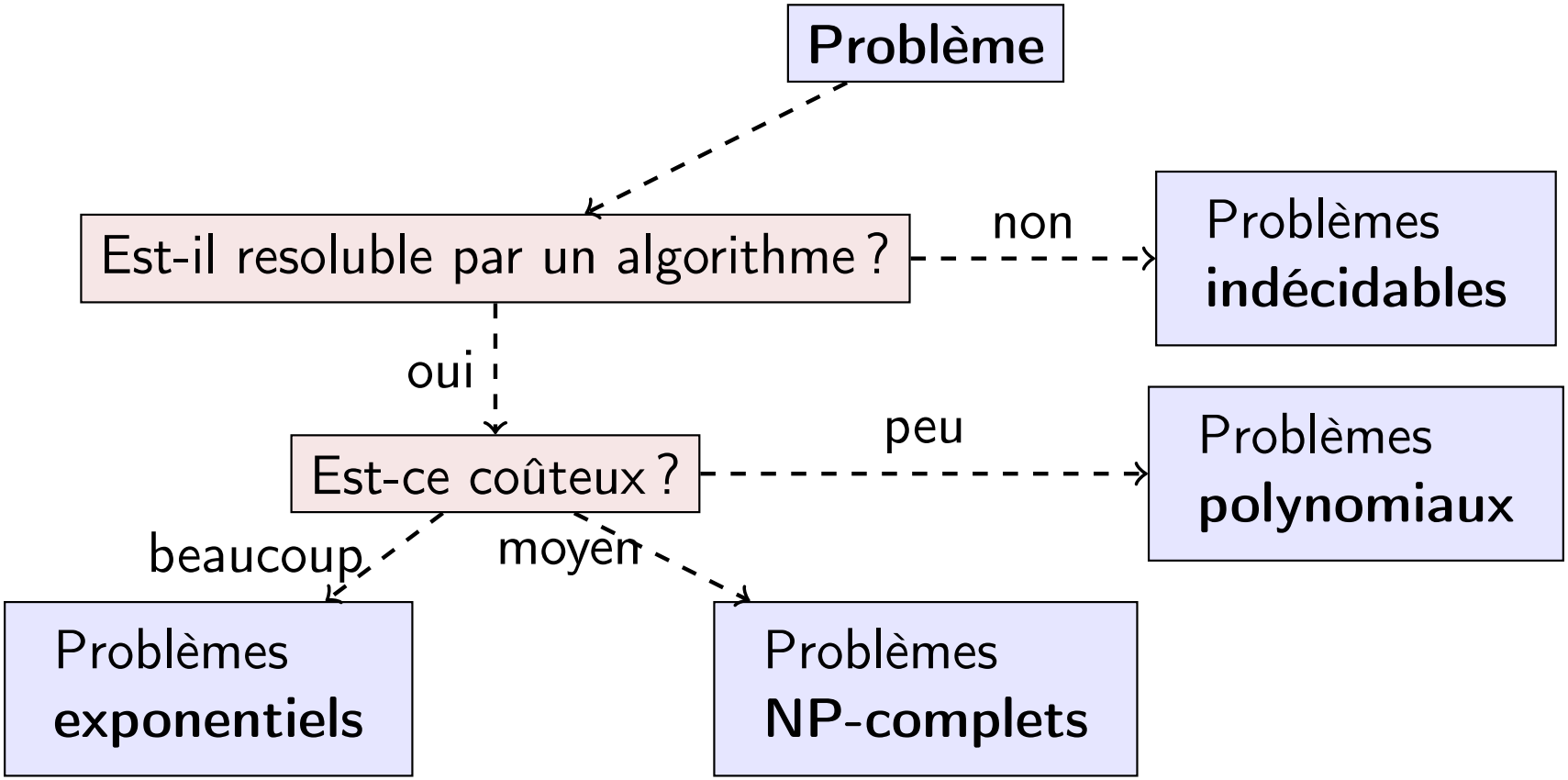
Calculabilité et NP-Complétude

Partie III : Problèmes NP-complet et SAT-solvers
Master 2

Architecture du cours (schématique)



Métaheuristiques, approximations, etc.

