

8. Classe NP

Enseignant: Arnaud Casteigts

*Assistants: M. De Francesco, M. Marseloo
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 raison pour laquelle la classe NP est importante est qu'elle admet une autre définition équivalente, qui n'utilise aucun non-déterminisme :

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

Autrement dit, si un langage L est dans NP, alors pour tout mot $w \in L$, il *existe* une preuve que $w \in L$ (un certificat positif) qui peut être *vérifiée* en temps polynomial. Ce certificat/preuve est potentiellement différent pour chaque mot de L .

Prenons pour exemple le problème de décision 3-COLORING :

Entrée : Un graphe G

Question : Est-ce que G est 3-colorable ? Autrement dit, peut-on colorier tous ses sommets en rouge, vert ou bleu de sorte que toute paire de sommets voisins a des couleurs différentes ?

Résoudre ce problème revient à décider le langage du même nom :

$$3\text{-COLORING} = \{G \mid G \text{ est un graphe 3-colorable}\}$$

À ce jour, on ne sait pas si 3-COLORING est dans P : on ne connaît aucun algorithme qui résoud ce problème en temps polynomial quel que soit le graphe donné en entrée. En revanche, si un graphe G est 3-colorable, alors il existe clairement une preuve facile à *vérifier* : la coloration elle-même. En effet, si l'on nous donne cette coloration, on peut facilement vérifier qu'elle est valide, ce qui implique que G est bien dans le langage 3-COLORING. Ce langage est donc dans NP.

8.1.1 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 en temps polynomial avec une machine non-déterministe. Notre machine va simplement choisir parmi les 3 couleurs possibles pour le premier sommet, ce qui crée 3 branches d'exécution. Dans chaque branche, la machine choisit alors de manière non-déterministe la couleur du second sommet, ce qui crée trois nouvelles sous-branches dans chaque branche, *etc.* Une fois que tous les noeuds sont coloriés dans une branche, on *vérifie* que la coloration est valide, et si elle l'est, la branche accepte (et donc la machine accepte).

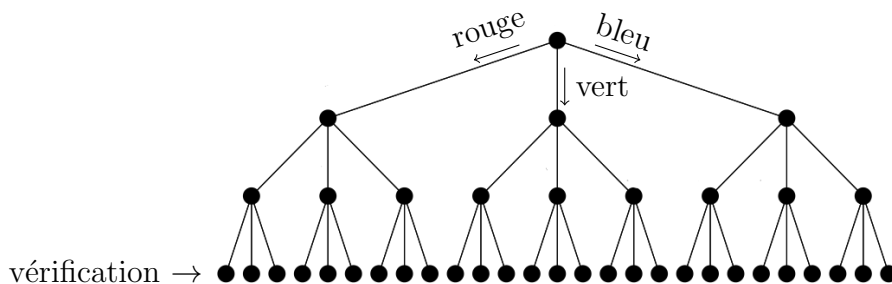


FIGURE 1 – Exploration non-déterministe d'une 3-coloration.

Cet exemple nous montre qu'une machine non-déterministe peut *deviner* le certificat, la seule difficulté étant ensuite de le vérifier. 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. Il suffit de montrer que $NP_2 \subseteq NP_1$ et $NP_1 \subseteq NP_2$.

- $NP_2 \subseteq NP_1$: 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 $L \in NP_1$.
- $NP_1 \subseteq NP_2$: 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 $L \in NP_2$. □

8.2 Relations avec P

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 ($P \subseteq NP$). 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 $P \subseteq NP$ à nouveau).

L'une des questions les plus fondamentales en informatique et en mathématiques est de savoir si $P = NP$. 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 (polynomiales dans la taille de l'énoncé). Par définition, ce langage est dans NP. Si $P = NP$, cela implique qu'il existe un algorithme unique capable de décider tous ces énoncés (on ne gagne donc pas seulement 1 million, mais 6... :-)). Cependant, la majorité des spécialistes pensent que $P \neq NP$. Notez que l'expression " $P = NP$ " est elle-même un énoncé mathématique... (vertige ?)

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

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

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 très étudiés sont dans cette classe. Voici quelques exemples pour lesquels on ne connaît pas d'algorithme polynomial (donc, potentiellement, ces problèmes sont dans $NP \setminus P$) :

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

- 3-COLORING (déjà vu)
- SUBSET SUM : Étant donné une liste de nombres, peut-on en sélectionner certains pour obtenir une somme à 0 ?
- FACTORING : Étant donné un entier N et un entier k , N admet-il un facteur $\leq k$?
- SAT : Étant donné une formule booléenne ϕ sous forme normale conjonctive, par exemple $\phi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3)$ (où \bar{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 ? (Notez que \vee signifie OU et \wedge signifie ET.) Nous reviendrons sur ce problème important.

- 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 ?
- 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 ?
- 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 ?

Exercice 1 : pour chaque problème, réfléchir au fait qu'il admet des certificats positifs.

Exercice 2 : connaître la définition de ces problèmes par coeur.

8.4 Une première réduction ?

(Si le temps le permet...)

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 **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 $|w|$ tel que $M(w, p)$ accepte, et pour tout $w \notin L$, il n'existe aucun mot p de taille polynomiale $|w|$ tel que $M(w, p)$ accepte.

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