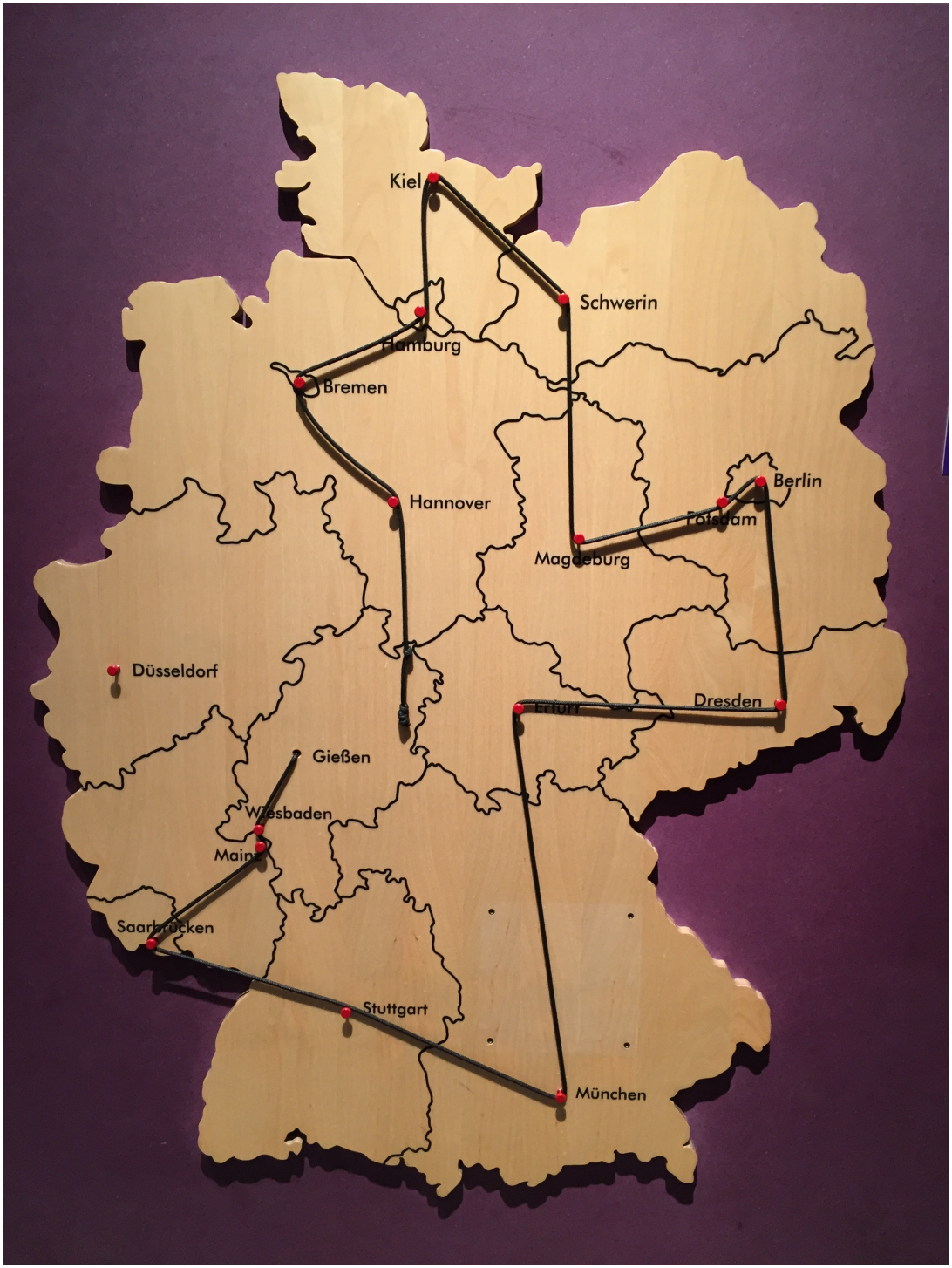
M1if09 – Modèles de calcul & complexité

sylvain.brandel@univ-lyon1.fr

COMPLEXITÉ
CLASSES P ET NP


Kiel
Schwerin
Hamburg
Bremen
Berlin
Hannover
Potsdam
Magdeburg
Düsseldorf
Dresden
Erfurt
Gießen
Wiesbaden
Mainz
Saarbrücken
Stuttgart
München

