Complexités exponentielles et problèmes NP
ALGORITHMIQUE - EISTI - ING 1

Yannick Le Nir
yannick.lenir@eisti.fr

Ecole Internationale des Sciences du Traitement de l'Information

Sommaire

Yannick Le Nir
yannick.lenir@eisti.fr

Cadre général

La classe P

La classe NP

Problèmes $NP-$complets

Résolution de problèmes NP

Problématique

Complexité intrinsèque d'un problème

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

Yannick Le Nir
yannick.lenir@eisti.fr

Réductions

Réduction peu coûteuse

Supposons que l'on sache résoudre le problème X en temps exponentiel et que tout algorithme le résolvant est exponentiel.

Yannick Le Nir
yannick.lenir@eisti.fr

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.

Yannick Le Nir
yannick.lenir@eisti.fr

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".

Yannick Le Nir
yannick.lenir@eisti.fr

Problèmes praticables

Définition

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

Praticable

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

Yannick Le Nir
yannick.lenir@eisti.fr

Classes de complexité

Exemple de classes

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

Yannick Le Nir
yannick.lenir@eisti.fr

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

Yannick Le Nir
yannick.lenir@eisti.fr

Certificats

Propriété

Représentation d'une propriété comme l'ensemble de ses instances positives.

Définition

L est dit NP s'il existe un polynome Q et un algorithme A 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.

Non déterminisme

Un problème NP est un problème que l'on peut vérifier avec un certificat (donc en temps polynomial).

Yannick Le Nir
yannick.lenir@eisti.fr

Certificats

Exemple

- 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)

Yannick Le Nir
yannick.lenir@eisti.fr

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*

Yannick Le Nir
yannick.lenir@eisti.fr

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 $0x^2 + ax + b = 0$.

Yannick Le Nir
yannick.lenir@eisti.fr

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.

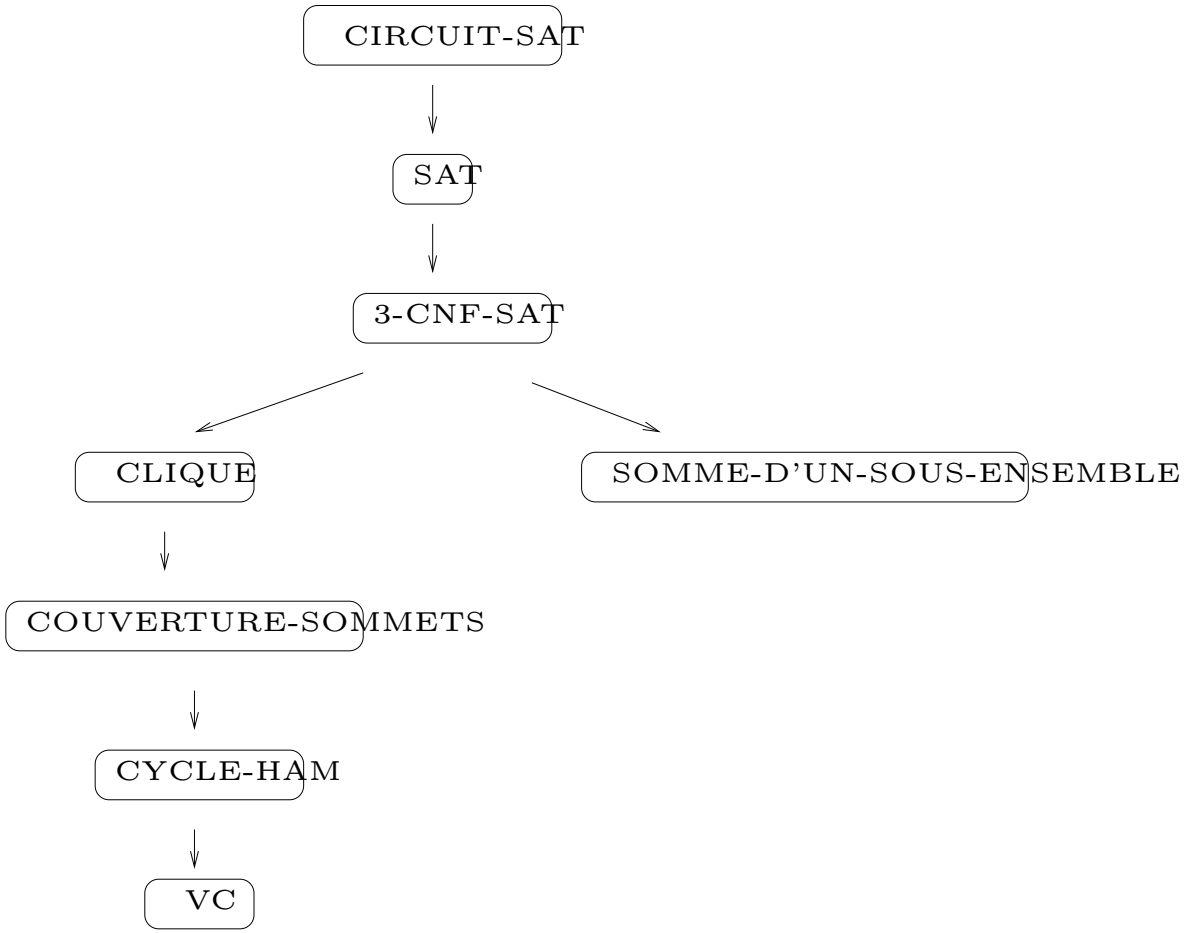
Problème de base

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

Yannick Le Nir
yannick.lenir@eisti.fr

Quelques problèmes *NP*−complets

Liste de problèmes *NP*−complets



Yannick Le Nir
yannick.lenir@eisti.fr

Comment résoudre les problèmes *NP* ?

Impossibilité d'énumérer les solutions

- algorithmes dits naïfs
- explosion combinatoire : algorithme exponentiels
- complexité en temps ou en espace trop importante

Solutions possibles

- méthodes exactes d'amélioration :
  - diviser pour régner
  - algorithmes gloutons sous certaines conditions
- méthodes heuristiques : obtention rapide (polynomiale) de solutions approchées, pas forcément optimales.

Yannick Le Nir
yannick.lenir@eisti.fr

Algorithmes gloutons

Contexte

Algorithmes souvent utilisables si la solution résulte d'un ensemble de choix

- jeux
- parcours d'un arbre ou d'un graphe
- ordonnancement de tâches
- ...

Principe

Toujours choisir le meilleur choix localement sans jamais revenir sur ce choix : ne garantit pas toujours de trouver la solution optimale

Yannick Le Nir
yannick.lenir@eisti.fr

Algorithmes heuristiques

Exemples

Algorithmes souvent issus d'applications d'intelligence artificielle (IA), qui seront étudiés en ING2 à l'EISTI :

- algorithmes de recherche tabou
- algorithmes du recuit simulé
- algorithmes génétiques
- réseaux de neurones
- ...

Yannick Le Nir
yannick.lenir@eisti.fr