
CHAPITRE 2: CLASSES ET COMPLEXITÉ D'UN PROBLÈME

COURS : COMPLEXITÉ ET OPTIMISATION

RÉALISÉ Par : DR. S. SLATNIA

UNIVERSITÉ DE BISKRA
MASTER D'INFORMATIQUE
2021-2022



CLASSES ET COMPLEXITE D'UN PROBLEME



COMPLEXITÉ D'UN PROBLÈME

- L'étude de la complexité des problèmes -computational complexity- s'attache à déterminer la complexité nécessaire d'un problème et à **classifier les problèmes** selon celle-ci, donc à répondre à des questions comme:
 - ❖ Quelle est la complexité minimale d'un algorithme résolvant tel problème?
 - ❖ Comment peut-on dire qu'un algorithme est optimal (en complexité)?
 - ❖ Existe-t-il un algorithme polynomial pour résoudre un problème donné?
 - ❖ Qu'est-ce qu'un problème "dur"?
 - ❖ Comment prouver qu'un problème est au moins aussi "dur" qu'un autre?

COMPLEXITÉ D'UN PROBLÈME

■ Définition.

La complexité d'un problème est la complexité minimale dans le pire des cas d'un algorithme qui le résout. C'est souvent la complexité en temps qu'on considère mais on peut s'intéresser à d'autres mesures comme par exemple la complexité en espace.

- Calculer la complexité d'un problème est une chose extrêmement ardue en général. On se contente souvent d'encadrer:
 - ❖ Pour trouver une *borne supérieure*, il suffit de trouver **UN** algorithme.
 - ❖ Pour trouver une "**bonne**" *borne inférieure*, les choses sont souvent plus dures... : par exemple, pour montrer par exemple qu'un problème est de complexité au moins exponentielle, il faut montrer que **TOUT** algorithme le résolvant est exponentiel.

TROIS MÉTHODES POUR TROUVER UNE BORNE INFÉRIEURE

1. Les méthodes dites d'oracle ou d'adversaire

On suppose qu'il existe un algorithme utilisant moins d'un certain nombre d'opérations d'un certain type; on construit alors une donnée qui met en défaut l'algorithme.

2. Les arbres de décision

Ils sont utilisés pour les algorithmes de type recherche ou tri "par comparaisons" : on suppose que seules des comparaisons entre les éléments sont utilisées pour obtenir de l'information sur l'ordre ou l'égalité des éléments.

TROIS MÉTHODES POUR TROUVER UNE BORNE INFÉRIEURE

- Un arbre de décision "représente" toutes les comparaisons exécutées par l'algorithme sur les données d'une certaine taille:
 - ❖ Un noeud correspond à une comparaison,
 - ❖ Ses fils aux différentes réponses possibles de la comparaison (donc si le test est à valeur booléenne, les noeuds sont binaires);
 - ❖ Une exécution possible correspond donc à une branche
 - ❖ Deux données correspondant à la même branche correspondront à la même suite d'instructions.

3. Les Réductions

Le principe est simple. On essaie de ramener un problème à un autre (dont par exemple on connaît la complexité).

CLASSES DE COMPLEXITÉ D'UN PROBLÈME

- On s'intéresse aux ressources nécessaires pour la résolution de problèmes. Les ressources essentielles sont le temps et l'espace. L'idée est de classer les problèmes selon l'ordre de grandeur de leur complexité.
- On ne considère ici que des problèmes décidables. On suppose que toutes les machines de Turing considérées s'arrêtent toujours. On définit le temps et espace de calcul relativement aux machines de Turing.

CLASSES DE COMPLEXITÉ D'UN PROBLÈME

□ Classes de complexité en temps

❖ La classe P

Définition 1. La classe des problèmes qui peuvent être résolus en temps **polynomial** par une *machine déterministe*. $P = \bigcup_{k \geq 0} \mathbf{TIME}(n^k)$

Définition 2. La classe P (ou **P****TIME**) est la classe des problèmes de décision pour lesquels il existe un algorithme de résolution **polynomial** en temps.

Cette définition est indépendante du modèle d'algorithme choisi: un algorithme polynomial si il est traduit en termes de machine de Turing et les modèles classiques de calcul sont polynomialement équivalents.

CLASSES DE COMPLEXITÉ D'UN PROBLÈME

□ Classes de complexité en temps

❖ La classe P

Définition 3. P est la classe des problèmes praticables

Un problème de décision est dit praticable si il est dans P, impraticable sinon;

”existe-t-il un algorithme praticable pour ce problème?” se ramène à ”ce problème est-il dans P?”

CLASSES DE COMPLEXITÉ D'UN PROBLÈME

□ Classes de complexité en temps

❖ La classe NP

Définition 1. Classe des problèmes qui peuvent être résolus en temps **polynomial** par *une machine non déterministe*. $\mathbf{NP} = \bigcup_{k \geq 0} \mathbf{NTIME}(n^k)$

Définition 2. La classe **NP** contient énormément de propriétés associées à des problèmes courants qui apparaissent dans de multiples domaines: problèmes de tournées, d'emploi du temps, placement de tâches, affectation de fréquences, découpage, ...