La classe P

- Exemple 1 : voyageur de commerce
  - Visite de n villes en faisant le moins de km possibles
  - Algorithme ?
    - Produire toutes les permutations de villes possibles
      (sauf la première qui est toujours la même)
    - Pour chaque permutation, calculer le trajet

      $\rightarrow (n-1)!$ permutations possibles

  - Souci ?
    - 10 villes $\rightarrow 9! = 362880$
    - 40 villes $\rightarrow 39! = 20397882081197443358640281739902897356800000000$

La classe P

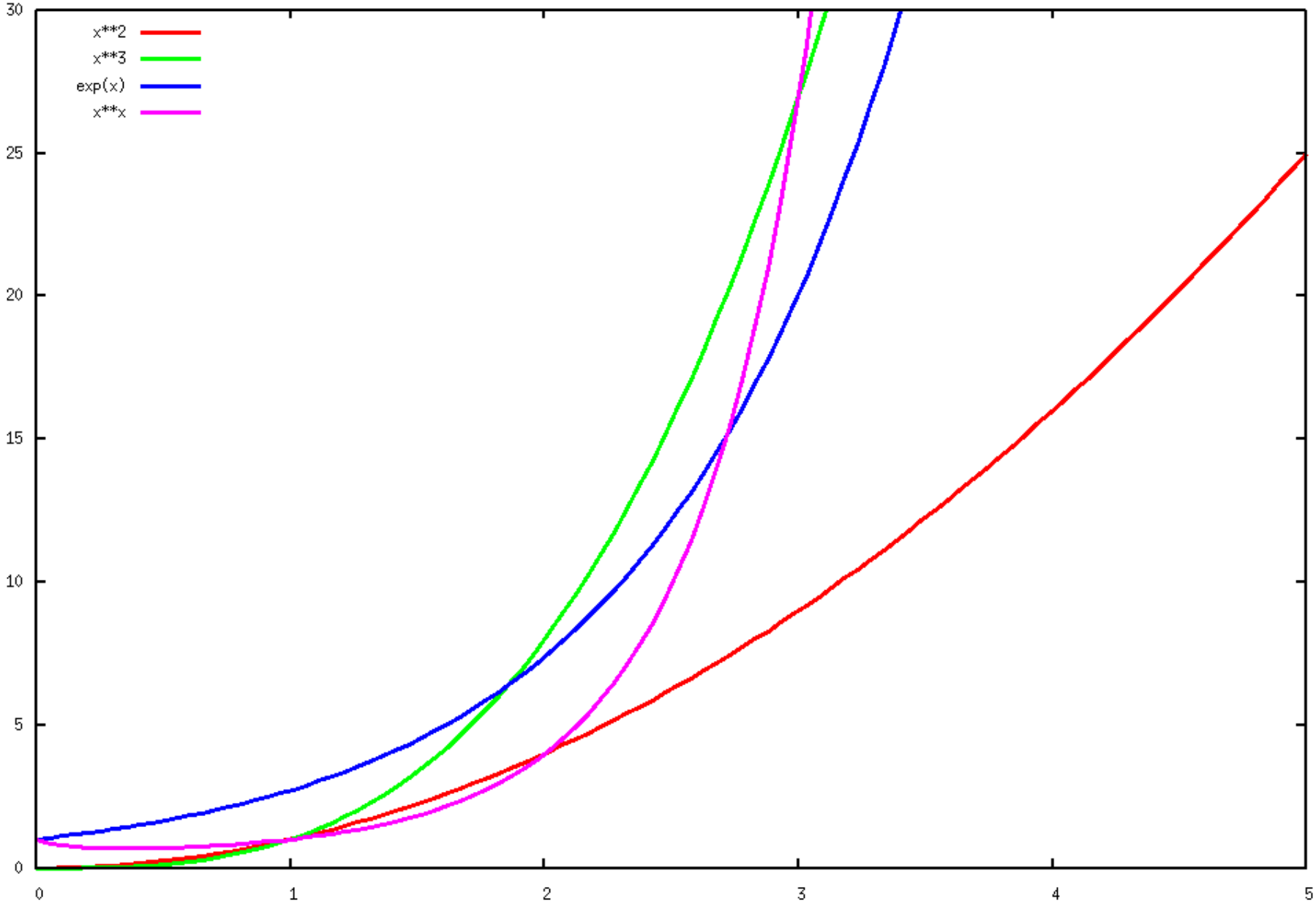
- Exemple 2 : géométrie tortue

```
void triangles1(float x, float y, float d, float h) {
  if (y+d < h) {
    tracer(x-d,y+d);
    triangles1(x-d,y+d,d,h);
    tracer(x+d,y+d);
    triangles1(x+d,y+d,d,h);
    tracer(x,y);
  }
}
```

