Calculabilité, Complexité,

3 – Complexité, vers la NP-complétude

P. Berthomé

INSA Centre Val de Loire
Département STI

5 octobre 2020

Introduction

Objectifs

- Évaluer l'efficacité d'une MT
- Connaître les grandes classes de complexité
- Savoir ce que recouvre la NP-complétude
- Avoir une idée des preuves dans le cas des MT

Complexités avec Machines de Turing

Contexte

- Afin de comparer des complexités,
- il faut utiliser les mêmes modèles
- MT avec un ruban semi-infini, 1 RWH

Complexité en espace

- $TM_{\text{space}}(x)$: Nombre de cases visitées sur l'entrée x
- Cases visitées **plus éventuellement** l'entrée

Complexité en temps

- $TM_{\text{time}}(x)$: temps passé
- Nombre d'étapes **plus** la longueur de l'entrée

Propriétés générales

Theorem

Les fonctions TM_{space} et TM_{time} sont des fonctions semi-définies

Preuve

Preuve directe : il suffit de construire une MT auxiliaire qui espionne ce que fait la MT

Theorem

Quand les deux fonctions sont définies, on a :

1. $\forall x \quad TM_{space}(x) \leq TM_{time}(x)$
2. $\exists k_T \geq 0 \ TM_{time}(x) \leq k_T^{TM_{space}(x)}$

Preuves des propriétés

Propriété 1

En plus de l'entrée, il faut parcourir chaque case écrite

Propriété 2

- La machine de Turing s'arrête :
  - les deux notions sont définies
  - On ne boucle pas à l'infini
  - On ne revoit pas deux fois la même configuration
- Combien de configurations différentes possibles ?
  - $M = TM_{\text{space}}(x)$ cases utilisées, pour chaque :
  - $|\Sigma|$ lettres possibles
  - $M \times |Q|$ positions de la RWH et d'états
- $TM_{\text{time}}(x) \leq |\Sigma|^M \times M \times |Q|$
- $K_T = 2 \times |\Sigma| \times |Q|$

Ordres de grandeur

Distinction des complexités

- n : efficace
- n^p : polynomial. OK en théorie, mais en pratique, il faut que p soit petit
- p^n : exponentiel, pas utilisable en pratique
- au delà, c'est intéressant seulement de manière théorique.

Autres modèles

Programmes RAM

Définition de la complexité en temps et en espace similaire.

Circuits booléens

- Circuits logiques à l'aide des portes logiques **ET**, **OU**, **XOR** et **NON**
- **Complexité en espace** : nombre de portes logiques pour réaliser le circuit
- Exemple : addition k bits en $6k - 4$ portes.
- **Complexité en temps** : profondeur du circuit
- Exemple : addition k bits en $2k + 1$ étapes (retenue).

Complexité d'un problème

Définition

Soit $\mathcal{P}$ un problème. Soit $\mathcal{M} = \{MT$ qui résout ce problème$\}$. La complexité du problème $\mathcal{P}$ est le meilleur temps mis par une machine dans $\mathcal{M}$ pour résoudre ce problème

Polynomial ou Exponentiel ?

Cette complexité $f(n)$ est

Polynomiale : $\exists\, m, \alpha \;\; f(n) \leq \alpha n^m$

Exponentielle : $\exists\, m, \alpha \;\; f(n) \leq \alpha m^n$

Machines de Turing Non déterministes

Définition

Une machine de Turing est non déterministe si la fonction de transition est de la forme :

$$\Delta \subseteq Q \times \Sigma^{||m||} \times Q \times \Sigma^{||m||} \times \{G, St, D\}^{||m||}$$

En d'autres termes, il y a potentiellement plusieurs (≥ 0) transitions possibles pour un état et un ensemble de lettres lues par les RWH.

Reconnaissance par une MT non déterministe

Un mot x est reconnu par une MT non déterministe s'il existe une exécution de la machine qui arrive sur un état d'acceptation.