Solveurs SAT

Solveurs IP (VS)

Plan

NP-complet

Un problème B est **NP**-complet si

- Il est dans **NP**,
- Pour tout problème A de **NP**,

$$A <<_{\mathbf{P}} B$$

Le second point est difficile à monter, il y a une infinité de problèmes dans **NP**. En général, on montrer que

- Il est dans **NP**,
- Il existse un problème C **NP**-complet tel que ,

$$C <<_{\mathbf{P}} B$$

SAT

SAT

Donnée : Une formule booléenne $\varphi(x_1, \ldots, x_k)$.
Questions : Peut-on trouver $x_1, \ldots, x_k$ tels que φ soit vraie ?

Théorème (Cook 73)

SAT est **NP**-complet.

Exercice, tables de vérité

On se propose de résoudre naïvement le problème SAT en calculant la table de vérité de la formule. On suppose que l'on dispose pour cela de 10000 ordinateurs en réseau qui vont chacun calculer un morceau de la table de vérité. On suppose aussi que chaque ordinateur a une vitesse de 10 GHz et qu'il peut évaluer la formule en un seul cycle d'horloge.

1. Combien de lignes a la table de vérité de la formule φ ? (en fonction de k).
2. On suppose que $k = 10$, en combien de temps le problème est-il résolu ?
3. Même question avec $k = 20, 30, 50, 80$?

On rappelle que 2^{10} est environ égal à 10^3 . On rappelle aussi qu'il 86400 secondes dans une journée et environ 3 10^7 secondes dans une année.

SAT est dans **NP**

Partie facile : SAT est dans **NP**.

```
def SAT(phi,k):
    x=[]
    for i in range(k):
        x.append(Choisir([True,False]))
    return phi(x)
```

Evaluer une formule booléenne se fait en temps polynomial.

SAT est **NP**-difficile

Codage d'une machine de Turing M non déterministe polynomiale $P(n)$ où n longueur du mot d'entrée w, décidant un problème A (donc dans **NP**).

- On peut restreindre le ruban pour $2 * P(n)$ cases.
- Si un mot est accepté, il l'est avec un calcul d'au plus $P(n)$ étapes.

SAT est **NP**-difficile

Codage d'une machine de Turing M non déterministe polynomiale $P(n)$ où n longueur du mot d'entrée w, décidant un problème A (donc dans **NP**).

- On peut restreindre le ruban pour $2 * P(n)$ cases.
- Si un mot est accepté, il l'est avec un calcul d'au plus $P(n)$ étapes.

On va tout coder avec des variables booléennes dans une formule φ_w telle que

- w est reconnu par M ssi φ_w est satisfiable,
- φ_w se calcule en temps polynomial par rapport à n.

On motre que

$$A <<_{\mathbf{P}} SAT$$

SAT est **NP**-difficile

On va coder dans φ_w l'existence d'un calcul acceptant pour w. Les variables propositionnelles sont :

SAT est **NP**-difficile

On va coder dans φ_w l'existence d'un calcul acceptant pour w. Les variables propositionnelles sont :

- $T_{i,k}$ est vraie ssi à l'étape k d'un calcul réussi, la tête pointe sur la case i du ruban. On a $O(P(n)^2)$ variables.

