Qualité algorithmique

BUT info 3A

Classes de complexité, réductions polynomiales

Florent Foucaud, Pascal Lafourcade, Malika More

IUT CLERMONT AUVERGNE

Aurillac - Clermont-Ferrand - Le Puy-en-Velay
Montluçon - Moulins - Vichy

Problèmes de décision

BUT : classifier les problèmes algorithmiques selon leur complexité.

Problèmes de décision

BUT : classifier les problèmes algorithmiques selon leur complexité.

On simplifie :
→ Problème de décision : entrée (instance), question à réponse OUI/NON

- Est-ce que cette liste est triée ?
- Est-ce que ce graphe est 3-colorable ?
- Est-ce que le nombre c est bien égal à $a \times b$?
- Est-ce que ce programme s'arrête pour toute entrée ?
- ...

Problèmes de décision

BUT : classifier les problèmes algorithmiques selon leur complexité.

On simplifie :
→ Problème de décision : entrée (instance), question à réponse OUI/NON

- Est-ce que cette liste est triée ?
- Est-ce que ce graphe est 3-colorable ?
- Est-ce que le nombre c est bien égal à $a \times b$?
- Est-ce que ce programme s'arrête pour toute entrée ?
- ...

On peut le plus souvent modéliser un problème algorithmique avec entrée, sortie comme un problème de décision (en posant la bonne question).

Problèmes de décision

BUT : classifier les problèmes algorithmiques selon leur complexité.

On simplifie :
→ Problème de décision : entrée (instance), question à réponse OUI/NON

- Est-ce que cette liste est triée ?
- Est-ce que ce graphe est 3-colorable ?
- Est-ce que le nombre c est bien égal à $a \times b$?
- Est-ce que ce programme s'arrête pour toute entrée ?
- ...

On peut le plus souvent modéliser un problème algorithmique avec entrée, sortie comme un problème de décision (en posant la bonne question).

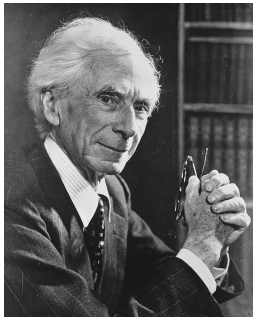
Liste d'Entiers Triée
Instance: Une liste d'entiers L
Question: Est-ce que L est triée dans l'ordre croissant ?

3-Coloration
Instance: Un graphe G
Question: Est-ce que G est 3-colorable ?

Paradoxe du barbier

Dans un village, le barbier rase exactement tous les hommes qui ne se rasent pas eux-mêmes.

Question : Qui rase le barbier ?



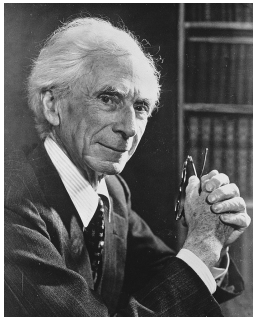
Bertrand Russell (1872-1970)

Paradoxe du barbier

Dans un village, le barbier rase exactement tous les hommes qui ne se rasent pas eux-mêmes.

Question : Qui rase le barbier ?

PARADOXE !



Bertrand Russell (1872-1970)

S’arrêter ou boucler ? Telle est la question

Problème de l’arrêt

Étant donné n’importe quel code de programme informatique, peut-on décider en temps fini :

1. s’il va s’arrêter un jour *ou bien* 2. s’il va tourner à l’infini ?

S’arrêter ou boucler ? Telle est la question

Problème de l’arrêt

Étant donné n’importe quel code de programme informatique, peut-on décider en temps fini :

1. s’il va s’arrêter un jour *ou bien* 2. s’il va tourner à l’infini ?

Théorème (Alan Turing, 1936)

Il n’existe aucun algorithme pour résoudre le problème de l’arrêt.

Alan Turing (1912-1954)

S’arrêter ou boucler ? Telle est la question

Problème de l’arrêt

Étant donné n’importe quel code de programme informatique, peut-on décider en temps fini :

1. s’il va s’arrêter un jour *ou bien* 2. s’il va tourner à l’infini ?

Théorème (Alan Turing, 1936)

Il n’existe aucun algorithme pour résoudre le problème de l’arrêt.

Démonstration : Supposons par l’absurde qu’il existe un programme en temps fini

```
arret(code, parametre)
```

qui renvoie
- VRAI si le code donné avec le paramètre s’arrêtera un jour, et
- FAUX si au contraire le code tourne à l’infini.

S’arrêter ou boucler ? Telle est la question

Problème de l’arrêt

Étant donné n’importe quel code de programme informatique, peut-on décider en temps fini :

1. s’il va s’arrêter un jour *ou bien* 2. s’il va tourner à l’infini ?

Théorème (Alan Turing, 1936)

Il n’existe aucun algorithme pour résoudre le problème de l’arrêt.

Démonstration : Supposons par l’absurde qu’il existe un programme en temps fini

```
arret(code, parametre)
```

qui renvoie
- VRAI si le code donné avec le paramètre s’arrêtera un jour, et
- FAUX si au contraire le code tourne à l’infini.

Soit le programme suivant :

```
def diag(x):
```
- `si arret(x,x) est VRAI alors:`
  - `boucle infinie`
- `sinon:`
  - `retourner VRAI`

S'arrêter ou boucler ? Telle est la question

Problème de l'arrêt

Étant donné n'importe quel code de programme informatique, peut-on décider en temps fini :

1. s'il va s'arrêter un jour *ou bien* 2. s'il va tourner à l'infini ?

Théorème (Alan Turing, 1936)

Il n'existe aucun algorithme pour résoudre le problème de l'arrêt.

Démonstration : Supposons par l'absurde qu'il existe un programme en temps fini

```
arret(code, parametre)
```

qui renvoie
- VRAI si le code donné avec le paramètre s'arrêtera un jour, et
- FAUX si au contraire le code tourne à l'infini.

Soit le programme suivant :

```
def diag(x):
  • si arret(x,x) est VRAI alors:
    ▸ boucle infinie
  • sinon:
    ▸ retourner VRAI
```

Que renvoie l'appel `diag(diag)` ?

S'arrêter ou boucler ? Telle est la question

Problème de l'arrêt

Étant donné n'importe quel code de programme informatique, peut-on décider en temps fini :

1. s'il va s'arrêter un jour *ou bien* 2. s'il va tourner à l'infini ?

Théorème (Alan Turing, 1936)

Il n'existe aucun algorithme pour résoudre le problème de l'arrêt.

Démonstration : Supposons par l'absurde qu'il existe un programme en temps fini

```
arret(code, parametre)
```

qui renvoie
- VRAI si le code donné avec le paramètre s'arrêtera un jour, et
- FAUX si au contraire le code tourne à l'infini.

Soit le programme suivant :

```
def diag(x):
  • si arret(x,x) est VRAI alors:
    ▸ boucle infinie
  • sinon:
    ▸ retourner VRAI
```

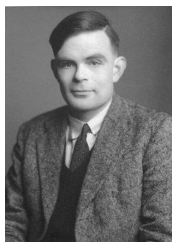
Que renvoie l'appel `diag(diag)` ?

PARADOXE !

Problèmes indécidables

Problèmes indécidables :

- Problème de l'arrêt (Alan Turing, 1936)
- Problème de correspondance de mots : 2 paquets de mots $a_1, \ldots, a_n$ et $b_1, \ldots, b_n$
  → Peut-on les arranger pour créer deux mots identiques ? (Emil Post, 1946)
- Trouver des solutions entières d'équations diophantiennes
  de type $2x^2 + 3y^3 - 2z = 0$ (10e problème de Hilbert, 1900 - Youri Matyasevitch, 1970)
- Déterminer le vainqueur d'une partie du jeu "Magic : The gathering"
  (Churchill-Biderman-Herrick, 2019)



Alan Turing (1912-1954)

Emil L. Post (1897-1954)

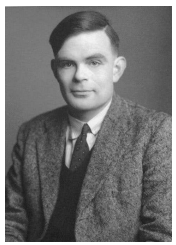
Youri Matyasevitch (1947-)

David Hilbert (1862-1943)

Problèmes indécidables

Problèmes indécidables :

- Problème de l'arrêt (Alan Turing, 1936)
- Problème de correspondance de mots : 2 paquets de mots $a_1, \ldots, a_n$ et $b_1, \ldots, b_n$
  → Peut-on les arranger pour créer deux mots identiques ? (Emil Post, 1946)
- Trouver des solutions entières d'équations diophantiennes
  de type $2x^2 + 3y^3 - 2z = 0$ (10e problème de Hilbert, 1900 - Youri Matyasevitch, 1970)
- Déterminer le vainqueur d'une partie du jeu "Magic : The gathering"
  (Churchill-Biderman-Herrick, 2019)



Alan Turing (1912-1954)

Emil L. Post (1897-1954)

Youri Matyasevitch (1947-)

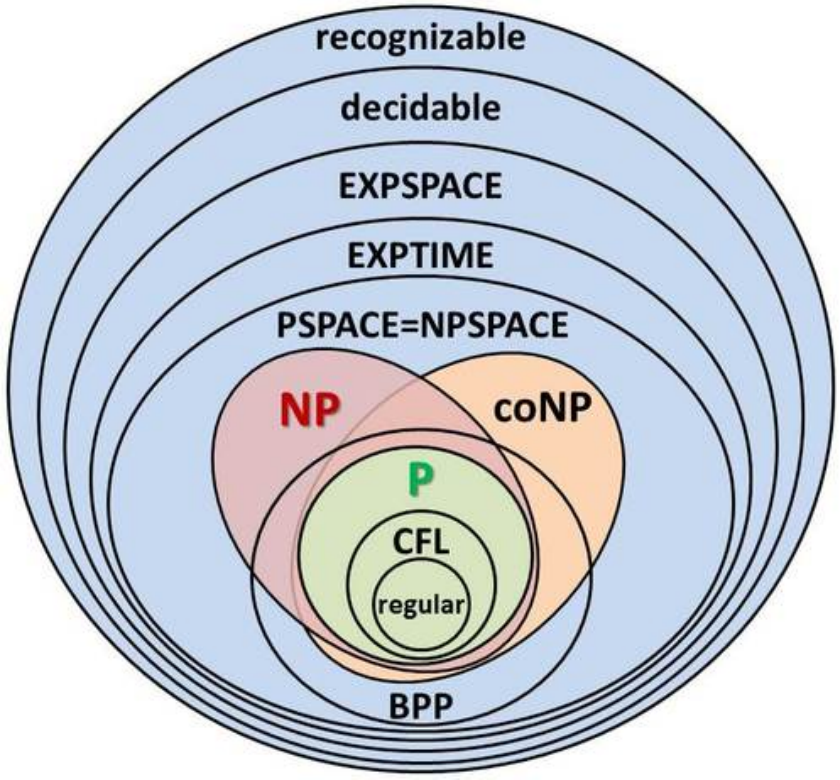
David Hilbert (1862-1943)

Liens avec la logique mathématique :

théorème d'incomplétude de Gödel (1931)

Kurt Gödel (1906-1978)