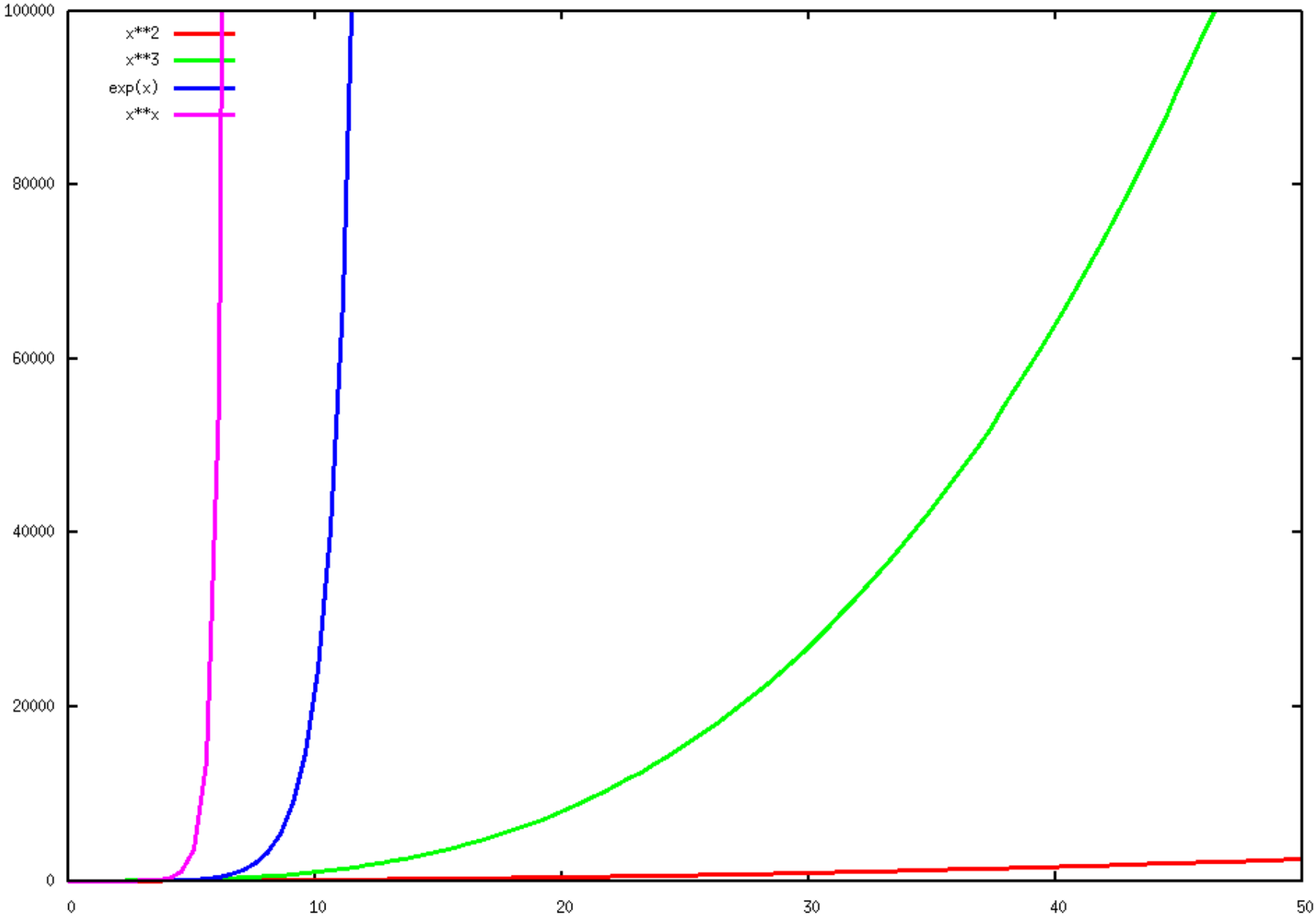
Appel avec

```
placer(x,y); triangles1(x,y,d,h);
```