SAT est **NP**-difficile

On va coder dans φ_w l'existence d'un calcul acceptant pour w. Les variables propositionnelles sont :

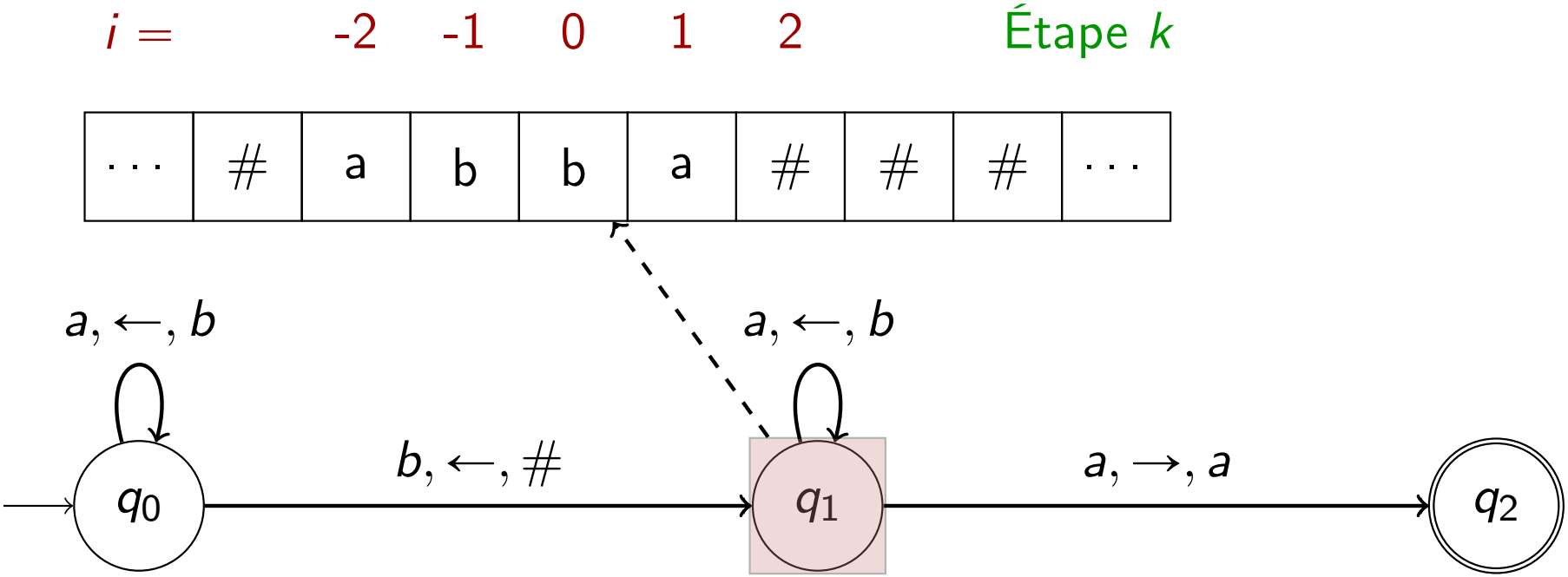
- $T_{i,k}$ est vraie ssi à l'étape k d'un calcul réussi, la tête pointe sur la case i du ruban. On a $O(P(n)^2)$ variables.
- $E_{q,k}$ est vraie ssi à l'étape k d'un calcul réussi, la tête est dans l'état q de la machine. On a $O(P(n))$ variables.