Quelques classes de complexité algorithmique

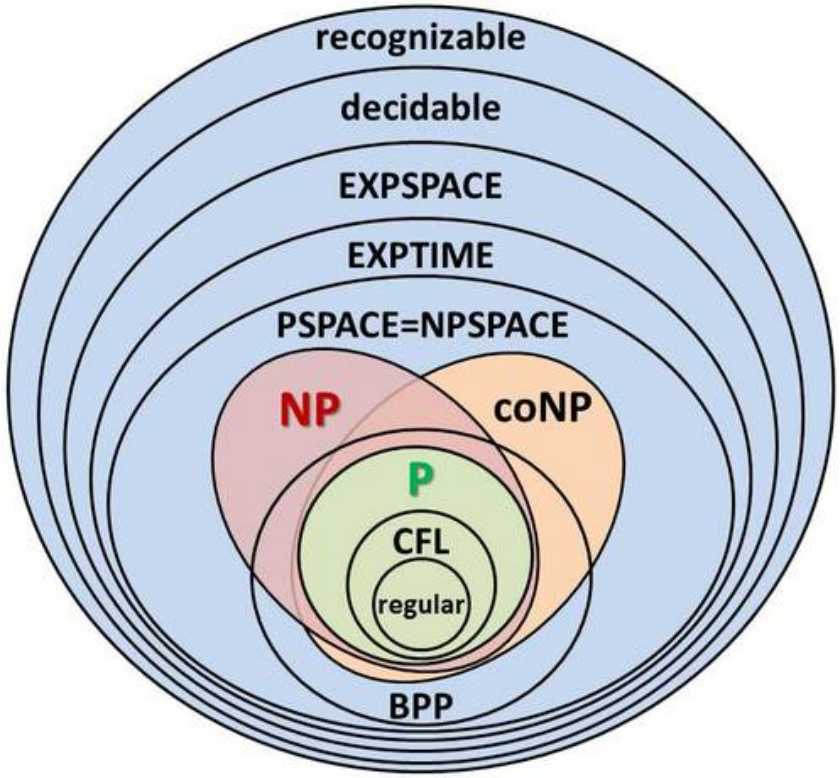


Classe P (“polynomiaux”) : problèmes “raisonnables” (Cobham-Edmonds, 1965)

Jack Edmonds (1934-)

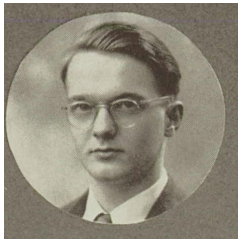
Alan B. Cobham (1927-2011)

Quelques classes de complexité algorithmique



Classe P (“polynomiaux”) : problèmes “raisonnables” (Cobham-Edmonds, 1965)

Jack Edmonds (1934-)



Alan B. Cobham (1927-2011)

Au-dessus : problèmes (probablement) algorithmiquement difficiles

Machine de Turing (Alan Turing, 1936)

Définition :

- Un alphabet de travail Σ
- Un ruban infini (= mémoire vive), avec des positions $\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$
- Une tête de lecture/écriture positionnée sur le ruban
- un ensemble Q d'états, avec un état initial q_0 et un ensemble d'états acceptants F
- une fonction de transition $\delta : Q \times \Sigma \to \{\text{gauche}, \text{droite}\} \times \Sigma \times Q$: en fonction de l'état courant et du symble lu par la tête, la tête se déplace d'un cran à gauche ou à droite, peut écrire un symbole, et peut changer d'état.

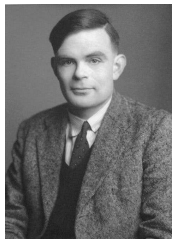
Machine de Turing (Alan Turing, 1936)

Définition :

- Un alphabet de travail Σ
- Un ruban infini (= mémoire vive), avec des positions $\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$
- Une tête de lecture/écriture positionnée sur le ruban
- un ensemble Q d'états, avec un état initial q_0 et un ensemble d'états acceptants F
- une fonction de transition $\delta : Q \times \Sigma \rightarrow \{\text{gauche}, \text{droite}\} \times \Sigma \times Q$: en fonction de l'état courant et du symble lu par la tête, la tête se déplace d'un cran à gauche ou à droite, peut écrire un symbole, et peut changer d'état.

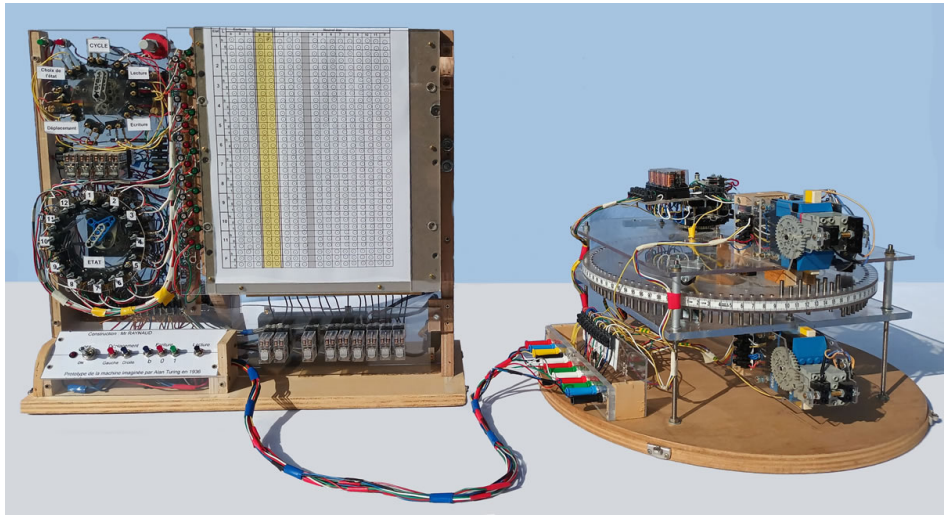
Fonctionnement de la machine de Turing :

- une entrée quelconque est écrite sur le ruban.
- l'état courant est l'état initial q_0
- la machine exécute la fonction de transition jusqu'à arriver dans un un état acceptant ou bien être bloquée

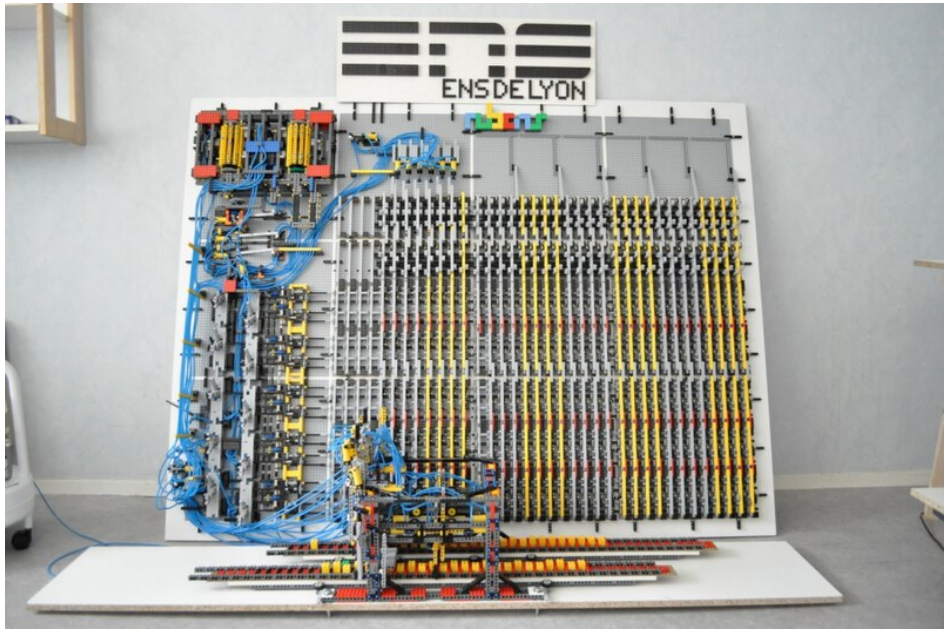


Alan Turing (1912-1954)

Dans la vraie vie



Machine de Turing construite par Marc Raynaud en 2013
https://machinedeturing.com/



Machine de Turing en LEGO construite par des étudiants de l'ENS de Lyon en 2012
https://www.ens-lyon.fr/actualite/lecole/la-machine-de-turing-en-legos

Machine de Turing universelle

Théorème (Alan Turing, 1936)

Il existe une machine de Turing universelle capable de simuler tout autre machine de Turing. (entrée : code de la machine à simuler, et entrée à exécuter)

(Cela peut être réalisé avec un facteur de complexité logarithmique.)

Machine de Turing universelle

Théorème (Alan Turing, 1936)

Il existe une machine de Turing universelle capable de simuler tout autre machine de Turing. (entrée : code de la machine à simuler, et entrée à exécuter)

(Cela peut être réalisé avec un facteur de complexité logarithmique.)

Remarque

Tous les ordinateurs actuels sont équivalents à une machine de Turing universelle ! (modulo la mémoire finie vs. infinie)

L'article de Turing de 1936 représente l'invention de l'ordinateur moderne et certaines techniques de programmation qui l'ont accompagné. Minsky, 1967.

Classes de complexité P et NP

Définition (Classe P)

Problèmes de décision qui peuvent être résolus en temps polynomial.

⟶ problèmes pour lesquels on peut calculer une solution en temps polynomial, si elle existe

Classes de complexité P et NP

Définition (Classe P)

Problèmes de décision qui peuvent être résolus en temps polynomial.

⟶ problèmes pour lesquels on peut calculer une solution en temps polynomial, si elle existe

Exemples :

- Trier une liste d'entiers
- Chemin le plus court dans un graphe
- Programmation linéaire
- ...

Classes de complexité P et NP

Définition (Classe P)

Problèmes de décision qui peuvent être résolus en temps polynomial.

⟶ problèmes pour lesquels on peut calculer une solution en temps polynomial, si elle existe

Exemples :

- Trier une liste d'entiers
- Chemin le plus court dans un graphe
- Programmation linéaire
- ...

Définition (Classe NP = "non-déterministe polynomial")

Problèmes de décision pour lesquels il existe un certificat (fonction de l'instance) tel que, pour une instance I et son certificat $C(I)$, on peut vérifier en temps polynomial en I, si I est une instance positive ou pas.

⟶ problèmes pour lesquels on peut vérifier en temps polynomial si une solution est correcte ou pas

Classes de complexité P et NP

Définition (Classe P)

Problèmes de décision qui peuvent être résolus en temps polynomial.

⟶ problèmes pour lesquels on peut calculer une solution en temps polynomial, si elle existe

Exemples :

- Trier une liste d'entiers
- Chemin le plus court dans un graphe
- Programmation linéaire
- ...

Définition (Classe NP = "non-déterministe polynomial")

Problèmes de décision pour lesquels il existe un certificat (fonction de l'instance) tel que, pour une instance I et son certificat $C(I)$, on peut vérifier en temps polynomial en I, si I est une instance positive ou pas.

⟶ problèmes pour lesquels on peut vérifier en temps polynomial si une solution est correcte ou pas

Exemples :

- Tous les problème de la classe P
- Coloration de graphe
- Voyageur de commerce
- ...

P versus NP

Question de philosophie des sciences :
Est-ce que vérifier une solution et trouver une solution ont la même complexité ?

P versus NP

Question de philosophie des sciences :
Est-ce que vérifier une solution et trouver une solution ont la même complexité ?

