Classe P et NP. Problèmes NP-complets. Exemples. [TOR 13]

Motivation Les problèmes décidables sont-ils « simples » à résoudre ? On apprend ici une manière de classer les problèmes en fonction de leur « dureté ».

Remarque 1 Dans la suite, on appellera « algorithme » un programme Ocaml ou C qui termine.

I. Théorie de la complexité

A. Problèmes de décision [TOR 13.1 13.2]

Définition 2 **Un problème de décision** sur un domaine d'entrée E est défini par une fonction totale $f : E \to \mathbb{B} = \{V, F\}$.

Exemple 3 **SAT** est un problème de décision avec comme domaine d'entrée l'ensemble des formules propositionnelles $\mathcal{F}_P$ et comme fonction sat tel que $\text{sat}(\varphi \in \mathcal{F}_P) = V \Leftrightarrow \varphi$ est satisfiable.

Définition 4 **Un problème de recherche** est défini par une relation binaire $R \subseteq E \times S$. Un algorithme A résoud ce problème de recherche R si pour toute entrée $e \in E, A$ appliqué à e produit une sortie $s \in S$ tel que $R(e, s)$ lorsqu'une telle solution existe.

Exemple 5 **Trouver un chemin dans un graphe** est un problème de recherche avec comme domaine des solutions les séquences de sommets.

Exemple 6 **La coloration d'un graphe** g est un problème de recherche avec comme domaine des solutions les fonctions partielles des sommets vers les couleurs.

Définition 7 **Un problème de vérification** associé à un problème de recherche $R \subseteq E \times S$ est le problème de décision ayant pour domaine d'entrées $E \times S$ et comme fonction : $f_V : E \times S \to \mathbb{B}$ et $f_V(e, s) = V \Leftrightarrow R(e, s)$.

Définition 8 **Un problème d'optimisation** , définit par $R \subseteq E \times S$ et $c : S \to \mathbb{R}^+$, détermine, étant donnée un $e \in E$, un $s \in S$ tel que $c(s)$ minimal.

Exemple 9 Le problème de trouver un plus court chemin dans un graphe ou trouver un nombre minimum de couleur pour colorer un graphe sont des problèmes d'optimisation.

Définition 10 **Un problème de seuil** associé à un problème d'optimisation est un problème de décision définit par $R \subseteq E \times S$, $c : S \to \mathbb{R}^+$ et c_0 avec $f_{c_0}(e \in E) = V \Leftrightarrow \exists s \in S, R(e, s), c(s) \leq c_0$.

Définition 11 **Un chemin hamiltonien** dans un graphe non-orienté est un chemin passant par chaque chemin du graphe exactement une fois.

DEV

Exemple 12 **Le problème du voyageur de commerce** (ou TSP pour Traveling Salesman Problem) est un problème d'optimisation recherchant un chemin hamiltonien dans graphe complet pondéré (poids positifs) non orienté de poids minimum.

B. Complexité [TOR 6.3]

Définition 13 **la complexité temporelle** d'une éxécution d'un programme sur une entrée donnée mesure le nombre d'opérations atomiques réalisées.

Remarque 14 On étudiera uniquement des domaine d'entrée infinis.

Remarque 15 On étudiera l'ordre de grandeur asympotique de la complexité en fonction de la taille de l'entrée souvent noté n.

Remarque 16 **La taille d'un entier** de valeur maximal n est en $O(\log(n))$, quel que soit la base utilisée.

Exemple 17 **Le test de primalité** de n se fait au mieux en $O(\sqrt{n})$ étapes (admis) ce qui est exponentiel en la taille de l'entrée $|e| = O(\log(n))$.

II. P et NP

A. Classe P [TOR 13.2.3]

Définition 18 **La classe P** est l'ensemble des problèmes de décision qui admettent une solution dont la complexité temporelle est majorée asymptotiquement par un polynôme en la taille de l'entrée.

Remarque 19 Le test de primalité est un problème de décision dans P. Ce fut montré à l'aide de algorithme de AKS découvert en 2002.

Remarque 20 **Un algorithme galactique** est un algorithme de complexité meilleur que d'autres algorithmes pour des entrées de taille immense. Un tel algorithme existe pour la multiplication d'entiers.

Exemple 21 **La déterminisation d'un automate** est un problème algorithmique sans algorithme solution en complexité polynomiale. En effet l'automate déterminisé peut être de taille exponentielle en la taille de l'entrée.

B. Reduction Polynomiale [TOR 13.2.4]

Définition 22 Soient deux problèmes de décision définies par $f_1 : E_1 \to \mathbb{B}$ et $f_2 : E_2 \to \mathbb{B}$. f_1 se **réduit polynomialement** à f_2 s'il existe une fonction $g : E_1 \to E_2$ de complexité temporelle polynomiale tel que $\forall e \in E_1, f_1(e) = f_2(g(e))$ et on note $f_1 \underset{P}{\leq} f_2$.