SAT est **NP**-difficile

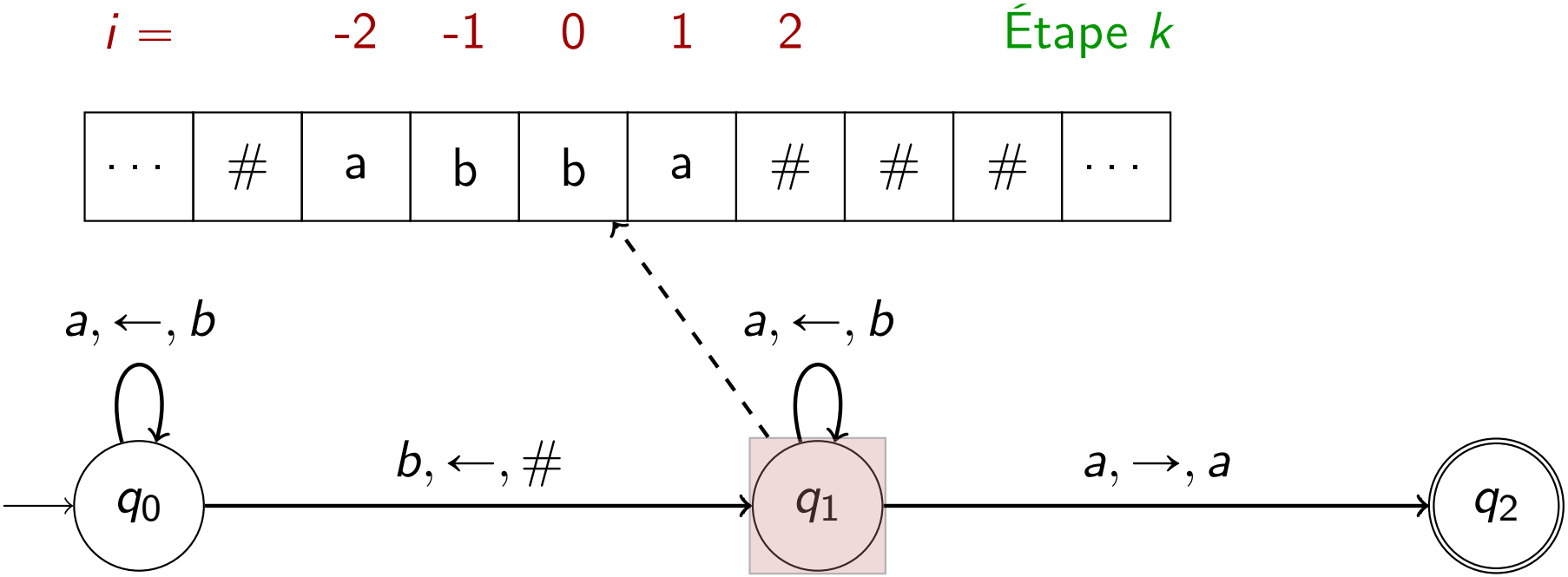
On va coder dans φ_w l'existence d'un calcul acceptant pour w. Les variables propositionnelles sont :

- $T_{i,k}$ est vraie ssi à l'étape k d'un calcul réussi, la tête pointe sur la case i du ruban. On a $O(P(n)^2)$ variables.
- $E_{q,k}$ est vraie ssi à l'étape k d'un calcul réussi, la tête est dans l'état q de la machine. On a $O(P(n))$ variables.
- $R_{i,\alpha,k}$ est vraie ssi à l'étape k d'un calcul réussi, la case i du ruban contient la lettre α. On a $O(P(n)^2)$ variables.