→ Intuitivement, non...

P versus NP

Question de philosophie des sciences :
Est-ce que vérifier une solution et trouver une solution ont la même complexité ?

→ Intuitivement, non...

Question (P versus NP - une question à 1 million de $)

Est-ce que $P = NP$?

P versus NP

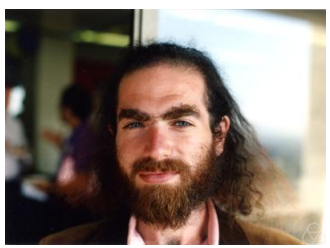
Question de philosophie des sciences :
Est-ce que vérifier une solution et trouver une solution ont la même complexité ?

→ Intuitivement, non...

Question (P versus NP - une question à 1 million de $)

Est-ce que $P = NP$?

7 problèmes du millénaire à 1 million de $



Grigori Perelman (1966-)

Un problème plus facile qu'un autre ?

Min « plus facile » que Tri

- je sais trier un tableau ⇝ je sais trouver le minimum de ce tableau
- un algorithme pour **Tri** permet de fabriquer un algorithme pour **Min**

Réduction (polynomiale)

$\mathcal{P}_1$

- Instance : E_1
- Question : E_1 possède P_1 ?

$\mathcal{P}_2$

- Instance : E_2
- Question : E_2 possède P_2 ?

Transformer E_1 en $f(E_1) = E_2$ (en temps polynomial)
de telle sorte que
E_1 possède $P_1 \iff E_2$ possède P_2

$\mathcal{P}_1$ se réduit à $\mathcal{P}_2$ (en temps polynomial)

- Algo pour $\mathcal{P}_2$: A_2 (polynomial)
- Algo pour $\mathcal{P}_1$: $E_1 \rightsquigarrow E_2 \overset{A_2}{\rightsquigarrow}$ oui ou non (en temps polynomial)

$\mathcal{P}_1$ est « plus facile » que $\mathcal{P}_2$

Réduction (polynomiale)

$\mathcal{P}_1$

- Instance : E_1
- Question : E_1 possède P_1 ?

$\mathcal{P}_2$

- Instance : E_2
- Question : E_2 possède P_2 ?

Transformer E_1 en $f(E_1) = E_2$ (en temps polynomial)
de telle sorte que
E_1 possède $P_1 \iff E_2$ possède P_2

$\mathcal{P}_1$ se réduit à $\mathcal{P}_2$ (en temps polynomial)

- Algo pour $\mathcal{P}_2$: A_2 (polynomial)
- Algo pour $\mathcal{P}_1$: $E_1 \rightsquigarrow E_2 \overset{A_2}{\rightsquigarrow}$ oui ou non (en temps polynomial)

$\mathcal{P}_1$ est « plus facile » que $\mathcal{P}_2$

Des exemples déjà vus :

- Cours de graphes en BUT 1 :
  Chemins disjoints pour graphe orienté → Flot de graphe orienté
- Cours de graphes en BUT 1 :
  Couplage de graphe biparti → Flot de graphe orienté
- Cours de Méthodes d'optimisation en BUT 2 :
  Flot de graphe orienté → Programmation Linéaire
- Cours précédent : **2-Coloration → SAT**

k-colorabilité se réduit à SAT

SAT
Instance: Une formule booléenne F en CNF
Question: Est-ce que F est satisfaisable ?

k-Colorabilité
Instance: Un graphe G
Question: Est-ce que G est k-colorable ?

Réduction polynomiale de k-colorabilité à SAT

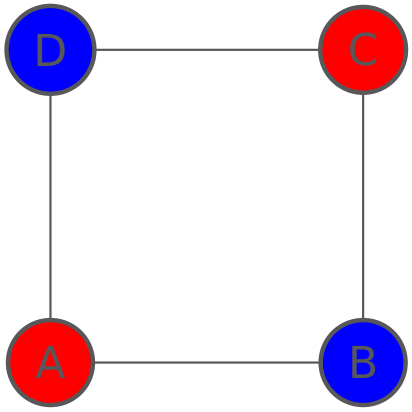
- Écrire F en temps polynomial en $n + m$ de telle sorte que

$$G \text{ est } k\text{-colorable} \iff F \text{ est satisfaisable}$$

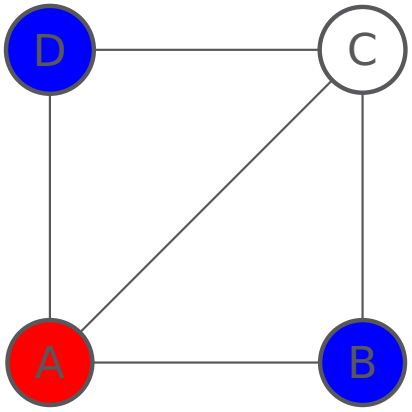
Si jamais on inventait un algorithme en temps polynomial pour **SAT**, on aurait aussi un algorithme en temps polynomial pour **k-colorabilité**

Un petit exemple

2-**Colorabilité**
Instance: Un graphe *G*
Question: Est-ce que *G* est 2-colorable ?



OUI

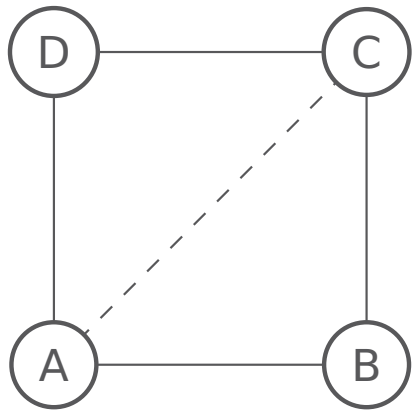


NON

Les variables du problème

Les couleurs des sommets

	Rouge	Bleu
A	$X1$	$X2$
B	$X3$	$X4$
C	$X5$	$X6$
D	$X7$	$X8$

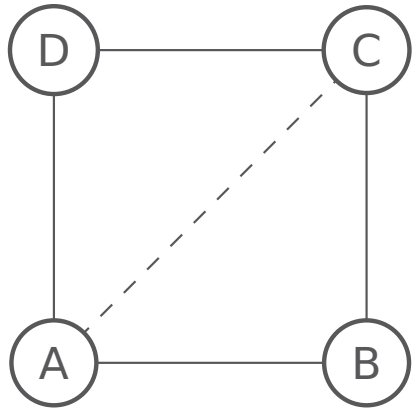


- $X1 = 1$: le sommet A est rouge
- $X1 = 0$: le sommet A n'est pas rouge
- $X2 = 1$: le sommet A est bleu
- $X2 = 0$: le sommet A n'est pas bleu
- etc.

Les contraintes portant sur les sommets

Chaque sommet est d'une couleur, mais pas des deux

	Rouge	Bleu
A	$X1$	$X2$
B	$X3$	$X4$
C	$X5$	$X6$
D	$X7$	$X8$



Pour A :

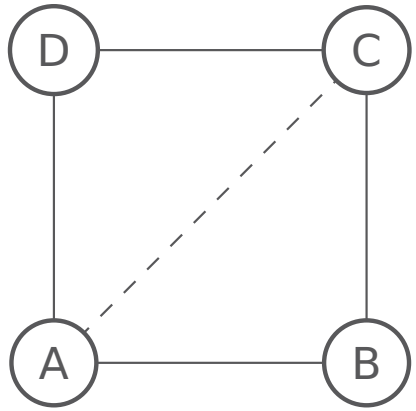
- $X1 + X2$
- $\overline{X1\,X2} = \overline{X1} + \overline{X2}$

Etc.

Les contraintes portant sur les arêtes

Deux sommets reliés ne peuvent pas être de la même couleur

	Rouge	Bleu
A	$X1$	$X2$
B	$X3$	$X4$
C	$X5$	$X6$
D	$X7$	$X8$



Pour AB :

- $\overline{X1\,X3} = \overline{X1} + \overline{X3}$
- $\overline{X2\,X4} = \overline{X1} + \overline{X4}$

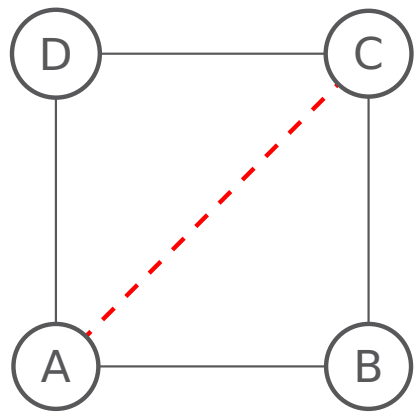
Etc.

La modélisation complète du problème

$(X1 + X2)(\overline{X1} + \overline{X2})$ (sommet A)
$(X3 + X4)(\overline{X3} + \overline{X4})$ (sommet B)
$(X5 + X6)(\overline{X5} + \overline{X6})$ (sommet C)
$(X7 + X8)(\overline{X7} + \overline{X8})$ (sommet D)
$(\overline{X1} + \overline{X3})(\overline{X2} + \overline{X4})$ (arête AB)
$(\overline{X1} + \overline{X7})(\overline{X2} + \overline{X8})$ (arête AD)
$(\overline{X7} + \overline{X5})(\overline{X8} + \overline{X6})$ (arête DC)
$(\overline{X3} + \overline{X5})(\overline{X4} + \overline{X6})$ (arête BC)
$(\overline{X1} + \overline{X5})(\overline{X2} + \overline{X6})$ (arête AC)

⇝ une formule en **CNF**

⇝ est-elle **satisfaisable** ?

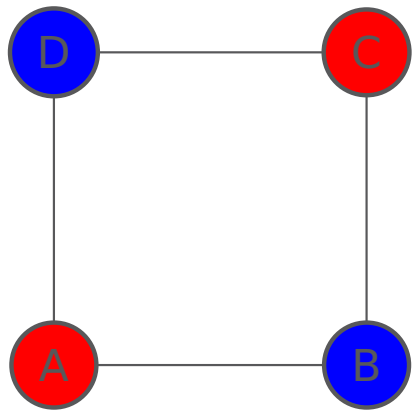


Formule satisfaisable = graphe 2-colorable

$$(X1 + X2)(\overline{X1} + \overline{X2}) \quad \text{(sommet } A\text{)}$$
$$(X3 + X4)(\overline{X3} + \overline{X4}) \quad \text{(sommet } B\text{)}$$
$$(X5 + X6)(\overline{X5} + \overline{X6}) \quad \text{(sommet } C\text{)}$$
$$(X7 + X8)(\overline{X7} + \overline{X8}) \quad \text{(sommet } D\text{)}$$
$$(\overline{X1} + \overline{X3})(\overline{X2} + \overline{X4}) \quad \text{(arête } AB\text{)}$$
$$(\overline{X1} + \overline{X7})(\overline{X2} + \overline{X8}) \quad \text{(arête } AD\text{)}$$
$$(\overline{X7} + \overline{X5})(\overline{X8} + \overline{X6}) \quad \text{(arête } DC\text{)}$$
$$(\overline{X3} + \overline{X5})(\overline{X4} + \overline{X6}) \quad \text{(arête } BC\text{)}$$

