude

Théorème de Cook-Levin

Théorème Cook-Levin

Le problème SAT est NP-complet.



Conséquense

Corollaire

$$\mathrm{P} = \mathrm{NP} \text{ si et seulement si } \mathrm{SAT} \in \mathrm{P}.$$

Preuve:

- Si $\mathrm{P} = \mathrm{NP}$, alors puisque SAT est dans NP,

  $\mathrm{SAT} \in \mathrm{P}$.

- Réciproquement, si $\mathrm{SAT} \in \mathrm{P}$,
  - Puisque SAT est complet,

    pour tout problème $B \in \mathrm{NP}$, $B \leq \mathrm{SAT}$
  - Donc $B \in \mathrm{P}$

□

Remarque: généralisation à n'importe problème NP-complet

- Soit A un problème NP-complet.

  $$\mathrm{P} = \mathrm{NP} \text{ si et seulement si } A \in \mathrm{P}.$$

- D'où l'intérêt de produire de nombreux problèmes NP-complets

A quoi sert de prouver la NP-complétude d'un problème ?

- Arriver à prouver que $\mathrm{P} = \mathrm{NP}$...
  - Si un problème A et un problème B sont NP-complets, alors $A \leq B$ et $B \leq A$:

    Tous les problèmes NP-complets sont donc de même difficulté.

- Surtout :
  - Supposons que l'on n'arrive pas à trouver un algorithme polynomial pour un problème.

    Prouver sa NP-complétude permet de se convaincre que cela n'est pas possible, sauf si $\mathrm{P} = \mathrm{NP}$.

Plus précisément

- NP-complétude
  - Théorème de Cook-Levin
  - Comment pour prouver la NP-complétude

Stratégie pour prouver la NP-complétude

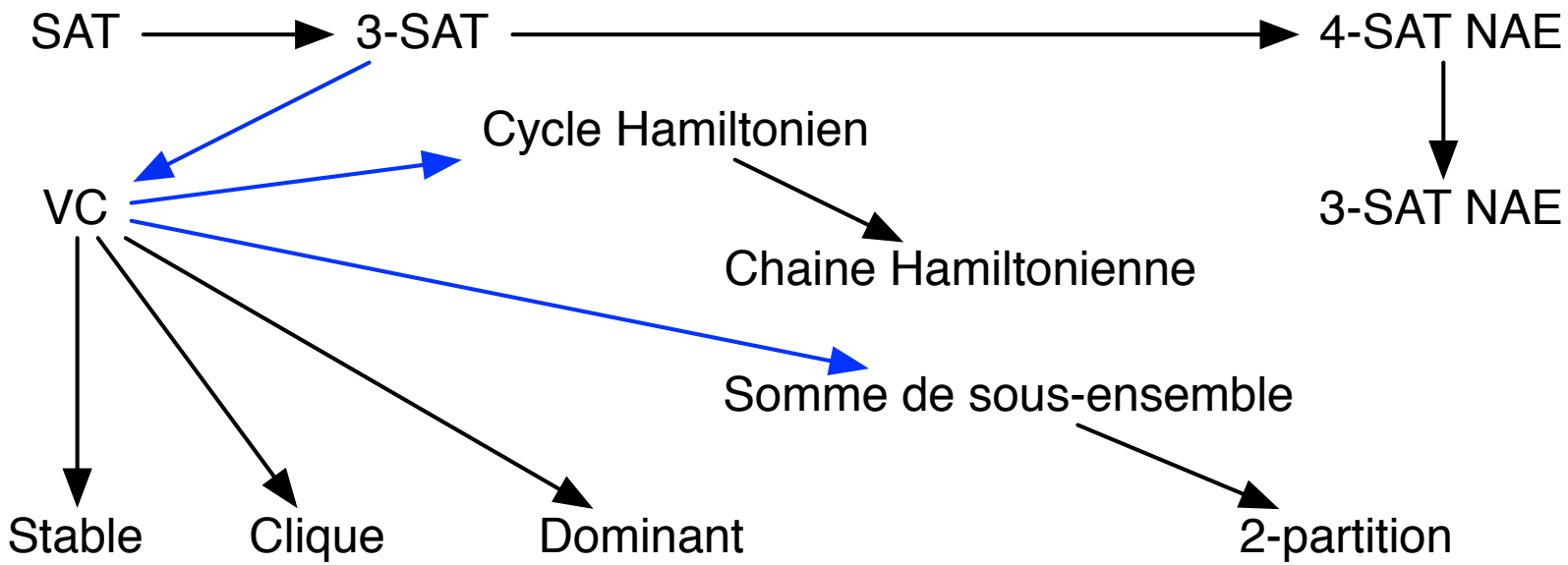
Pour prouver la NP-complétude d'un problème A, il suffit :

1. de prouver qu'il admet un vérificateur polynomial ;
2. de prouver que $B \leq A$ pour un problème B NP-complet

Pourquoi ?