CLASSES DE COMPLEXITÉ D'UN PROBLÈME

□ Classes de complexité en temps

❖ La classe **EXP**

Définition 1. Classe des problèmes qui peuvent être résolus en temps **exponentiel** par *une machine déterministe*. $\mathbf{EXP} = \bigcup_{k \geq 0} \mathbf{TIME}(2^{nk})$

Définition 2. **EXPTIME**, la classe des problèmes de décision pour lesquels il existe un algorithme de résolution exponentiel en temps.

CLASSES DE COMPLEXITÉ D'UN PROBLÈME

□ Classes de complexité en espace

PSPACE: la classe des problèmes de décision pour lesquels il existe un algorithme de résolution polynomial en espace. Bien sur PTIME est inclus dans PSPACE: un algo polynomial en temps "consomme au plus un espace polynomial".

CLASSES DE COMPLEXITÉ D'UN PROBLÈME

▪ La classe **NP** contient **P** et est contenue dans **ExpTime** (et même dans **Pspace**). Pour beaucoup de propriétés NP, on n'a pas trouvé d'algorithme polynomial, mais pour aucune d'entre elles, on n'a prouvé qu'elle ne pouvait pas être décidée en temps polynomial.

$$\diamond \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{EXP}$$

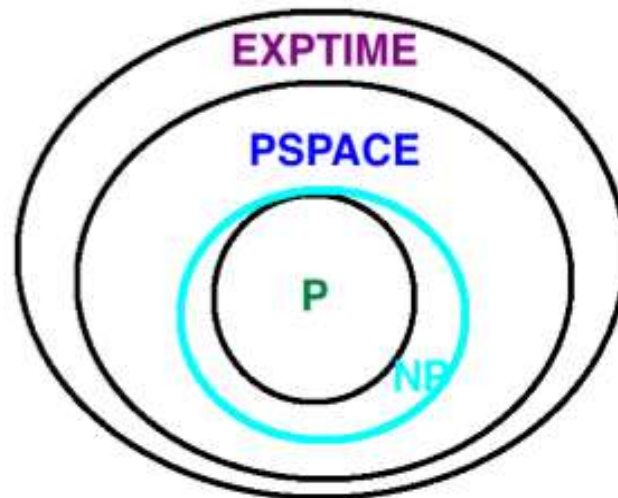


Figure. Classes de complexité d'un problème

NP-COMPLÉTUDE

- Un problème est simple quand le nombre d'étapes nécessaires à sa résolution :
 - ❖ Borné par un polynôme = problème **polynomial** (P).
 - ❖ problème non-polynomial = **Difficile (NP)**.
- De petites variations d'un problème peuvent être suffisantes pour passer d'un problème P à un problème NP :
 - ❖ Partitionner un ensemble en 2 sous-ensembles de même cardinalité, telle que la différence des sommes de ces sous-ensembles soit :
 - maximale
 - minimale

➤ Il est facile de couper un ensemble en deux de façon à avoir cette différence maximale.

Exemple.

❖ On peut trier l'ensemble, et prendre comme parties les éléments de la fin et les éléments du début.

→ Obtenir une *différence minimale* est un *problème combinatoire*.

□ Il existe de nombreux problèmes NP (combinatoire) = Algm difficile :

❖ Problème du voyageur de commerce (Travelling Salesman)

Y a-t-il un cycle qui visite tous les noeuds d'un graphe, et dont le poids soit au plus K ? Minimiser le trajet pour voir les clients

❖ Factorisation

On veut décomposer un grand nombre en nombres premiers.

❖ Coloration de graphes

On veut attribuer à chaque sommet une couleur sans que deux sommets reliés soient de la même couleur.

- Le nombre minimal de couleurs requises est dit 'nombre chromatique' et noté $\chi(G)$.
- Déterminer $\chi(G)$ est NP-Complet dans le cas général.

□ Le problème qui se pose maintenant est de voir lorsque deux problèmes sont du même niveau de difficulté.

□ **L'outil de base pour établir des relations entre les complexités** de différents problèmes est la ***Réduction***.

RÉDUCTION D'UN PROBLÈME Q À UN PROBLÈME Π

- Soit X une donnée d'entrée pour le problème Q , dont on veut une réponse 'oui' ou 'non'.
- On transforme X en une donnée d'entrée Y pour le problème Π avec un algorithme A .
- La réponse donnée au problème Π avec la donnée Y doit être la même que celle donnée au problème Q avec la donnée Y .

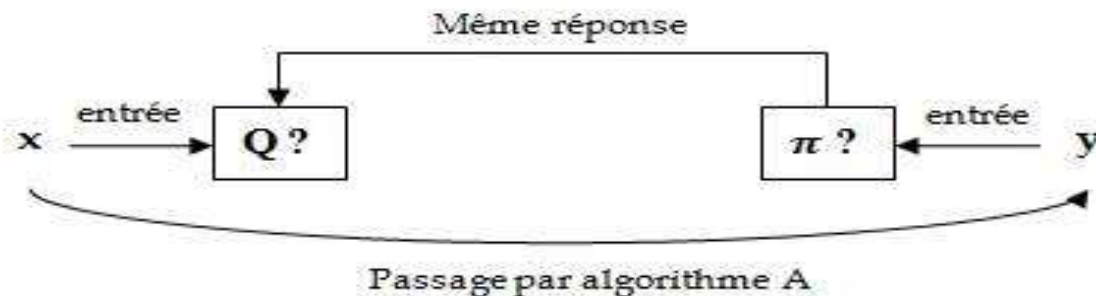


Figure. Réduction d'un problème Q à un problème Π