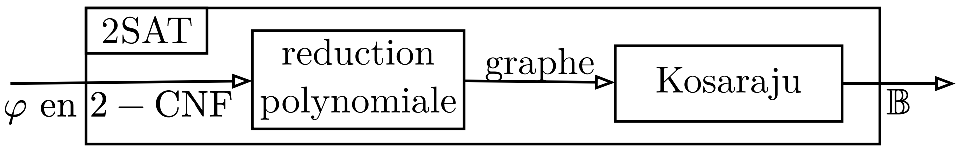
Définition 23 Soient deux problèmes de décisions décrit par f_1 et f_2. Si f_2 appartient à P et si $f_1 \underset{P}{\leq} f_2$ alors f_1 appartient à P.

Exemple 24 **L'algorithme de Kosaraju** est dans P et permet de calculer les composantes fortement connexes dans un graphe.

Exemple 25 **2SAT** est le problème de décision restreint de SAT avec des formules en 2-FNC.

Remarque 26 On peut réduire de façon polynomiale 2SAT au problème de calcul de composantes fortement connexe dans un graphe, d'où 2SAT est dans P.

Schémas 27 **Reduction de 2SAT à Kosaraju.**



C. Classe NP [TOR 13.2.5 13.3]

Remarque 28 La classe NP décrit des problèmes de décision liées à des problèmes de recherche dont le problème de vérification associé est dans P.

Définition 29 **La classe NP** est l'ensemble des $f : E \to \mathbb{B}$ tel que :

- $\exists C, g : E \times C \to \mathbb{B} \mid (f(e) = V \Leftrightarrow \exists c \in C, g(e,c) = V)$ avec c de taille polynomiale en la taille de e.
- le problème de décision défini par g, appelé problème de vérification d'un certificat est dans la classe P.

Exemple 30 **Le problème SAT** est dans NP.

Définition 31 **Un problème NP-difficile** est un problème algorithmique auquel peut être réduit polynomialement tout problème de NP.

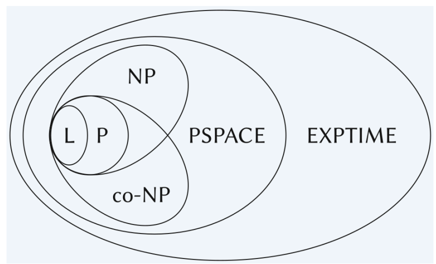
Définition 32 **Un problème NP-complet** est un problème NP-difficile et appartenant à NP.

Théorème 33 **de Cook Levin.** Le problème SAT est NP-Complet.

Exemple 34 **Le problème 3SAT** est NP-complet.

Remarque 35 **Relation entre P et NP.** P est inclu dans NP cependant l'inclusion dans l'autre sens (NP $\subset P$) est un problème encore ouvert en recherche informatique.

Remarque 36 **Hierarchie des classes de complexités.** [TOR 13.2]



III. Que faire en cas de NP-dureté ?

A. Backtracking

Remarque 37 Certains problèmes NP complet peuvent être résolus en temps exponentiel en itérant sur toutes les possibilités.

Idée 38 **Le backtracking** consiste à énumérer toutes ces solutions pour les tester. De nombreuses méthodes permettent d'accélérer cette recherche en pratique.

Remarque 39 **Le Sudoku généralisée** sur un tableau de taille quelconque n peut être résolue de façon efficace par des méthodes de « branch and bound » bien que le problème soit NP-complet (admis). Une méthode pour le résoudre consiste à réduire ce problème à SAT. DEV

B. Contraintes sur un problème

Idée 40 En limitant un problème NP-difficile, on arrive parfois à un problème plus simple à résoudre dans P.

Exemple 41 **2SAT** est une simplification de SAT dans P.

Exemple 42 **Bicolorer un graphe** est une simplification de la k-coloration d'un graphe. Trouver si un graphe peut être 2-coloré est un problème dans P.

C. Problème d'Approximation

Idée 43 Pour certains problèmes NP-difficile on peut trouver des approximations de la solution. Le problème d'approximation associé peut se trouver dans P.

Définition 44 **Une ρ-approximation** d'un problème d'optimisation est un algorithme qui renvoie des solutions au plus ρ fois plus coûteuse que la solution optimale.

Exemple 45 **Une log(n)-Approximation** du problème de couverture de sommet de taille minimale est atteinte par un algorithme glouton polynomial.

Exemple 46 **TSP métrique** est une variation du problème du voyageur du commerce où la fonction de pondération w du graphe G respecte l'inégalité triangulaire :

$$\forall s, t, u \in G, w(s, u) \leq w(s, t) + w(t, u)$$

Exemple 47 TSP métrique est NP-complet et 2-approximable en temps polynomial.

IV. Modèle de Calcul - Machine de Turing [TOR 13.5.2]

Idée 48 On souhaite définir formellement la notion d'algorithme pour nous assurer que les classes P et NP ne dépendent pas du langage utilisé.

Définition 49 **Une machine de Turing** est un modèle de calcul décrit par un ensemble fini de règles de transitions qui, en fonction d'un état courant de la machine et de l'information lue à la position courante d'un ruban de taille infinie donne :

- une intruction d'écriture sur la case courante du ruban
- une instruction de déplacement sur le ruban
- le nouvel état de la machine après cette étape

Remarque 50 Ce modèle de calcul a beaucoup inspiré le paradigme impératif et la conception des processeurs.