- le point 1. permet de garantir que $A \in \mathrm{NP}$,
- le point 2. permet que on a $C \leq A$ pour tout problème $C \in \mathrm{NP}$
  - on a $C \leq B$ comme B est NP-complet,
  - et $C \leq A$ puisque $B \leq A$.

Ce qu'on va prouver dans la suite



Le symbole $\longrightarrow$ désigne le relation "est plus facile que"

Plus précisément

Exemple de réduction

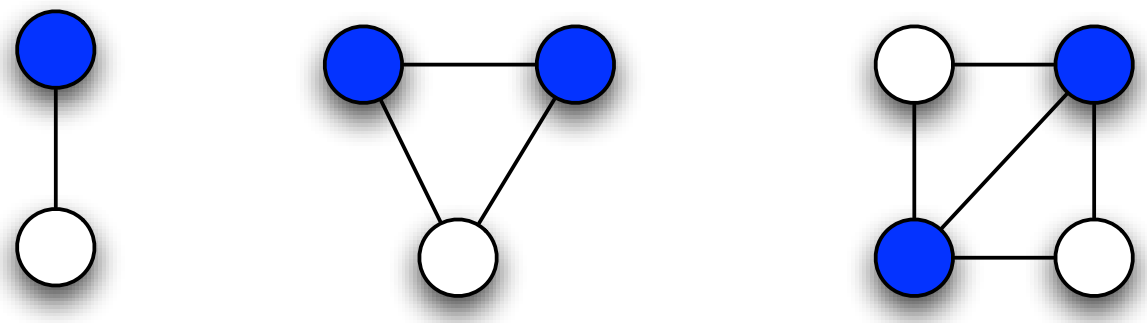
- Problème de la couverture de sommets
- Problème du cycle hamiltonien

Couverture sommet

Couverture de sommets (VC)
Données : un graphe non-orienté $G = (V, E)$ et un entier k.
Question : G contient-il une *couverture de sommets* $\mathcal{S}$ de cardinalité au plus k ?

Théorème

Le problème Couverture de sommets est NP-complet.



Les sommets en bleu font partie de la couverture de sommets

VC est dans NP

- Certificat : liste $\mathcal{S}$ de sommets.
- On peut vérifier en temps polynomial que
  1. la cardinalité de $\mathcal{S}$ est inférieure ou égale à k ;
  2. pour chaque arête $(u, v) \in E$, u ou v est dans $\mathcal{S}$

3-SAT $\leq$ VC

3-SAT
Données : un ensemble U de variables $\{u_1, u_2, \ldots, u_n\}$ et une formule logique $L = C_1 \wedge \cdots \wedge C_\ell$ ayant des clauses de 3 littéraux
Question : Existe-t-il une fonction $t : U \to \{0, 1\}$ telle que t satisfait L ?

Nous allons transformer une instance (U, L) de 3-SAT en une instance (G, k) de *VC* en un temps polynomial.

- Pour chaque variable u de U, on associe deux sommets u et $\overline{u}$, et une arête $(u, \overline{u})$ dans G
- Pour chaque clause $C_i = (x_1 \vee x_2 \vee x_3)$,
  - on associe un triangle dans G composé des sommets $c_{1,i}$, $c_{2,i}$, $c_{3,i}$,
  - on relie le sommet $c_{1,i}$ à ℓ_1 par une arête
- $k = |U| + 2\ell$