SAT est **NP**-difficile

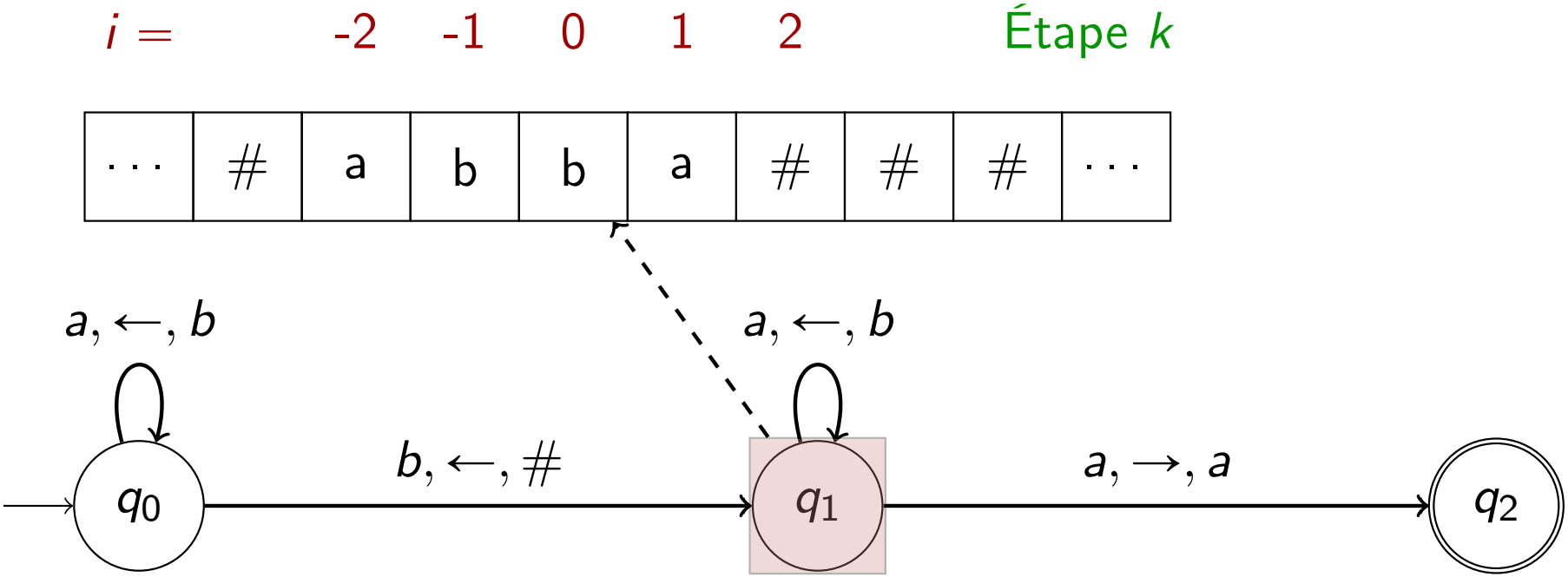


SAT est **NP**-difficile



Pour un calcul réussi sur le mot d'entrée, on devra avoir dans la formule : $E_{q_1,k}$ est Vraie. $E_{q_0,k}$ est Fausse. $E_{q_3,k}$ est Fausse.

SAT est **NP**-difficile



Pour un calcul réussi sur le mot d'entrée, on devra avoir dans la formule :
$E_{q_1,k}$ est Vraie. $E_{q_0,k}$ est Fausse. $E_{q_3,k}$ est Fausse.
$T_{0,k}$ est Vraie. $T_{1,k}$ est Fausse. $T_{-1,k}$ est Fausse...
$R_{0,b,k}$ est Vraie. $R_{0,a,k}$ est Fausse. $R_{0,\#,k}$ est Fausse. $R_{1,a,k}$ est Vraie...

SAT est **NP**-difficile : codage de la tête de lecture

$T_{i,k}$ est vraie ssi à l'étape k d'un calcul réussi, la tête pointe sur la case i du ruban. On a $O(P(n)^2)$ variables.

SAT est **NP**-difficile : codage de la tête de lecture

$T_{i,k}$ est vraie ssi à l'étape k d'un calcul réussi, la tête pointe sur la case i du ruban. On a $O(P(n)^2)$ variables.

La tête de lecture ne peut pas pointer sur deux cases : à l'étape k, si la tête pointe sur la case i, elle ne pointe par sur une case $j \neq i$

$$T_{i,k} \Rightarrow \neg T_{j,k}$$

SAT est **NP**-difficile : codage de la tête de lecture

$T_{i,k}$ est vraie ssi à l'étape k d'un calcul réussi, la tête pointe sur la case i du ruban. On a $O(P(n)^2)$ variables.