La classe P



La classe P



Machine de Turing

- Machine de Turing standard : quintuplet $M = (K, \Sigma, \Gamma, \delta, q_0)$
  - K : ensemble fini d'états
  - $\sum$: alphabet d'entrée
  - Γ : alphabet des symboles du ruban
  - δ : fonction de transition fonction partielle $K \times \Gamma \longrightarrow K \times \Gamma \times \{\rightarrow, \leftarrow, \downarrow\}$
  - $q_0 \in K$: état initial

- Machine de Turing non déterministe : sextuplet $M = (K, \Sigma, \Gamma, \Delta\ q_0, F)$
  - Δ : relation de transition $\Delta \subset K \times \Gamma \times K \times \Gamma \times \{\rightarrow, \leftarrow, \downarrow\}$
  - $F \subset K$: états acceptants

  $\rightarrow$ Plusieurs sorties différentes pour la même entrée

  $\rightarrow$ Accepteur dont le seul résultat qui nous intéresse est de savoir si la machine s'arrête ou non

La classe P

- Soit une machine de Turing déterministe $M = (K, \Sigma, \Gamma, \delta, q_0)$

- M est dite polynomialement bornée s'il existe un polynôme p tel que pour toute entrée w, il n'y a pas de configuration C telle que

$$(q_0, \underline{\#}w) \vdash_M^{p(|w|)+1} C$$

  c'est-à-dire M s'arrête toujours, et ce, en au plus $p(|w|)$ étapes

- Un langage est dit polynomialement décidable ssi il existe une machine de Turing polynomialement bornée qui le décide

- La classe des langages polynomialement décidables est notée P

Thèse de Church – Turing

- *Un algorithme est une machine de Turing qui s'arrête pour toutes ses entrées*
- *We therefore propose to adapt the Turing machine that halts on all inputs as the precise formal notion corresponding to the intuitive notion of an "algorithm"*
- *Les langages reconnus par une procédure effective sont ceux décidés par une machine de Turing*
- *Every computational process that is intuitively considered to be an algorithm can be converted to a Turing machine*

La classe P

- La thèse de Church-Turing est raffinée en : Les machines de Turing polynomialement bornées et la classe P correspondent aux notions
  - D'algorithmes pratiquement exécutables
  - Et de problèmes réellement solvables

- Théorème

  La classe P est stable par complément

- Théorème

  Il existe des langages récursifs non polynomialement décidables

Exemples de problèmes et de complexité

- Exemple 1 : existence d'un chemin
  - Entrées :
    - Un ensemble de sommets $V = \{v_1, \ldots, v_n\}$
    - Un graphe orienté $G \in V \times V$
    - Deux sommets v_i et $v_j \in V$
  - Question :
    - Existe-t-il un chemin entre v_i et v_j ?
  - Algorithme : calcul de la fermeture réflexive – transitive
    - Complexité : $O(n^3)$

- Le problème de l'existence d'un chemin $\in P$

Exemples de problèmes et de complexité

- Exemple 2 : graphes Eulériens
  - Entrées :
    - Un graphe G
  - Question :
    - Existe-t-il un chemin fermé (cycle) dans G qui utilise chaque arête une fois et une seule ?
  - Un graphe qui contient un tel cycle est dit Eulérien ou unicursal