Machines de Turing Non déterministes

Temps de calcul d'une MT non déterministe

Comme pour une MT déterministe, il faut compter le meilleur temps pour arriver à un état d'acceptation en fonction de la taille des entrées.

Équivalence

- Soit $\mathcal{L}$ un langage reconnu par une MT non déterministe T.
- Il existe une MT T' déterministe qui reconnaît le même langage
- Il existe un surcoût de simulation exponentiel

Classes de problèmes

Classe P

La classe P est l'ensemble des langages qui sont reconnus en temps **polynomial** par une machine de Turing **déterministe**

Classe NP

- La classe NP est l'ensemble des langages qui sont reconnus en temps **polynomial** par une machine de Turing **non déterministe**
- **Ce n'est pas** non polynomial
- Un des sept problèmes du Clay Institute à résoudre : $P = NP$?

Transformation polynomiale

Définition

- Soit L et L_0 deux langages (potentiellement sur deux alphabets différents).
- L est **polynomialement transformable** en L_0 si :
  - Il existe une machine de Turing T
  - qui convertit tout mot ω en un mot ω_0
  - en temps polynomial (fonction de la longueur de ω)
  - tq :

$$\omega \in L \iff \omega_0 \in L_0$$

Problème ou Langage NP-Complet

Définition

Un langage L est NP-Complet si et seulement si :

1. $L \in NP$
2. tout langage de NP est polynomialement transformable en L

Discussion

- Supposons qu'il existe un langage NP-Complet L_0 et l'on prouve que $L_0 \in P$ alors $P = NP$
- A priori, si on a montré qu'un problème est NP-Complet, il est inutile (illusoire) de chercher un algorithme performant (polynomial)

Au fait, on en a vu ?

Problème SAT

Entrées : $x_1, \ldots, x_n$ des variables booléennes et φ une formule booléenne contenant ces variables. La longueur de φ est polynomiale en n

Question : Existe-t-il une affectation des variables telle que φ soit vraie ?

Théorème de Cook-Levin (1971)

Le problème SAT est NP-Complet

Un petit bout de preuve

SAT est dans NP

- On veut trouver une MT non déterministe qui répond à la question en temps polynomial.
- On sépare la MT en deux phases polynomiales
  1. Déterminer les valeurs des variables
  2. Vérifier que cela fonctionne

Vérification

- Entrées : la formule sur un ruban et les variables avec leur affectation sur l'autre
- vérification déterministe de la formule

Choix non déterministe

Affectation des variables (Oracle)

- Entrée : un ruban avec le nombre de variables
- On construit la MT qui prépare le ruban *Variable* pour la machine de vérification
- Algorithme
  - Pour i allant de 1 à n
  - Définir de manière **non déterministe** la valeur de x_i
  - Écrire sur le ruban le codage de x_i prend la valeur définie

Au final

- L'exécution de la MT prend un temps polynomial
- S'il existe une bonne affectation (φ est satisfiable)
- il existe une exécution en temps polynomial qui fonctionne
- Donc SAT est dans NP

Idées complémentaires

Principes généraux

- Un peu plus de détails sur Celene
- Codage d'une MT non déterministe, des descriptions instantanées et du fonctionnement par une formule logique
- Les variables représentent les états à chaque instant (est-ce que la MT est dans l'état à l'instant t, ...)
- Le temps d'exécution des MT est polynomial (hypothèse)
- donc la formule est de taille polynomiale

Quelques autres problèmes NP-Complets

Suite de réductions

- SAT vers CNF-SAT : Satisfiabilité des formules booléennes sous forme conjonctive normale
- CNF-SAT vers 3-CNF-SAT : Satisfiabilité des formules booléennes sous forme conjonctive normale où chaque terme contient au plus 2 signes **OU**
- 3-CNF-SAT vers colorabilité des graphes
- SAT vers Clique Maximum
- Clique Maximum vers circuits Hamiltoniens
- ...