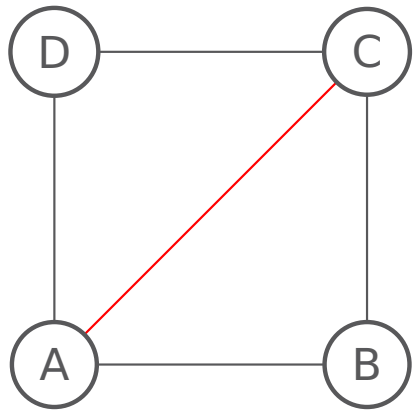
$$\begin{array}{lcl} X1 & \mapsto & 1 \\ X2 & \mapsto & 0 \\ X3 & \mapsto & 0 \\ X4 & \mapsto & 1 \\ X5 & \mapsto & 1 \\ X6 & \mapsto & 0 \\ X7 & \mapsto & 0 \\ X8 & \mapsto & 1 \end{array}$$

	Rouge	Bleu
A	$X1 = 1$	$X2 = 0$
B	$X3 = 0$	$X4 = 1$
C	$X5 = 1$	$X6 = 0$
D	$X7 = 0$	$X8 = 1$



Formule non satisfaisable = graphe non 2-colorable

$(X1 + X2)(\overline{X1} + \overline{X2})$ (sommet A)
$(X3 + X4)(\overline{X3} + \overline{X4})$ (sommet B)
$(X5 + X6)(\overline{X5} + \overline{X6})$ (sommet C)
$(X7 + X8)(\overline{X7} + \overline{X8})$ (sommet D)
$(\overline{X1} + \overline{X3})(\overline{X2} + \overline{X4})$ (arête AB)
$(\overline{X1} + \overline{X7})(\overline{X2} + \overline{X8})$ (arête AD)
$(\overline{X7} + \overline{X5})(\overline{X8} + \overline{X6})$ (arête DC)
$(\overline{X3} + \overline{X5})(\overline{X4} + \overline{X6})$ (arête BC)
$(\overline{X1} + \overline{X5})(\overline{X2} + \overline{X6})$ (arête AC)

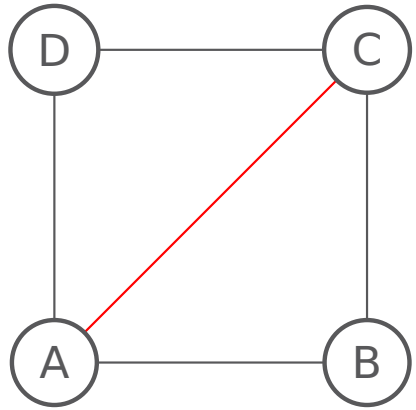


$$(\overline{X1} + \overline{X3})(\overline{X3} + \overline{X5})(\overline{X1} + \overline{X5})$$

$X1$	$X3$	$X5$	
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Formule non satisfaisable = graphe non 2-colorable

$(X1 + X2)(\overline{X1} + \overline{X2})$ (sommet A)
$(X3 + X4)(\overline{X3} + \overline{X4})$ (sommet B)
$(X5 + X6)(\overline{X5} + \overline{X6})$ (sommet C)
$(X7 + X8)(\overline{X7} + \overline{X8})$ (sommet D)
$(\overline{X1} + \overline{X3})(\overline{X2} + \overline{X4})$ (arête AB)
$(\overline{X1} + \overline{X7})(\overline{X2} + \overline{X8})$ (arête AD)
$(\overline{X7} + \overline{X5})(\overline{X8} + \overline{X6})$ (arête DC)
$(\overline{X3} + \overline{X5})(\overline{X4} + \overline{X6})$ (arête BC)
$(\overline{X1} + \overline{X5})(\overline{X2} + \overline{X6})$ (arête AC)

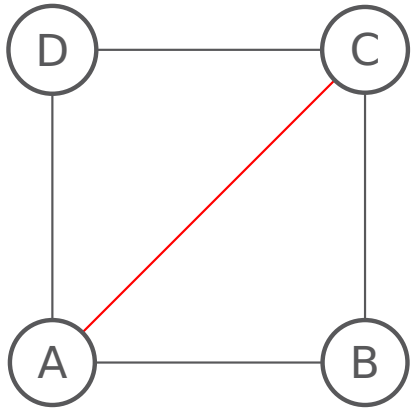


$$(\overline{X2} + \overline{X4})(\overline{X4} + \overline{X6})(\overline{X2} + \overline{X6})$$

$X2$	$X4$	$X6$	
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Formule non satisfaisable = graphe non 2-colorable

$(X1 + X2)(\overline{X1} + \overline{X2})$ (sommet A)
$(X3 + X4)(\overline{X3} + \overline{X4})$ (sommet B)
$(X5 + X6)(\overline{X5} + \overline{X6})$ (sommet C)
$(X7 + X8)(\overline{X7} + \overline{X8})$ (sommet D)
$(\overline{X1} + \overline{X3})(\overline{X2} + \overline{X4})$ (arête AB)
$(\overline{X1} + \overline{X7})(\overline{X2} + \overline{X8})$ (arête AD)
$(\overline{X7} + \overline{X5})(\overline{X8} + \overline{X6})$ (arête DC)
$(\overline{X3} + \overline{X5})(\overline{X4} + \overline{X6})$ (arête BC)
$(\overline{X1} + \overline{X5})(\overline{X2} + \overline{X6})$ (arête AC)



$X1$	$X3$	$X5$
0	0	0
0	0	1
0	1	0
1	0	0

$X2$	$X4$	$X6$
0	0	0
0	0	1
0	1	0
1	0	0

$X1$	$X2$
0	1
1	0

$X3$	$X4$
0	1
1	0

$X5$	$X6$
0	1
1	0

k-colorabilité se réduit à SAT

n sommets et m arêtes

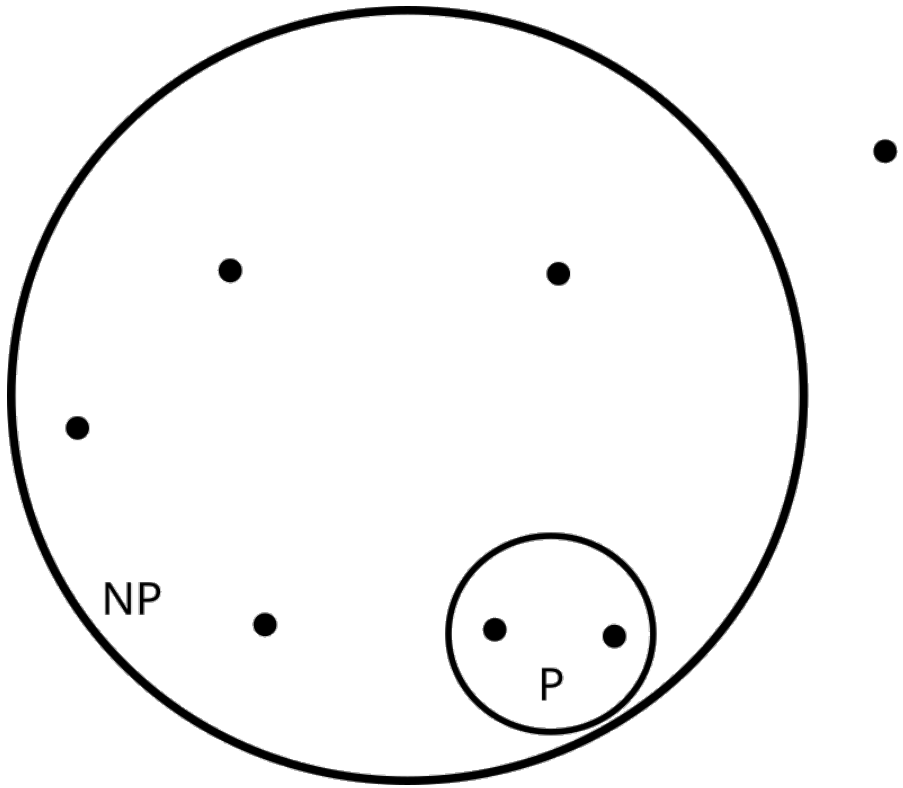
- kn variables $X_{1,1}, \ldots, X_{1,k}, \ldots, X_{n,1}, \ldots, X_{n,k}$
  - $X_{i,j} = 1$
    « le sommet i est de la couleur j »
- contraintes sur les sommets : n clauses de k littéraux et $n\frac{k(k-1)}{2}$ clauses de 2 littéraux
  - $X_{i,1} + \ldots + X_{i,k}$
    « le sommet i est d'une des k couleurs »
  - $\overline{X_{i,j_1} X_{1,j_2}} = \overline{X_{1,j_1}} + \overline{X_{1,j_2}}$ pour $j_1 < j_2$
    « le sommet i n'est pas de deux couleurs à la fois »
- contraintes sur les arêtes : km clauses de 2 littéraux
  - $(\overline{X_{i_1,1}} + \overline{X_{i_2,1}}) \ldots (\overline{X_{i_1,k}} + \overline{X_{i_2,k}})$ pour chaque arête i_1, i_2
    « deux sommets reliés ne sont pas de la même couleur »

⇝ formule en CNF construite en temps polynomial en $n + m$

Problèmes NP-complets

Définition (Problème NP-difficile et NP-complet)

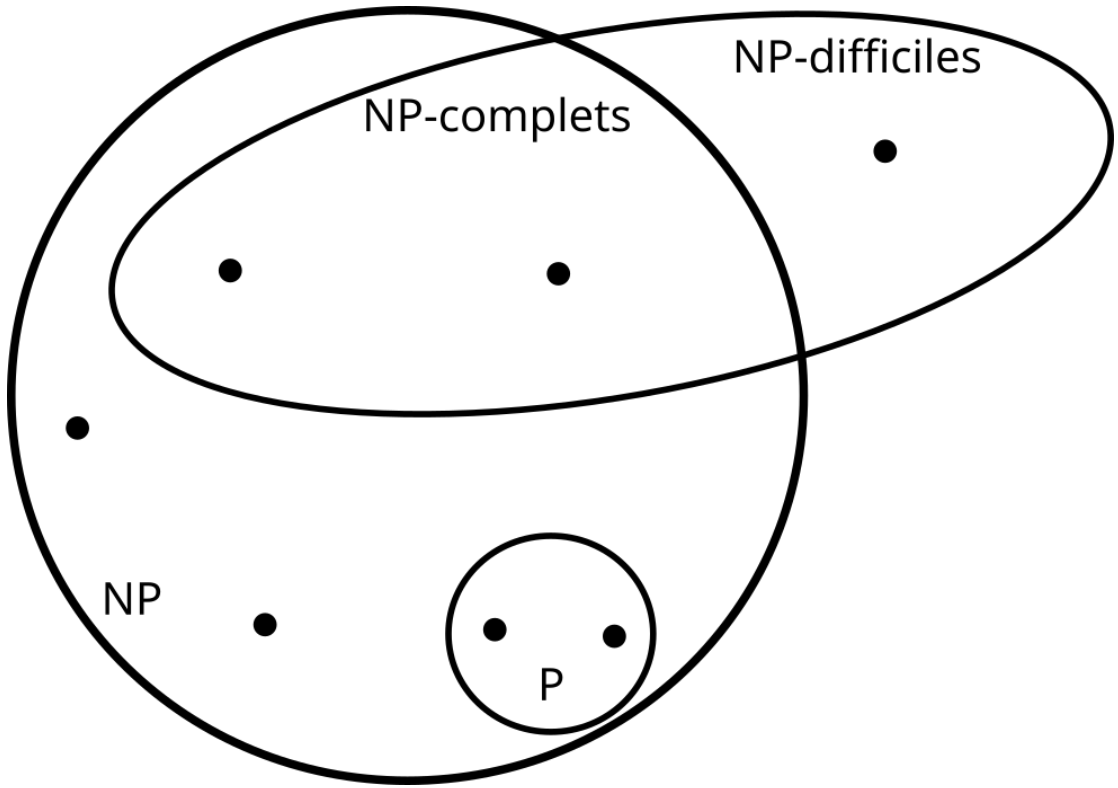
Un problème de décision P_1 est NP-difficile si tous les problèmes de NP ont une réduction polynomiale vers P_1.
Le problème de décision P_1 est NP-complet si il est dans NP et il est NP-difficile.