- Le problème du cycle Eulérien $\in P$

Exemples de problèmes et de complexité

- Exemple 3 : graphes Hamiltoniens
  - Entrées :
    - Un graphe G
  - Question :
    - Existe-t-il un cycle passant par chaque sommet une fois et une seule ?
  - Un graphe qui contient un tel cycle est dit Hamiltonien
  - Algorithme :
    - Examiner toutes les permutations de nœuds possibles ($\rightarrow$ exponentiel)
    - Regarder si le parcours correspondant existe dans G
- Le problème des cycles Hamiltoniens n'est pas connu comme étant dans P

Classe NP

- Soit une machine de Turing non déterministe $M = (K, \Sigma, \Gamma, \Delta\, q_0, F)$

- M est dite polynomialement bornée s'il existe un polynôme p tel que pour toute entrée w, il n'y a pas de configuration C telle que

$$(q_0, \#w) \vdash_M^{p(|w|)+1} C$$

c'est-à-dire M s'arrête toujours, et ce, en au plus $p(|w|)$ étapes.

- La classe des langages décidés par une machine de Turing non déterministe polynomialement bornée est notée NP

NP pour *Non déterministe Polynomial*

Classe NP

- Rappel : Logique d'ordre 0 – calcul des propositions :
  - Variables booléennes
  - Connecteurs (opérations) : $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$ et $\neg$
  - Littéral : expression de la forme p ou $\bar{p}$ — p : variable propositionnelle
  - Clause : disjonction de littéraux

    $E = x_1 \vee x_2 \vee \cdots \vee x_n$ — chaque x_i : littéral
  - Forme normale conjonctive : conjonction de clauses

    $F = E_1 \wedge E_2 \wedge \cdots \wedge E_n$ — chaque E_i : clause

    $F = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n} x_i^j$ — chaque x_i^j : littéral
  - Assignation booléenne ou interprétation :

    fonction $X \rightarrow (\top, \bot)$ — X : ensemble des variables; $\top$: vrai, $\bot$: faux
  - Satisfiabilité : il existe (au moins) une interprétation rendant la formule vraie

Classe NP

- Problème SAT
  - Entrée :
    - F : forme normale conjonctive
  - Question :
    - F est-elle satisfiable ?

- Proposition

Le problème SAT est dans NP

Classe NP

- Algorithme brutal pour le problème SAT
  - Faire une table de vérité

    $\rightarrow$ exponentiel en fonction du nombre de variables
  - Pour chaque assignation, tester la FNC

    $\rightarrow$ linéaire

  $\Longrightarrow$ exponentiel

- 2-SAT (au plus 2 littéraux par clause) : $(a \vee b) \wedge (c \vee d) \wedge e \wedge \cdots$
  - 2-SAT $\in P$ (algorithme P connu, même si table de vérité de taille exponentielle)

- 3-SAT (exactement 3 littéraux par clause) : $(a \vee b \vee c) \wedge (d \vee e \vee f) \wedge \cdots$
  - 3-SAT $\in NP$ (pas d'algorithme P connu, algorithme NP connu)

- SAT (FNC avec clauses avec nombre quelconque de littéraux)
  - SAT $\in NP$

Classe NP

- Voyageur de commerce (faible)
  - Entrées :
    - Un entier $n \geq 2$
    - Une matrice de distance d_{ij}
    - Un entier $B \geq 0$ B : budget du voyageur de commerce
  - Question :
    - Existe-t-il une permutation π sur $\{1, 2, \ldots, n\}$ telle que $C(\pi) \leq B$, où :

$$C(\pi) = d_{\pi(1)\,\pi(2)} + d_{\pi(2)\,\pi(3)} + \ldots + d_{\pi(n-1)\,\pi(n)} + d_{\pi(n)\,\pi(1)}$$

- Proposition

Le problème du voyageur de commerce est dans NP

Classe NP

- De manière semblable, des algorithmes de type :
  - Générer de manière non déterministe une situation
  - Tester de manière polynomiale cette situation

