Calculabilité et Complexité (11X008) - Printemps 2025

8. Classe NP

Enseignant: Arnaud Casteigts

Assistants: A. Berger, M. De Francesco
Moniteurs: E. Bussod, N. Beghdadi

8.1 Définitions et un exemple

Et voici nos deux célébrités :

- P : Langages décidables en temps $n^{O(1)}$ par une machine *déterministe*.
- NP : Langages décidables en temps $n^{O(1)}$ par une machine *non-déterministe*.

La classe NP admet une autre définition équivalente :

- NP : Langages qui admettent des *certificats positifs* vérifiables en temps $n^{O(1)}$ par une machine *déterministe*.

Un certificat positif est une preuve d'appartenance d'un mot au langage. Autrement dit, une preuve qu'un mot w est bien dans le langage L considéré. Ce certificat est potentiellement différent pour chaque mot w de L.

Prenons pour exemple le problème de décision / langage suivant :
3-Coloring
Entrée : Un graphe G
Question : Ce graphe peut-il être colorié avec 3 couleurs de sorte qu'aucun sommet n'a la même couleur que ses voisins ?

Si un graphe G est 3-colorable (c.à.d. si $G \in$ 3-Coloring), alors il existe une preuve facile à *vérifier* : la coloration elle-même. En effet, si l'on nous donne cette coloration, on peut vérifier facilement que G est 3-colorable. Le problème 3-Coloring admet donc des certificats positifs vérifiables en temps polynomial, il est donc bien dans NP. (Notez que cette coloration pourrait être difficile à *trouver*, mais ce n'est pas la question ici.)

Plus bas, nous expliquerons aussi comment 3-Coloring peut être résolu par une machine non-déterministe (montrant donc que ce langage est dans NP selon la première définition).

8.2 Relations

Qu'en est-il de la relation entre P et NP ? Comme déjà évoqué, une machine déterministe est un cas particulier de machine non-déterministe. Tous les langages de P sont donc aussi dans NP ($\mathsf{P} \subseteq \mathsf{NP}$). En utilisant plutôt la deuxième définition de NP, on pourrait simplement observer qu'un algorithme qui décide un langage en temps polynomial permet aussi de vérifier l'appartenance avec un certificat vide (donc $\mathsf{P} \subseteq \mathsf{NP}$ à nouveau).

L'une des questions les plus fondamentales en informatique et en mathématiques est de savoir si $\mathsf{P} = \mathsf{NP}$. Pourquoi en mathématiques ? En simplifiant un peu, on peut définir le langage L_{MATH} de tous les énoncés mathématiques qui sont vrais et qui ont des preuves de taille raisonnables. Par définition, ce langage est dans NP. Si $\mathsf{P} = \mathsf{NP}$, cela implique que tous ces énoncés peuvent être décidés rapidement par un unique algorithme (on ne gagne donc pas seulement 1 million, mais 6... :-)). La majorité des spécialistes pensent que $\mathsf{P} \neq \mathsf{NP}$. Notez que la question P *vs.* NP est elle-même un énoncé mathématique... (vertige ?)

Que peut-on dire d'autre sur NP ? On a d'un côté $\mathsf{NP} \subseteq \mathsf{NPSPACE}$ (même preuve que $\mathsf{P} \in \mathsf{PSPACE}$) et de l'autre $\mathsf{NPSPACE} = \mathsf{PSPACE}$ (via le théorème de Savitch). On peut donc encadrer NP comme suit :

$$\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE}$$

Et comme d'habitude, on ne sait pas si ces inclusions sont strictes...

8.3 Exemples de problèmes dans NP

La classe NP est importante car de nombreux problèmes de la vie courante sont dans cette classe. Voici quelques problèmes dans NP pour lesquels on ne connaît pas d'algorithme polynomial (donc, potentiellement, ces problèmes sont dans $\mathsf{NP} \setminus \mathsf{P}$) :

Plutôt que de définir les langages eux-mêmes, nous les formulons sous forme de problèmes de décision (c'est équivalent).

- 3-Coloring (déjà vu)
- Clique : Étant donné un graphe G et un entier k, est-ce que G contient k sommets qui sont tous voisins ?
- Independent Set : Étant donné un graphe G et un entier k, est-ce que G contient k sommets qui ne sont pas voisins ?
- Subset Sum : Étant donné une liste de nombre, peut-on en sélectionner certains pour obtenir une somme à 0 ?
- Graph Isomorphism : Étant donnés deux graphes G_1 et G_2, est-ce que ces graphes sont structurellement identiques ?

- SUBGRAPH ISOMORPHISM : Étant donnés deux graphes G_1 et G_2, est-ce que G_1 contient G_2 comme sous-graphe ?
- FACTORING : Étant un entier N et un entier k, N admet-il un facteur $\leq k$?
- HAMILTONIAN CYCLE : Étant donné un graphe G, existe-t-il un chemin qui visite tous les sommets exactement une fois, puis termine au point de départ ?
- SAT : Étant donné une formule booléenne ϕ sous forme normale conjonctive, par exemple $\phi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee \overline{x_3})$ (où $\overline{x_i}$ est la négation de x_i), existe-t-il des valeurs `vrai`/`faux` pour chaque variable x_i telles que la formule est satisfaite ?