Définition 51 **Une machine universelle** existe et peut lancer l'éxécution d'une machine de Turing sur une entrée donnée. Turing introduit ici donc la notion « d'interpréteur ».

Remarque 52 Une machine de Turing est proche de la notion d'automate aux détails près qu'elle modifie à la volée le mot pris en entrée.

Remarque 53 On peut fixer des états finaux soit acceptant soit non acceptant dans une machine de Turing pour lui permettre de répondre à des problèmes de décision.

Définition 54 **Les machines de Turing non déterministes** peuvent être définit, à l'image des automates, avec des transitions non déterministes.

Propriété 55 L'ensemble des problèmes de décisions décidé par une machine de Turing non déterministes en temps polynomiale est égale à la classe NP précédemment définit.

Remarque 56 Sans pertes de généralités on peut assimiler une machine de Turing à un algorithme écrit en C ou Ocaml.

Classe P et NP. Problèmes NP-complets. Exemples. [TOR 13]

1 Rem

I. Théorie de la complexité

A. Problèmes de décision [TOR 13.1 13.2]

2 Def Un problème de décision
3 Ex SAT
4 Def Un problème de recherche
5 Ex Trouver un chemin dans un graphe
6 Ex La coloration d'un graphe
7 Def Un problème de vérification
8 Def Un problème d'optimisation
9 Ex
10 Def Un problème de seuil
11 Def Un chemin hamiltonien
12 Ex Le problème du voyageur de commerce

B. Complexité [TOR 6.3]

13 Def la complexité temporelle
14 Rem
15 Rem
16 Rem La taille d'un entier
17 Ex Le test de primalité

II. P et NP

A. Classe P [TOR 13.2.3]

18 Def La classe P
19 Rem
20 Rem Un algorithme galactique
21 Ex La déterminisation d'un automate

B. Reduction Polynomiale [TOR 13.2.4]

22 Def
23 Def
24 Ex L'algorithme de Kosaraju
25 Ex 2SAT
26 Rem
27 Schémas Reduction de 2SAT à Kosaraju.

C. Classe NP [TOR 13.2.5 13.3]

28 Rem
29 Def La classe NP
30 Ex Le problème SAT
31 Def Un problème NP-difficile
32 Def Un problème NP-complet
33 Thm de Cook Levin.
34 Ex Le problème 3SAT
35 Rem Relation entre P et NP.
36 Rem Hierarchie des classes de complexités. [TOR 13.2]

III. Que faire en cas de NP-dureté ?

A. Backtracking

37 Rem
38 Idée Le backtracking
39 Rem Le Sudoku généralisée

B. Contraintes sur un problème

40 Idée
41 Ex 2SAT
42 Ex Bicolorer un graphe

C. Problème d'Approximation

43 Idée
44 Def Une ρ-approximation
45 Ex Une log(n)-Approximation
46 Ex TSP métrique
47 Ex

IV. Modèle de Calcul - Machine de Turing [TOR 13.5.2]

48 Idée
49 Def Une machine de Turing
50 Rem
51 Def Une machine universelle
52 Rem
53 Rem
54 Def Les machines de Turing non déterministes
55 Prop
56 Rem

Remarque

Programme MPI La notion de machine de Turing est hors programme. On s'en tient à une présentation intuitive du modèle de calcul (code exécuté avec une machine à mémoire infinie). On insiste sur le fait que la classe P concerne des problèmes de décision.

- connaître la vraie classe de complexité de accessibilité dans un graphe.
- bien connaitre tous les réductions présentées dans le plan.
- présenter un algo d'approx
- présentér un problème NP qui devient P avec des solutions
- Donner des problèmes dans différentes branches de l'informatique pour diversifier :
  - logique : SAT, 2SAT ...
  - graphe : Accessibilité dans un graphe
  - entier : subset sum, entier est premier
  - théorie des languages par séparation par automates. 2 languages finis, existent ils un automate déterministe (au plus k états) qui acceptent tous les mots du premier et refusent tous les mots du second.
  - programmation linéaire. Algo du simplexe != Algo de l'ellypsoïde.

Partie Machine de Turing à compléter

Si on a plus de place des choses peuvent être ajouter :
- équivalence avec biruban et un seul ruban.
- définition machine de Turing Universel.

Bonus TSP

En fonction de si TSP est un dev ou non :

Exemple 57 En supposant P $\neq$ NP, le problème du voyageur de commerce dans le cas générak n'est pas ρ-approximable en temps polynomial pour tous ρ constant.

Developpements Possibles

- 3 coloration à partir de 3 SAT, Kosaraju - 2SAT, Sudoku, TSP

Bibliographie

[TOR] T. Balabonski & S. Conchon & J. Filliâtre & K. Nguyen & L. Sartre, *MP2I MPI, Informatique Cours et exercices corrigés*.

Paul Adam & Axel Stengel