Problèmes NP-complets

Définition (Problème NP-difficile et NP-complet)

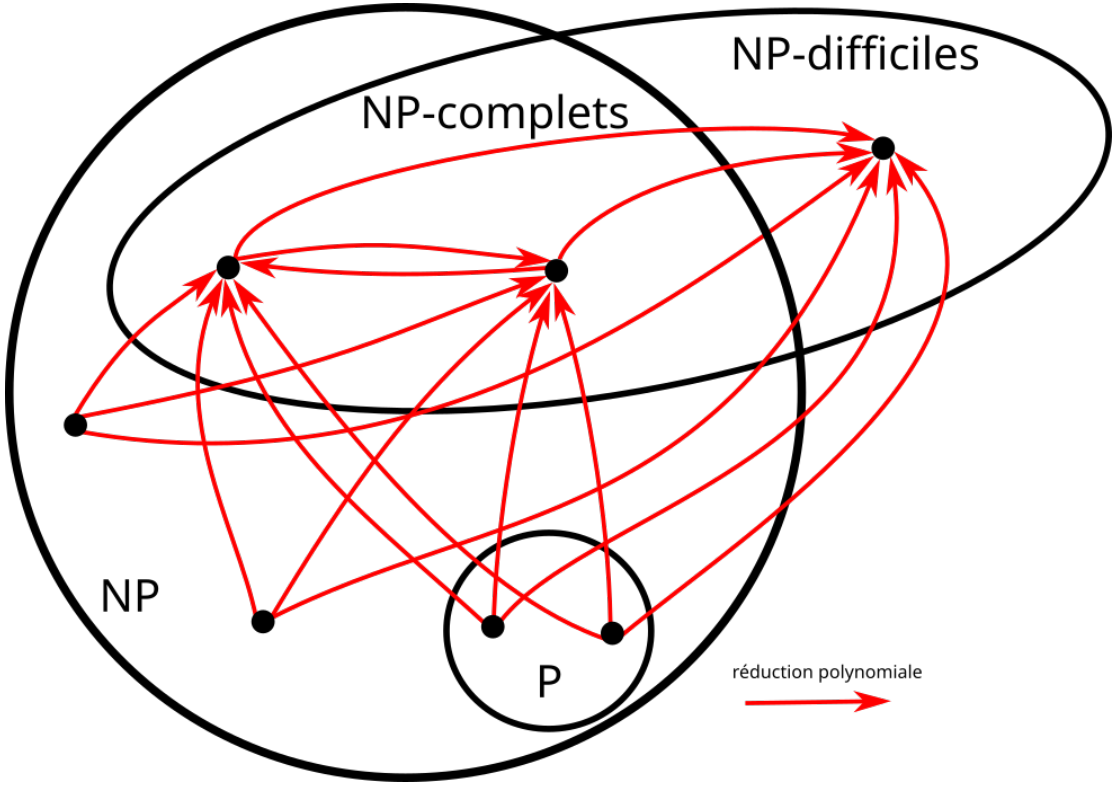
Un problème de décision P_1 est NP-difficile si tous les problèmes de NP ont une réduction polynomiale vers P_1.
Le problème de décision P_1 est NP-complet si il est dans NP et il est NP-difficile.



Problèmes NP-complets

Définition (Problème NP-difficile et NP-complet)

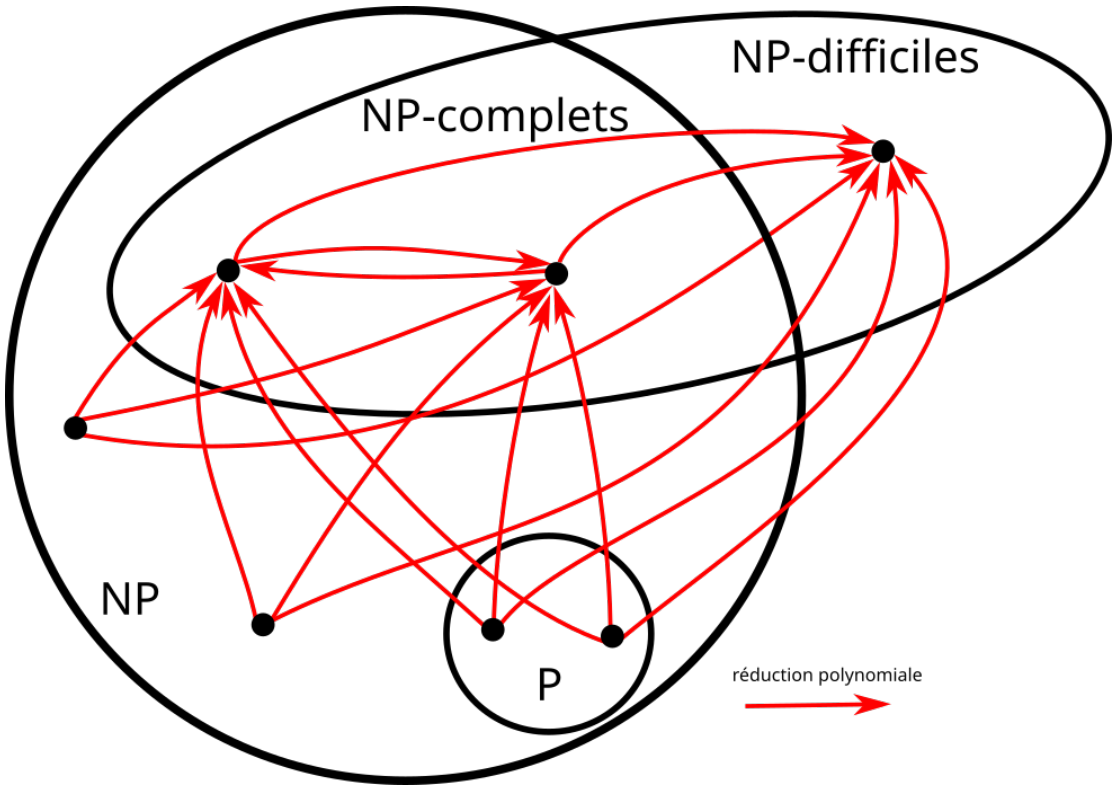
Un problème de décision P_1 est NP-difficile si tous les problèmes de NP ont une réduction polynomiale vers P_1.
Le problème de décision P_1 est NP-complet si il est dans NP et il est NP-difficile.



Problèmes NP-complets

Définition (Problème NP-difficile et NP-complet)

Un problème de décision P_1 est NP-difficile si tous les problèmes de NP ont une réduction polynomiale vers P_1.
Le problème de décision P_1 est NP-complet si il est dans NP et il est NP-difficile.



Intuitivement : un problème NP-difficile est “au moins aussi difficile” que tous les problèmes de NP (à facteur polynomial près).

Le premier problème NP-complet

Théorème (Cook-Levin, 1971)

*Le problème **SAT** est NP-complet.*

Stephen Cook (1939-)

Leonid Levin (1948-)

Richard C. Karp (1935-)

Le premier problème NP-complet

Théorème (Cook-Levin, 1971)

*Le problème **SAT** est NP-complet.*

Stephen Cook (1939-)

Leonid Levin (1948-)

Richard C. Karp (1935-)

Idée de la preuve : on encode n'importe quelle machine de Turing dans une formule booléenne.

Le premier problème NP-complet

Théorème (Cook-Levin, 1971)

*Le problème **SAT** est NP-complet.*

Stephen Cook (1939-)

Leonid Levin (1948-)

Richard C. Karp (1935-)

Idée de la preuve : on encode n'importe quelle machine de Turing dans une formule booléenne.

3-SAT
Instance: Une formule booléenne F en 3-CNF
Question: Est-ce que F est satisfaisable ?

Exemple : $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_2} \vee \overline{x_1}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_2 \vee x_3)$

Solution possible : x_1 =vrai, x_2 =faux, x_3 =vrai.

Le premier problème NP-complet

Théorème (Cook-Levin, 1971)

*Le problème **SAT** est NP-complet.*

Stephen Cook (1939-)

Leonid Levin (1948-)

Richard C. Karp (1935-)

Idée de la preuve : on encode n'importe quelle machine de Turing dans une formule booléenne.

3-SAT
Instance: Une formule booléenne F en 3-CNF
Question: Est-ce que F est satisfaisable ?

Exemple : $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_2} \vee \overline{x_1}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_2 \vee x_3)$

Solution possible : x_1 =vrai, x_2 =faux, x_3 =vrai.

Théorème (Karp, 1972)

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

Le premier problème NP-complet

Théorème (Cook-Levin, 1971)

*Le problème **SAT** est NP-complet.*

Stephen Cook (1939-)

Leonid Levin (1948-)

Richard C. Karp (1935-)

Idée de la preuve : on encode n'importe quelle machine de Turing dans une formule booléenne.

3-SAT
Instance: Une formule booléenne F en 3-CNF
Question: Est-ce que F est satisfaisable ?

Exemple : $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_2} \vee \overline{x_1}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_2 \vee x_3)$
Solution possible : x_1 =vrai, x_2 =faux, x_3 =vrai.

Théorème (Karp, 1972)

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

Conséquence

*Si on trouve un algorithme polynomial pour **SAT** ou **3-SAT**, alors il en existe un pour chaque problème de NP ! (et on gagne 1 million de $)*

Comment montrer qu'un problème est NP-difficile ?