La tête de lecture ne peut pas pointer sur deux cases : à l'étape k, si la tête pointe sur la case i, elle ne pointe par sur une case $j \neq i$

$$T_{i,k} \Rightarrow \neg T_{j,k}$$

$$\bigwedge_{k} \bigwedge_{i} \bigwedge_{j \neq i} (T_{i,k} \Rightarrow \neg T_{j,k})$$

SAT est **NP**-difficile : codage de la tête de lecture

$T_{i,k}$ est vraie ssi à l'étape k d'un calcul réussi, la tête pointe sur la case i du ruban. On a $O(P(n)^2)$ variables.

La tête de lecture ne peut pas pointer sur deux cases : à l'étape k, si la tête pointe sur la case i, elle ne pointe par sur une case $j \neq i$

$$T_{i,k} \Rightarrow \neg T_{j,k}$$

$$\bigwedge_k \bigwedge_i \bigwedge_{j \neq i} (T_{i,k} \Rightarrow \neg T_{j,k})$$

La tête pointe sur au moins une case :

$$\bigwedge_k \bigvee_i T_{i,k}$$

SAT est **NP**-difficile : codage de la tête de lecture

$T_{i,k}$ est vraie ssi à l'étape k d'un calcul réussi, la tête pointe sur la case i du ruban. On a $O(P(n)^2)$ variables.

La tête de lecture ne peut pas pointer sur deux cases : à l'étape k, si la tête pointe sur la case i, elle ne pointe par sur une case $j \neq i$

$$T_{i,k} \Rightarrow \neg T_{j,k}$$

$$\bigwedge_k \bigwedge_i \bigwedge_{j \neq i} (T_{i,k} \Rightarrow \neg T_{j,k})$$

La tête pointe sur au moins une case :

$$\bigwedge_k \bigvee_i T_{i,k}$$

$$\varphi_T = \left(\bigwedge_k \bigvee_i T_{i,k} \right) \wedge \left(\bigwedge_k \bigwedge_i \bigwedge_{j \neq i} (T_{i,k} \Rightarrow \neg T_{j,k}) \right)$$

SAT est **NP**-difficile : état

$E_{q,k}$ est vraie ssi à l'étape k d'un calcul réussi, la tête est dans l'état q de la machine. On a $O(P(n))$ variables.

La machine est à chaque étape dans un et un seul état :

$$\varphi_E = \left(\bigwedge_k \bigvee_q E_{q,k} \right) \wedge \left(\bigwedge_k \bigwedge_q \bigwedge_{q' \neq q} (E_{q,k} \Rightarrow \neg E_{q',k}) \right)$$

La formule a une taille dans $O(P(n))$ (le nombre d'états est une constante)

SAT est **NP**-difficile : le ruban

$R_{i,\alpha,k}$ est vraie ssi à l'étape k d'un calcul réussi, la case i du ruban contient la lettre α. On a $O(P(n)^2)$ variables.

Chaque case contient une et une seule lettre à chaque étape :

$$\varphi_R = \left(\bigwedge_k \bigwedge_i \bigvee_\alpha R_{i,\alpha,k} \right) \wedge \left(\bigwedge_k \bigwedge_i \bigwedge_\alpha \bigwedge_{\alpha' \neq \alpha} (R_{q,\alpha k} \Rightarrow \neg R_{i,\alpha',k}) \right)$$

La formule a une taille dans $O(P(n)^2)$ (le nombre de lettres est une constante)

SAT est **NP**-difficile : configuration initiale

SAT est **NP**-difficile : configuration initiale

Le mot w est écrit sur le ruban :

$$\left(\bigwedge_{r=1}^{|w|} R_{r,w(r),0}\right) \wedge \left(\bigwedge_{r<0, r>|w|} R_{r,\#,0}\right)$$

SAT est **NP**-difficile : configuration initiale

Le mot w est écrit sur le ruban :

$$\left(\bigwedge_{r=1}^{|w|} R_{r,w(r),0}\right) \wedge \left(\bigwedge_{r<0,r>|w|} R_{r,\#,0}\right)$$