Exercice : pour chacun de ces problèmes, réfléchir au fait qu'il admet des certificats positifs.

8.3.1 Équivalence entre les deux définitions de NP

Il n'est pas évident à première vue que les deux définitions de NP sont équivalentes. Pour s'échauffer, voyons comment le problème 3-COLORING peut être résolu avec une machine non-déterministe. Notre machine va simplement choisir parmi les 3 couleurs possibles pour le premier sommet, ce qui créée 3 branches d'exécution. Dans chaque branche, la machine choisit alors de manière non-déterministe la couleur du second sommet, ce qui créée trois nouvelles branches dans chaque branche, *etc.* Une fois que tous les noeuds sont coloriés dans une branche, cette branche vérifie que la coloration est valide, et si elle l'est, la branche accepte (et donc la machine accepte).

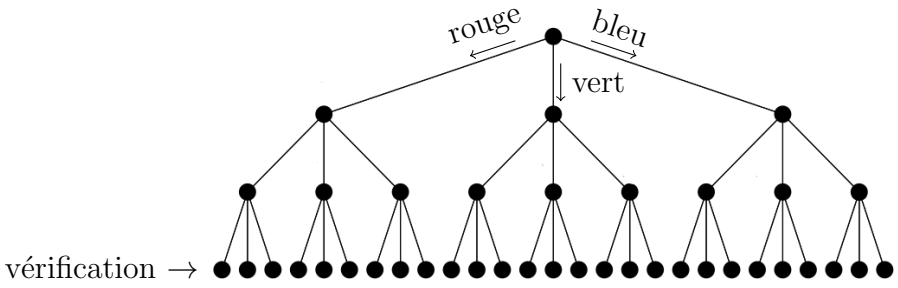

FIGURE 1 – Exploration non-déterministe d'une 3-coloration.

Cet exemple nous montre qu'une machine non-déterministe peut *deviner* le certificat, la seule difficulté étant ensuite de le vérifier. Plus précisément, nous allons montrer que les deux définitions de NP sont équivalentes en utilisant des arguments similaires.

Notons NP_1 la définition de NP basée sur le non-déterminisme et NP_2 la définition de NP basée sur l'existence de certificats positifs vérifiable rapidement (les noms NP_1 et NP_2 ont été inventés, ne les cherchez pas ailleurs).

Theorème 8.1. $NP_1 = NP_2$

Preuve. Nous allons montrer que $\mathsf{NP}_2 \subseteq \mathsf{NP}_1$ et $\mathsf{NP}_1 \subseteq \mathsf{NP}_2$.

- $\mathsf{NP}_2 \subseteq \mathsf{NP}_1$: Soit L un langage dans NP_2. Par définition, pour tout $w \in L$, il existe un certificat positif que w appartient bien à L, vérifiable en temps polynomial (par exemple, la coloration pour 3-Coloring). Ce certificat admet une description $\langle C \rangle$, codée en binaire. Une machine non-déterministe peut *deviner* un à un les bits de $\langle C \rangle$ et ensuite vérifier ce certificat. Donc $L \in \mathsf{NP}_1$.
- $\mathsf{NP}_1 \subseteq \mathsf{NP}_2$: Soit L un langage dans NP_1. Par définition, il existe une machine non-déterministe M qui décide L en temps polynomial. Autrement dit, pour tout $w \in L$, il existe un chemin d'exécution de longueur polynomiale qui mène la machine M à accepter. Dans ce cas, il existe aussi un vérifieur déterministe connaissant $\langle M \rangle$ qui, étant donné la suite de choix correspondant à ce chemin (le certificat), peut vérifier que ce chemin mène bien M à accepter w (il suffit de simuler M en effectuant seulement les choix indiqués et vérifier que la configuration sur laquelle on arrive est acceptante). On a donc $L \in \mathsf{NP}_2$.

□

8.4 Une première réduction

Comme déjà évoqué, nous ne connaissons pas d'algorithme polynomial pour résoudre des problèmes comme Clique ou encore Independent Set. Cependant, nous avons montré à la fin du cours que si l'on pouvait résoudre Independent Set en temps polynomial, cela permettrai de résoudre Clique en temps polynomial.

Nous avons montré cela par le biais d'une **réduction**. Cette réduction ainsi que d'autres seront décrites dans les notes du prochain cours.

8.5 Définition plus formelle de NP (version à base de certificats)

La deuxième version de la définition de NP que nous avons utilisé plus haut est valide, mais n'est pas très précise. Pour information, voici la définition complète :

- NP : Un langage L est dans NP si et seulement si il existe une machine de Turing *déterministe* M appelée un **vérifieur** qui s'exécute en temps polynomial en la taille de l'entrée w et tel que pour tout $w \in L$, il existe un mot p (le certificat positif / la preuve), lui-aussi de taille polynomiale en la taille de w, tel que $M(w, p)$ accepte, et si $w \notin L$, il n'existe aucun mot p de taille polynomiale en la taille de w, tel que $M(w, p)$ accepte.

Cette définition est un peu lourde, utilisez-là en cas de doute.