Rappel : P_1 est NP-difficile = tous les problèmes de NP ont une réduction polynomiale vers P_1

Proposition

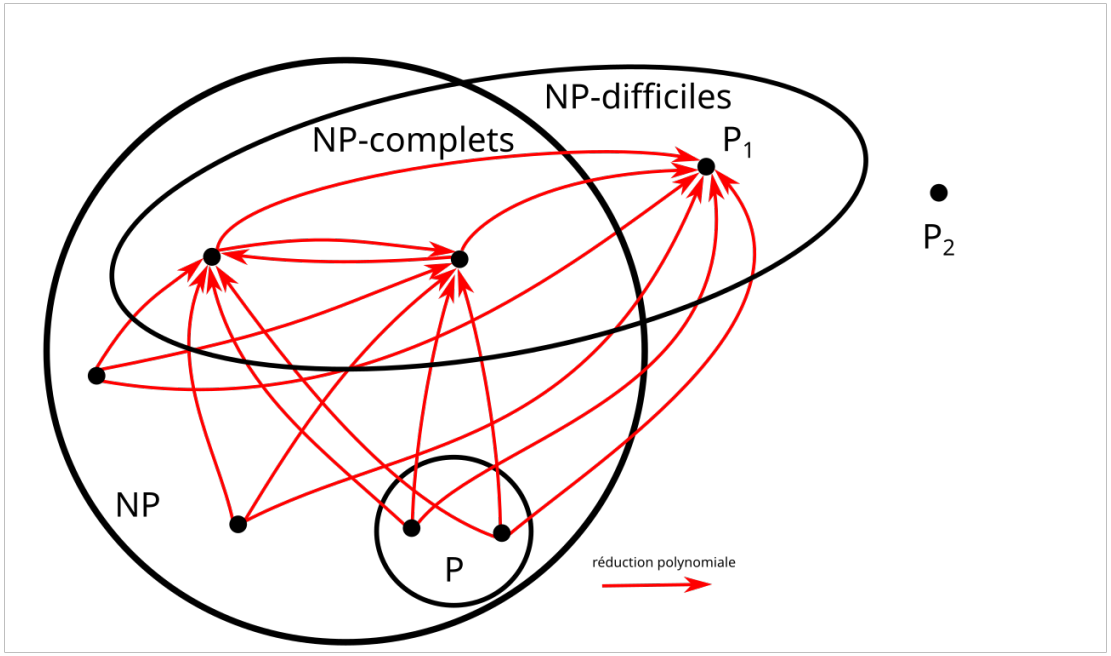
Si P_1 est NP-difficile et P_1 a une réduction polynomiale vers P_2, alors P_2 est aussi NP-difficile.

Comment montrer qu'un problème est NP-difficile ?

Rappel : P_1 est NP-difficile = tous les problèmes de NP ont une réduction polynomiale vers P_1

Proposition

Si P_1 est NP-difficile et P_1 a une réduction polynomiale vers P_2, alors P_2 est aussi NP-difficile.

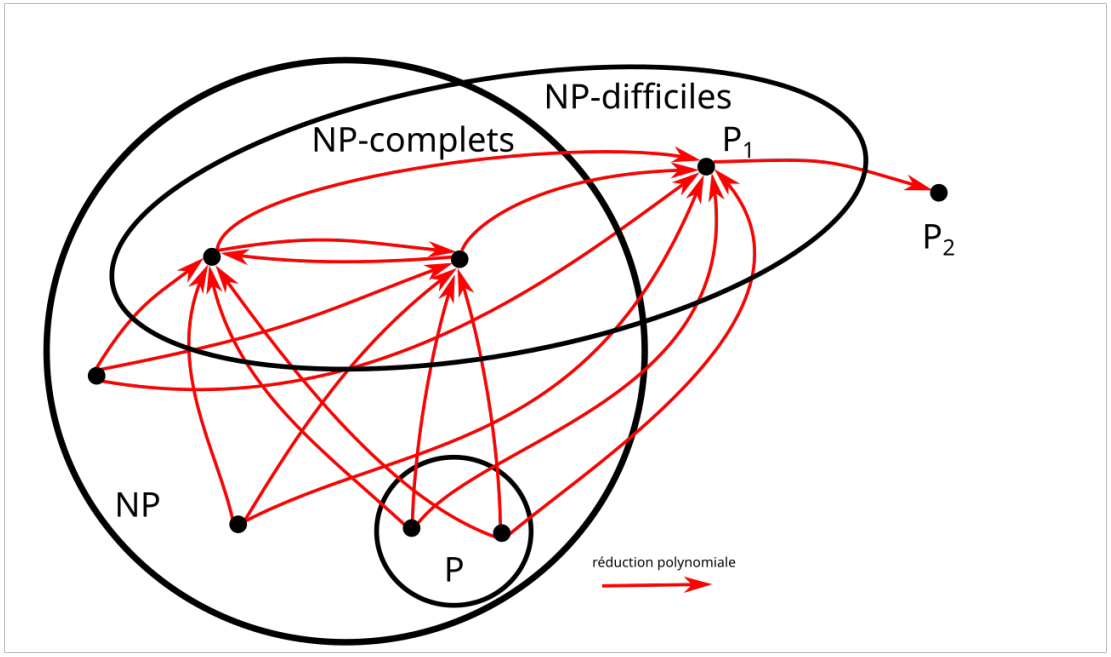


Comment montrer qu'un problème est NP-difficile ?

Rappel : P_1 est NP-difficile = tous les problèmes de NP ont une réduction polynomiale vers P_1

Proposition

Si P_1 est NP-difficile et P_1 a une réduction polynomiale vers P_2, alors P_2 est aussi NP-difficile.

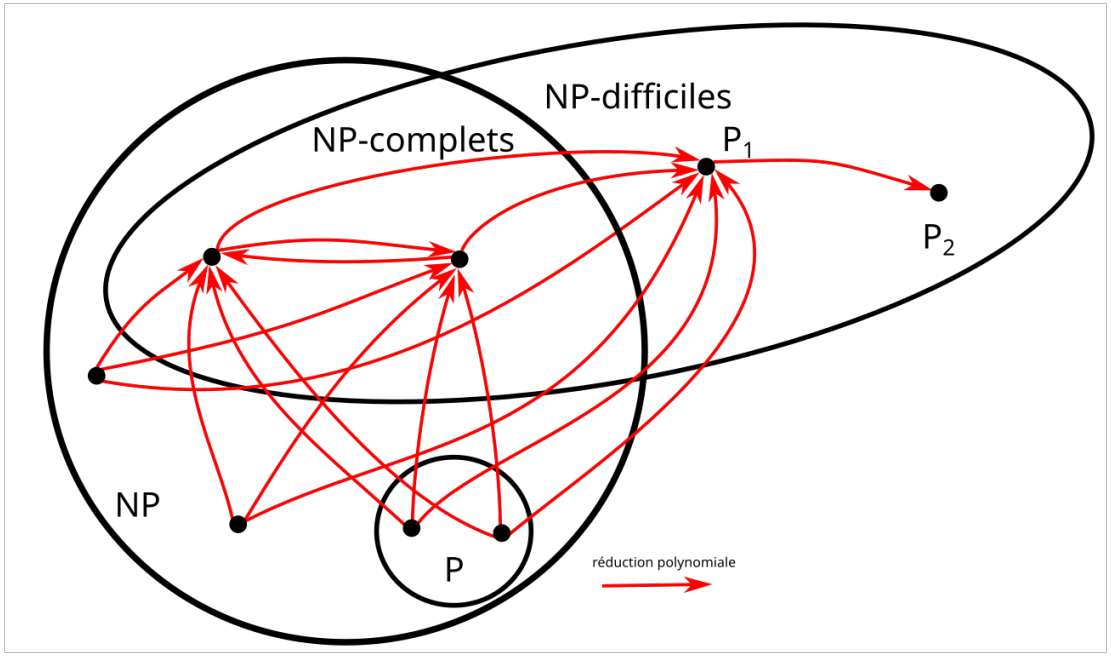


Comment montrer qu'un problème est NP-difficile ?

Rappel : P_1 est NP-difficile = tous les problèmes de NP ont une réduction polynomiale vers P_1

Proposition

Si P_1 est NP-difficile et P_1 a une réduction polynomiale vers P_2, alors P_2 est aussi NP-difficile.



Une réduction de 3-SAT vers 3-Coloration

3-SAT
Instance: Une formule booléenne F en 3-CNF
Question: Est-ce que F est satisfaisable ?

Exemple : $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$

3-Coloration
Instance: Un graphe G
Question: Est-ce que G est 3-colorable ?

Théorème (Garey, Johnson, Stockmeyer, 1976)

3-Coloration *est NP-complet.*

Une réduction de 3-SAT vers 3-Coloration

3-SAT
Instance: Une formule booléenne F en 3-CNF
Question: Est-ce que F est satisfaisable ?

Exemple : $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$

3-Coloration
Instance: Un graphe G
Question: Est-ce que G est 3-colorable ?

Théorème (Garey, Johnson, Stockmeyer, 1976)

3-Coloration *est NP-complet.*

1) **3-Coloration** est dans NP : pour une 3-coloration d'un graphe G, on peut vérifier en temps polynomial si elle est correcte (pour chaque arête, on vérifie que les deux extrémités ont une couleur différente).

Une réduction de 3-SAT vers 3-Coloration

3-SAT
Instance: Une formule booléenne F en 3-CNF
Question: Est-ce que F est satisfaisable ?

Exemple : $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$

3-Coloration
Instance: Un graphe G
Question: Est-ce que G est 3-colorable ?

Théorème (Garey, Johnson, Stockmeyer, 1976)

3-Coloration *est NP-complet.*

2) **3-Coloration** est NP-difficile : on construit une réduction polynomiale de **3-SAT** vers **3-Coloration** : pour toute formule F en 3-CNF, on veut un graphe $G(F)$ tel que F est satisfaisable $\Leftrightarrow$ $G(F)$ est 3-colorable.

Une réduction de 3-SAT vers 3-Coloration

3-SAT
Instance: Une formule booléenne F en 3-CNF
Question: Est-ce que F est satisfaisable ?

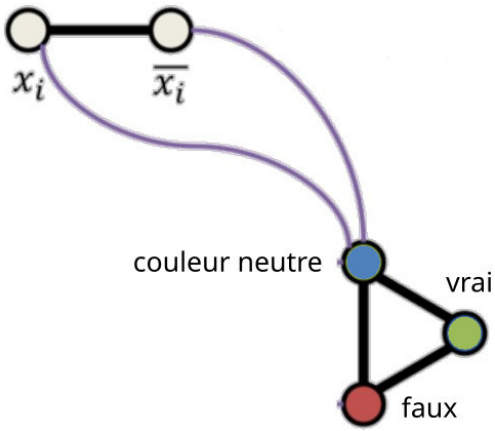
Exemple : $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$

3-Coloration
Instance: Un graphe G
Question: Est-ce que G est 3-colorable ?

Théorème (Garey, Johnson, Stockmeyer, 1976)

3-Coloration *est NP-complet.*

2.1) on modélise une variable x_i :



Une réduction de 3-SAT vers 3-Coloration

3-SAT
Instance: Une formule booléenne F en 3-CNF
Question: Est-ce que F est satisfaisable ?

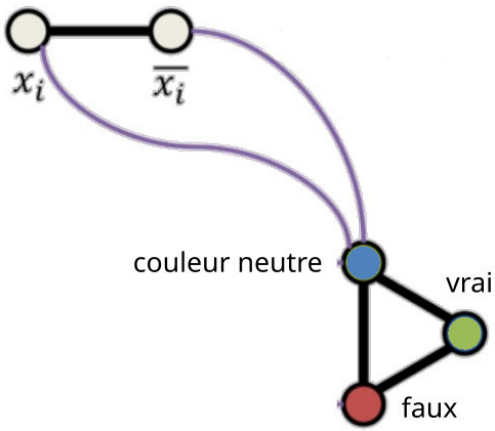
Exemple : $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$

3-Coloration
Instance: Un graphe G
Question: Est-ce que G est 3-colorable ?