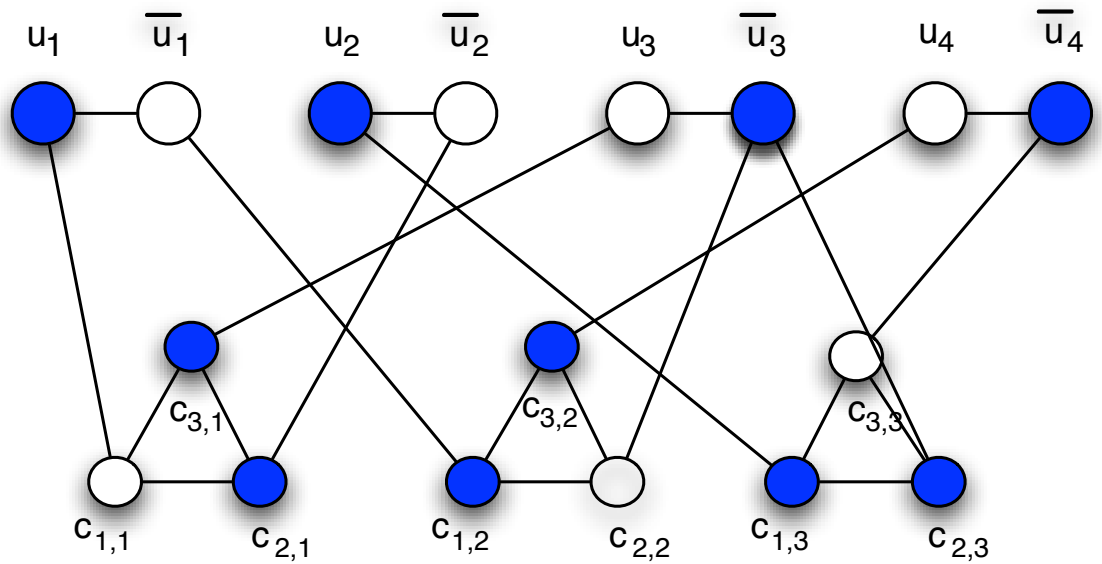
Exemple

On peut prouver :

Il existe une couverture de sommets du graphe G
d'au plus k sommets

$\Longleftrightarrow$

il existe une fonction $t : U \to \{0, 1\}$
qui satisfait toutes les clauses $C_1, \ldots, C_\ell$.



Les sommets en bleu font partie de la couverture de sommets

Plus précisément

Exemple de réduction

- Problème de la couverture de sommets
- Problème du cycle hamiltonien

Problème du cycle hamiltonien

Cycle Hamiltonien
Données : un graphe non-orienté G.
Question : G contient-il un cycle hamiltonien ?

Théorème
Le problème Cycle Hamiltonien est NP-complet

Preuve:

1. Cycle Hamiltonien est dans NP.

2. VC $\leq$ Cycle Hamiltonien.

□

VC $\leq$ Cycle Hamiltonien.

- On se donne un graphe $G = (V, E)$ et un entier k
  (correspondant à une instance de VC).

- On veut construire un graphe G' en temps polynomial tel que

  il existe une couverture de sommets du graphe G
  d'au plus k sommets

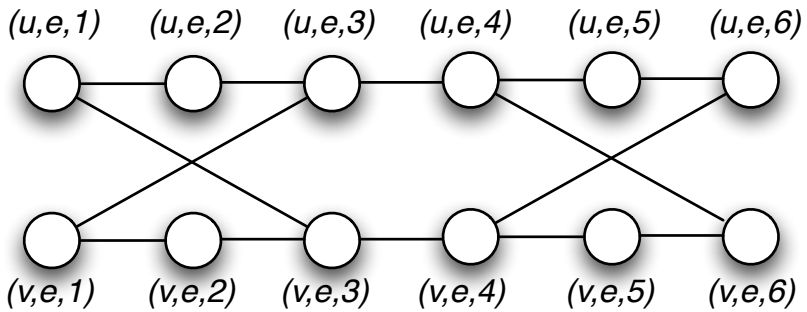
  $\Longleftrightarrow$

  le graphe G' a un cycle hamiltonien.

- Pour cela on associe un *gadget* pour représenter chaque arête de G dans G'.

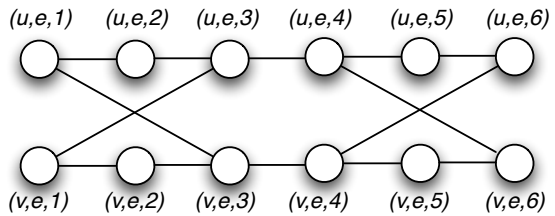
Le gadget pour une arête e

Pour chaque arête $e = (u, v)$ de G, on lui associe un *gadget* dans G' correspondant à un sous-graphe composé de 12 sommets :



La transformation de G en G'

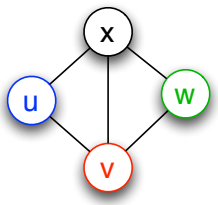
- G a k sommets notés $1, 2, \ldots, k$.
- Pour chaque arête e de G, créer un sous-graphe étant une copie du gadget.



- Pour chaque sommet u de G,
  1. Numéroter les arêtes incidentes à u : $e_1, e_2, \ldots e_d$
  2. Créer un chemin entre tous les sommets de type $(u, e_i, *)$ de la façon suivante :
     - 2.1 Relier $(u, e_i, 6)$ à $(u, e_{i+1}, 1)$ par une arête (pour i allant de 1 à $d-1$)
     - 2.2 Ajouter une arête entre $(u, e_1, 1)$ et j, et une autre arête $(u, e_d, 6)$ à j (pour j allant de 1 à k)

Un exemple

Une instance (G, k) de VC :



On peut prouver que

il existe une couverture de sommets du graphe G d'au plus k sommets

$\Longleftrightarrow$

G' a un cycle hamiltionien.

l'instance obtenue par transformation :