# Calculabilité - EISTI - ING 2

Yannick Le Nir

Ecole Internationale des Sciences du Traitement de l'Information

yannick.lenir@eisti.fr

# Problématique

## Complexité intrinsèque d'un problème

- Complexité minimale d'un algo résolvant tel problème.
- Existe-t-il un algo polynomial pour résoudre un problème donné.
- Comment-dire qu'un algorithme est optimal (en complexité).
- Comment montrer qu'un algo polynomial n'existe pas.
- Qu'est-ce qu'un problème dur.
- Comment prouver qu'un problème est aussi dur qu'un autre.

# Réductions



## Réduction peu coûteuse

Supposons que l'on sache résoudre le problème $X$ en temps exponentiel et que tout algorithme le résolvant est exponentiel. Si on arrive à réduire d'une façon "peu coûteuse", le problème $X$ dans le problème $Y$, le problème $Y$ sera lui aussi de complexité au moins exponentielle. Il reste donc à définir correctement ce qu'est une réduction "peu coûteuse".

# Problèmes praticables



## Définition

La classe $P$ (ou $PTIME$) est la classe des problèmes de décision (réponse OUI ou NON) pour lesquels il existe un algorithme de résolution polynomial en temps.

## Praticable

- Définition indépendante du modèle d'algorithme choisi (langage C, machine de Turing polynomialement équivalent), excepté les ordinateurs quantiques.
- Par convention, praticable=polynomial (abus de langage pour degré élevés du polynome).

# Classes de complexité



## Exemple de classes

- $PSPACE$ : problèmes de décision pour lesquels il existe un algorithme de résolution polynomial en espace ($PTIME \subset PSPACE$)
- $EXPTIME$ : problèmes de décision pour lesquels il existe un algorithme de résolution exponentiel en temps.

# Classe *NP*

## Généralités

*NP* contient *P* et est contenue dans *EXPTIME* et *PSPACE*.
Elle est souvent associée à des problèmes courants :

- emploi du temps
- placement de tâches
- problèmes de tournées...

## Non-déterministe Polynomial

- Conjecture $P \neq NP$ ouverte en 1971
- Institut Clay : 1 million de dollars pour sa résolution
- Chercheur partagés mais conjecture dominante

# Certificats



## Langage

Représentation d'un propriété comme le Langage de ses instances positives.

## Définition

$L$ est dit $NP$ s'il existe un polynome $Q$ et un algo polynomial à deux entrées et à valeurs booléennes tels que :
$L = \{u / \exists c, A(c,u) = Vrai, |c| \leq q(|U|)\}$.
$A$ est appelé certificat (ou preuve, ou vérification, ou témoin), facile à vérifier.

# Certificats

## Exemple

- Propriété "être satisfiable" pour une expression booléenne : certificat = valuation
- Propriété "être 3-coloriable" pour un graphe : certificat = coloriage des noeuds
- Propriété "avoir un chemin sans cycle de longueur au moins $k$ : certificat = suite de noeuds distincts
- Propriété "être composé" (i.e. non premier) : certificat = couple $(p, q)$

# Problème $P = NP$



## Problème $P = NP$

- $P \subset NP$ : algorithme de vérification = algorithme de décision (certificat = mot vide)
- $NP \subseteq PSPACE$ donc $NP \subset EXPTIME$ (algorithme qui énumère et teste tous les certificats possibles)
- Personne n'a réussi à prouver que $P \neq NP$
- Conjecture émise par S.Cook
- On conjecture également que $NP \subset PSPACE$

# $NP$ et $coNP$

## Fermeture par complément

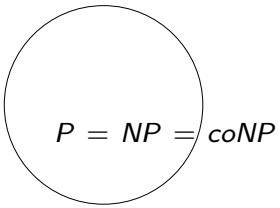
- Question ouverte : Est-ce que la classe $NP$ est fermée pour l'opération complément ?
- Est-ce que si $L \in NP$,alors $\bar{L} \in NP$ ?
- Classe $coNP$ : ensemble des langages $L$ tels que $\bar{L} \in NP$

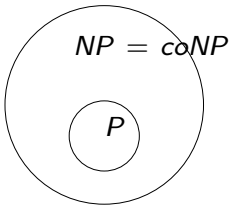
# *NP* et *coNP*

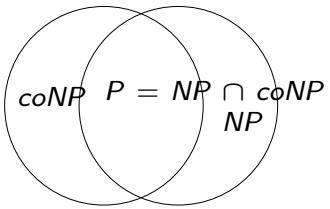
## Fermeture par complément

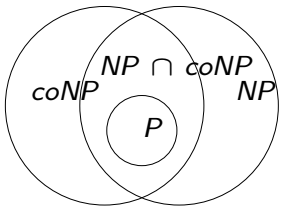
- Question ouverte : Est-ce que la classe *NP* est fermée pour l'opération complément ?
- Est-ce que si $L \in NP$,alors $\bar{L} \in NP$ ?
- Classe *coNP* : ensemble des langages $L$ tels que $\bar{L} \in NP$

4 possibilités :



# Problématique

## Equivalence des problèmes *NP*

- Conjecture $P \neq NP$ liée à l'existance de problèmes *NP*-complets
- Si un problème *NP*-complet peut être résolu en temps polynomial, alors tous les problèmes de *NP* peuvent être résolus en temps polynomial (i.e $P = NP$)
- Malgré des années de recherches, aucun algorithme polynomial n'a jamais été découvert pour quelque problème *NP*-comlet que ce soit.
- Ce sont les problèmes les plus difficiles de *NP*

# Réductions polynomiales



## Réductibilité

On ramène un problème $\mathcal{Q}$ à un autre problème $\mathcal{Q}'$ si une instance quelconque de $\mathcal{Q}$ peut être facilement reformulée comme une instance de $\mathcal{Q}'$, dont la solution fournira une solution pour l'instance de $\mathcal{Q}$.

## Exemple

La résolution d'une équation linéaire à une inconnue peut se réduire au problème de la résolution d'équations quadratiques : il suffit de transformer $ax + b = 0$ en $0x^2 + ax + b = 0$.

# Problèmes $NP-$complets

## Difficulté des problèmes

Les réductions à temps polynomial fournissent un moyen de montrer qu'un problème est au moins aussi difficile qu'un autre, à un facteur polynomial près.

## Problème de base

Il suffit dont de montrer qu'il existe un problème $NP$-complet, puis ensuite de s'y ramener par réductions polynomiales

# Quelques problèmes $NP-$complets

## Liste de problèmes $NP-$complets