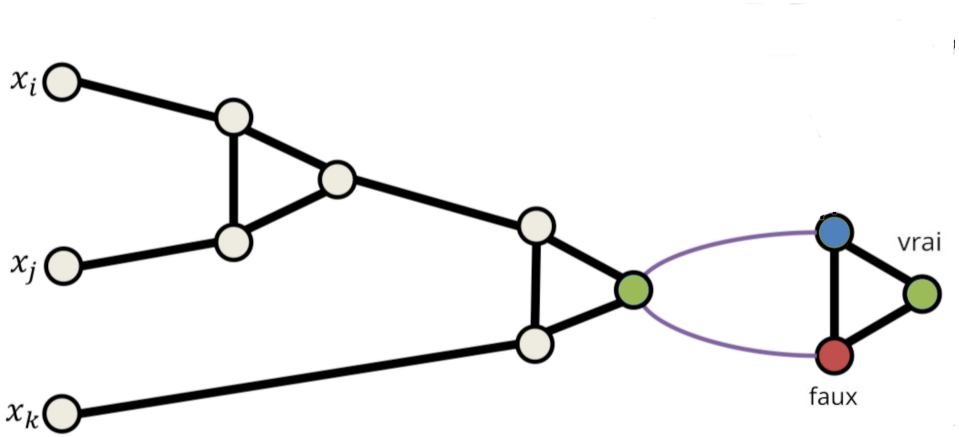
Théorème (Garey, Johnson, Stockmeyer, 1976)

3-Coloration *est NP-complet.*

2.1) on modélise une variable x_i :



2.2) on modélise une clause $(x_i \vee x_j \vee x_k)$:



Une réduction de 3-SAT vers 3-Coloration

3-SAT
Instance: Une formule booléenne F en 3-CNF
Question: Est-ce que F est satisfaisable ?

Exemple : $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$

3-Coloration
Instance: Un graphe G
Question: Est-ce que G est 3-colorable ?

Théorème (Garey, Johnson, Stockmeyer, 1976)

3-Coloration *est NP-complet.*

2.3) on met tout ensemble !

Une réduction de 3-SAT vers 3-Coloration

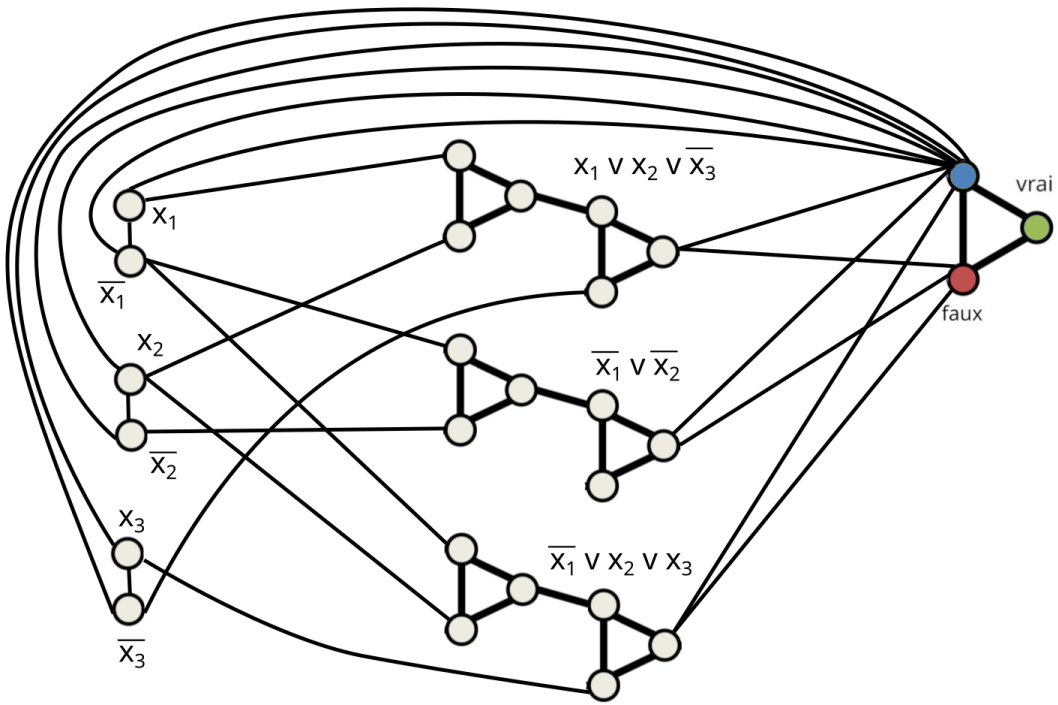
3-SAT
Instance: Une formule booléenne F en 3-CNF
Question: Est-ce que F est satisfaisable ?

Exemple : $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$

3-Coloration
Instance: Un graphe G
Question: Est-ce que G est 3-colorable ?

Théorème (Garey, Johnson, Stockmeyer, 1976)

3-Coloration *est NP-complet.*



Une réduction de 3-SAT vers 3-Coloration

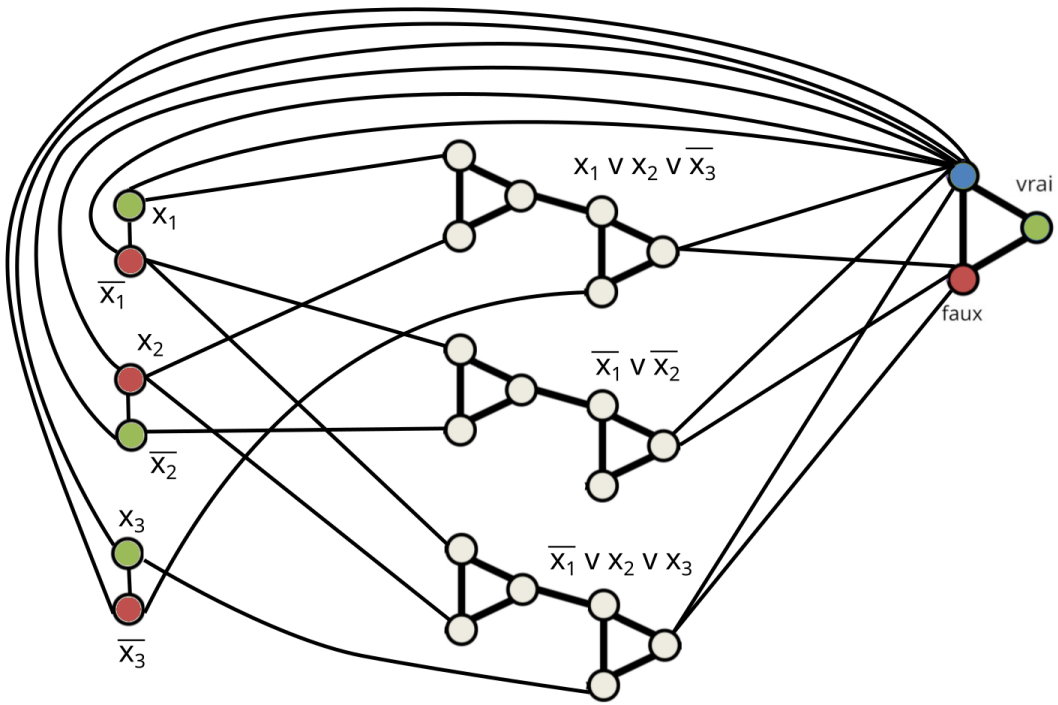
3-SAT
Instance: Une formule booléenne F en 3-CNF
Question: Est-ce que F est satisfaisable ?

Exemple : $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$

3-Coloration
Instance: Un graphe G
Question: Est-ce que G est 3-colorable ?

Théorème (Garey, Johnson, Stockmeyer, 1976)

3-Coloration *est NP-complet.*



Si F est satisfaisable,
alors $G(F)$ est 3-colorable

Une réduction de 3-SAT vers 3-Coloration

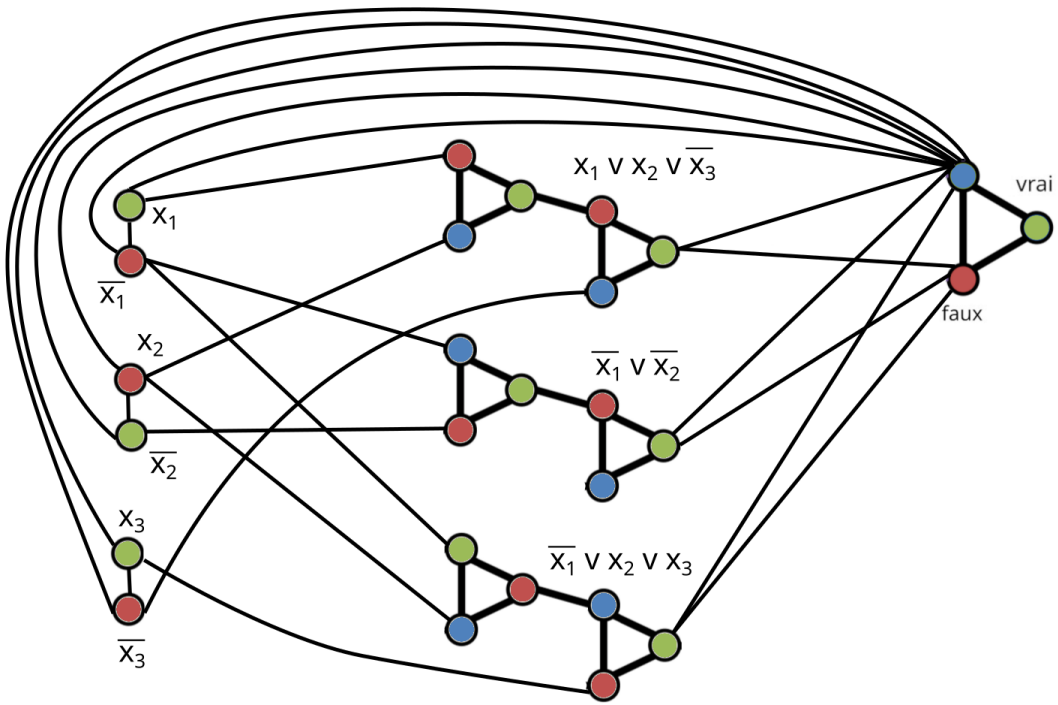
3-SAT
Instance: Une formule booléenne F en 3-CNF
Question: Est-ce que F est satisfaisable ?

Exemple : $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$

3-Coloration
Instance: Un graphe G
Question: Est-ce que G est 3-colorable ?