  Peuvent résoudre des problèmes précédents avec des machines de Turing ND polynomialement bornées
  - Si le test de la situation se fait avec une machine de Turing (déterministe) polynomialement bornée
  - Et si la taille de la situation est bornée polynomialement en fonction des entrées

Classe NP

- 2-SAT $\in P$, 3-SAT $\in NP$ $\Longrightarrow$ SAT $\in NP$
- Ce qui ne veut pas dire que SAT $\notin P$
- $P \subseteq NP$ (clair)
- $NP \subseteq P$? Cela voudrait dire :
  1) Qu'il existe une MT polynomialement bornée équivalente à une MT ND polynomialement bornée
  2) Que les problèmes SAT, voyageur de commerce, cycle de Hamilton, etc. seraient dans P
- En fait, on ne sait pas si $NP \subseteq P$ (et donc $NP = P$).

Classe EXP

- Soit une machine de Turing déterministe $M = (K, \Sigma, \Gamma, \delta, q_0)$
- M est dite exponentiellement bornée s'il existe un polynôme p tel que pour toute entrée w, il n'y a pas de configuration C telle que

  $(q_0, \#w) \vdash_M^{2^{p(|w|)}+1} C$

  C'est-à-dire M s'arrête toujours, et ce en au plus $2^{p(|w|)}$ étapes
- On note EXP la classe des langages qui peuvent être décidés par une MT exponentiellement bornée
- Théorème

  Si $L \in NP$ alors $L \in EXP$

  Autrement dit : $NP \subseteq EXP$

Classe NP
Certificat

- MT non dét. polynom. bornée : résolution de problèmes d'existence
  - Production d'une situation
  - Test
    - Réussite à au moins une → problème d'existence résolu
    - Échec à tous → problème d'existence résolu (réponse négative)

- Certificat (témoin) : Mot synthétisant une solution et passant le test avec succès
  - Tous les problèmes NP ont des certificats
  - Seuls les problèmes NP ont des certificats
  - En pratique
    - Longueur des certificats polynomiale par rapport aux entrées
    - Test des certificats en temps polynomial

Classe NP
Certificat

- Définition

  Soient $\sum$ un alphabet et "; " un symbole $\notin \sum$

  Soit L' un langage tel que $L' \subseteq \sum^*; \sum^*$

  On dit que L' est polynomialement équilibré s'il existe un polynôme p tel que

  si $x\ ;\ y \in L'$ alors $|y| \leq p(|x|)$

- Théorème

  Soit $\sum$ un alphabet pour lequel "; " $\notin \sum$ et $|\sum| \geq 2$

  Soit $L \subseteq \sum^*$ un langage

  $L \in NP$ ssi il existe un langage polynomialement équilibré $L' \subseteq \sum^*; \sum^*$ tel que

  $L' \in P$ *et* $L = \{x \mid \exists\ y \in \sum^*:\ x\ ;\ y \in L'\}$

  – $L' = \{\, x\ ;\ y \mid y$ est un certificat pour $x\,\}$

  – L' : langage de tous les certificats de toutes les entrées

  – Si $x \in L$, alors il y a au moins un certificat

    - Si L' existe, alors une MT non dét. décide L en testant tous les certificats, en utilisant une MT dét. décidant L', donc $L \in NP$
    - Si $L \in NP$, alors il existe une MT non dét. polynomialement bornée décidant L

Classe NP

Certificat

- SAT
  - L : SAT
  - L' : une interprétation validant chaque instance positive

- 3-COL
  - L : 3-COL
  - L' : un coloriage pour chaque graphe 3-coloriable

- Nombres composés
  - L : ensemble des nombres composites (c-à-d au moins 1 div. $\neq 1$ et du nb.)
  - L' : une paire d'entiers composant chaque nombre
  - Ex : $4 \in L$
  - Ex : $12 \in L$
  - Ex : $4\,294\,967\,297 \in L$

facile, 4 ; $(2, 2) \in L'$

facile, 12 ; $(6, 2) \in L'$, ou 12 ; $(3, 4) \in L'$, ou …

moins facile : $4\,294\,967\,297$; $(6\,700\,417, 641) \in L$

(Fermat 1640)

(Euler 1732)