Théorème (Garey, Johnson, Stockmeyer, 1976)

3-Coloration *est NP-complet.*



Si F est satisfaisable,
alors $G(F)$ est 3-colorable

Une réduction de 3-SAT vers 3-Coloration

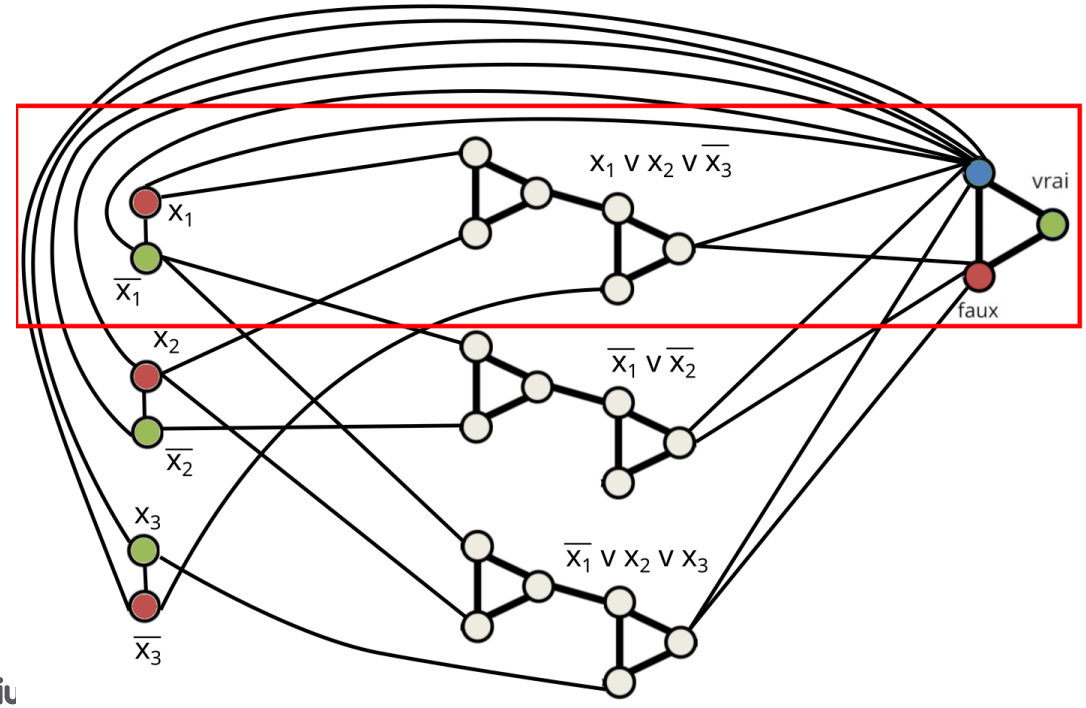
3-SAT
Instance: Une formule booléenne F en 3-CNF
Question: Est-ce que F est satisfaisable ?

Exemple : $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$

3-Coloration
Instance: Un graphe G
Question: Est-ce que G est 3-colorable ?

Théorème (Garey, Johnson, Stockmeyer, 1976)

3-Coloration *est NP-complet.*



Si $G(F)$ est 3-colorable, alors F est satisfaisable

Une réduction de 3-SAT vers 3-Coloration

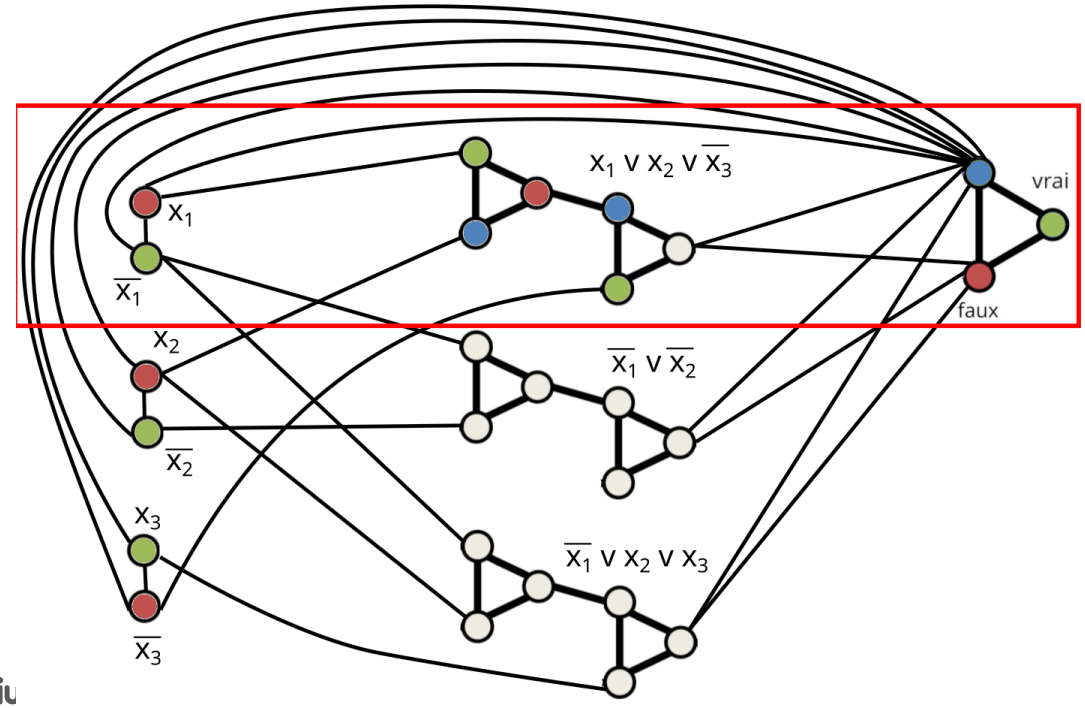
3-SAT
Instance: Une formule booléenne F en 3-CNF
Question: Est-ce que F est satisfaisable ?

Exemple : $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$

3-Coloration
Instance: Un graphe G
Question: Est-ce que G est 3-colorable ?

Théorème (Garey, Johnson, Stockmeyer, 1976)

3-Coloration *est NP-complet.*



Si $G(F)$ est 3-colorable, alors F est satisfaisable

Une réduction de 3-SAT vers 3-Coloration

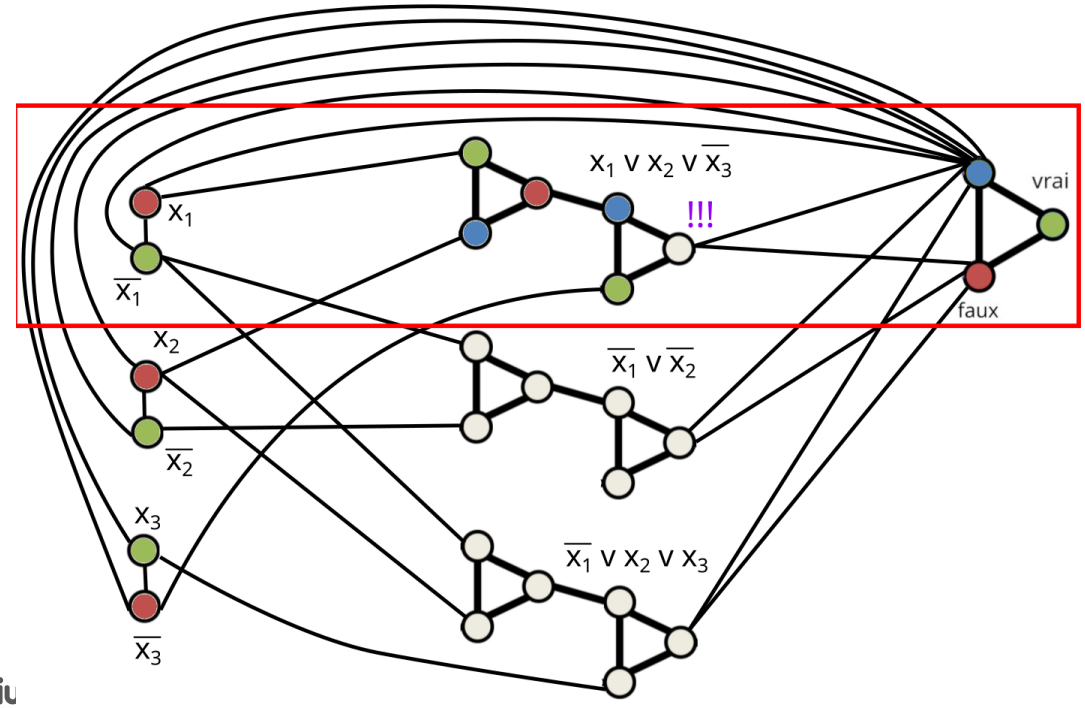
3-SAT
Instance: Une formule booléenne F en 3-CNF
Question: Est-ce que F est satisfaisable ?

Exemple : $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$

3-Coloration
Instance: Un graphe G
Question: Est-ce que G est 3-colorable ?

Théorème (Garey, Johnson, Stockmeyer, 1976)

3-Coloration *est NP-complet.*



Si $G(F)$ est 3-colorable,
alors F est satisfaisable

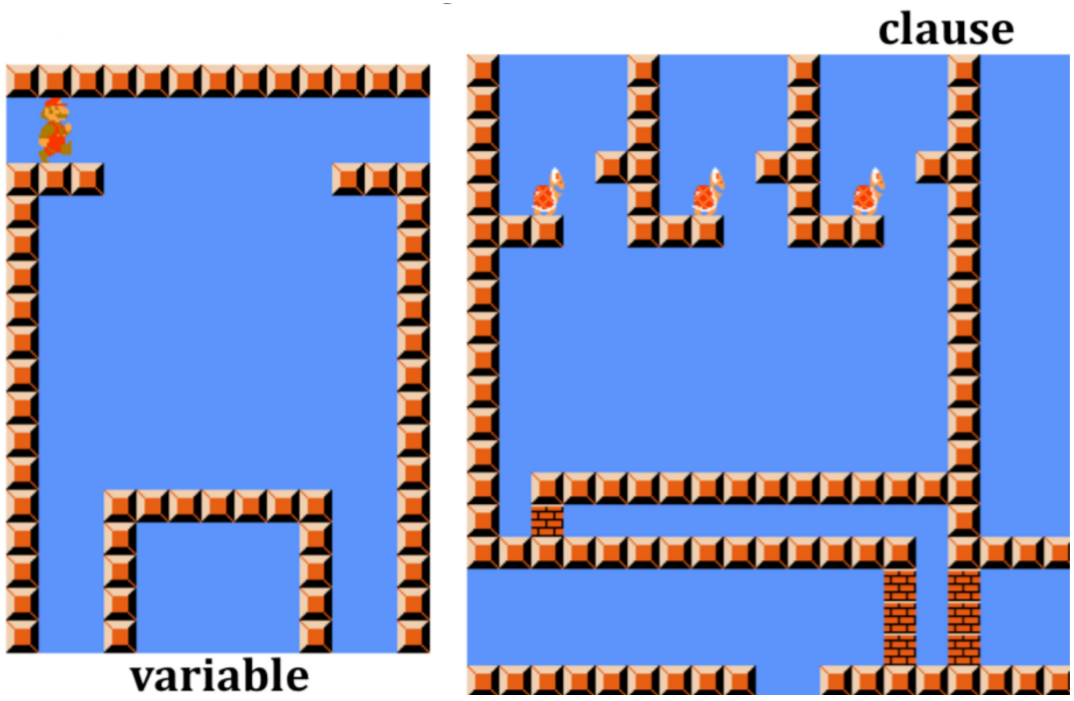
Super Mario

Super Mario
Instance: Un niveau de Super Mario.
Question: Est-ce que Mario peut aller du départ à l'arrivée ?

Théorème (Aloupis, Demaine, Guo, 2012)

Super Mario *est NP-difficile.*

Réduction depuis **3-SAT** :



Conclusion générale

- Tout algorithme n’est pas forcément efficace/termine/correct !
- Tout problème algorithmique n’a pas forcément une solution simple !
- Diverses techniques algorithmiques, plus ou moins avancées
- De nombreuses autres techniques existent
- Complexité algorithmique : fondement de l’informatique
- Culture importante pour tout informaticien