La machine est dans un état initial :

$$\bigvee_{q \in I} E_{q,0}$$

SAT est **NP**-difficile : configuration initiale

Le mot w est écrit sur le ruban :

$$\left(\bigwedge_{r=1}^{|w|} R_{r,w(r),0}\right) \wedge \left(\bigwedge_{r<0,r>|w|} R_{r,\#,0}\right)$$

La machine est dans un état initial :

$$\bigvee_{q \in I} E_{q,0}$$

SAT est **NP**-difficile : configuration initiale

Le mot w est écrit sur le ruban :

$$\left(\bigwedge_{r=1}^{|w|} R_{r,w(r),0}\right) \wedge \left(\bigwedge_{r<0,r>|w|} R_{r,\#,0}\right)$$

La machine est dans un état initial :

$$\bigvee_{q \in I} E_{q,0}$$

$$\varphi_{\text{init}} = \left(\bigwedge_{r=1}^{|w|} R_{r,w(r),0}\right) \wedge \left(\bigwedge_{r<0,r>|w|} R_{r,\#,0}\right) \wedge \left(\bigvee_{q \in I} E_{q,0}\right) \wedge T_{1,0}$$

SAT est **NP**-difficile : configuration finale et consécutives

Le calcul passe par une confuration finale :

$$\varphi_{\text{final}} = \left(\bigvee_{k} \bigvee_{q \in F} E_{q,k} \right)$$

SAT est **NP**-difficile : configuration finale et consécutives

Le calcul passe par une confuration finale :

$$\varphi_{\text{final}} = \left(\bigvee_{k} \bigvee_{q \in F} E_{q,k} \right)$$

Les configurations sont consécutives :

$$\varphi_{\text{chemin}} = \bigwedge_{k} \bigwedge_{q} \bigwedge_{i} \bigwedge_{\alpha} \Bigg(E_{q,k} \wedge T_{i,k} \wedge R_{i,\alpha,k} \Rightarrow \Big(\bigvee_{q \xrightarrow{\alpha,\rightarrow,\alpha'} q'} \left(E_{q',k+1} \wedge T_{i+1,k+1} \wedge R_{i+1,\alpha',k+1} \right) \vee \bigvee_{q \xrightarrow{\alpha,\leftarrow,\alpha'} q'} \left(E_{q',k+1} \wedge T_{i-1,k+1} \wedge R_{i-1,\alpha',k+1} \right) \Big) \Bigg)$$

SAT est **NP**-difficile

La formule

$$\varphi_T \wedge \varphi_E \wedge \varphi_R \wedge \varphi_{\text{init}} \wedge \varphi_{final} \wedge \varphi_{\text{chemin}}$$

se calcule en temps polynomial en $|w|$ et est satisfiable si et seulement si w est décidé par la machine de Turing.

Donc

$$A <<_{\text{P}} SAT$$

SAT CNF **NP**-difficile

Une formule SAT est sous forme CNF *Cunjunctive Normal Form*, si elle est de la forme :

$$\bigwedge \text{Clauses}$$

où les clauses sont des formules du type

$$x_1 \vee x_3 \vee \neg x_6 \vee x_8 \vee \neg x_9$$

i.e. une disjonction de variables ou de leur négation.
Par exemple :

$$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_4) \wedge (x_3 \vee x_5 \vee \neg x_6)$$

est sous forme CNF.

SAT CNF **NP**-difficile

Toute formule booléenne est équivalente à une formule sous forme CNF.

En utilisant le fait que

$$A \Rightarrow B \equiv A \vee \neg B,$$

on peut facilement mettre la formule du théorème précédent sous forme CNF.

Théorème

SAT restrint aux formules CNF est **NP**-complet.

On peut aussi montrer (avec un peu de travail) :

Théorème

3-SAT (les clauses ont au plus trois literraux) est **NP**-complet.

Plan

1. Prouver qu'un problème est NP-complet

2. Approches pour les problèmes NP-complets, SAT solvers

Comment faire ?

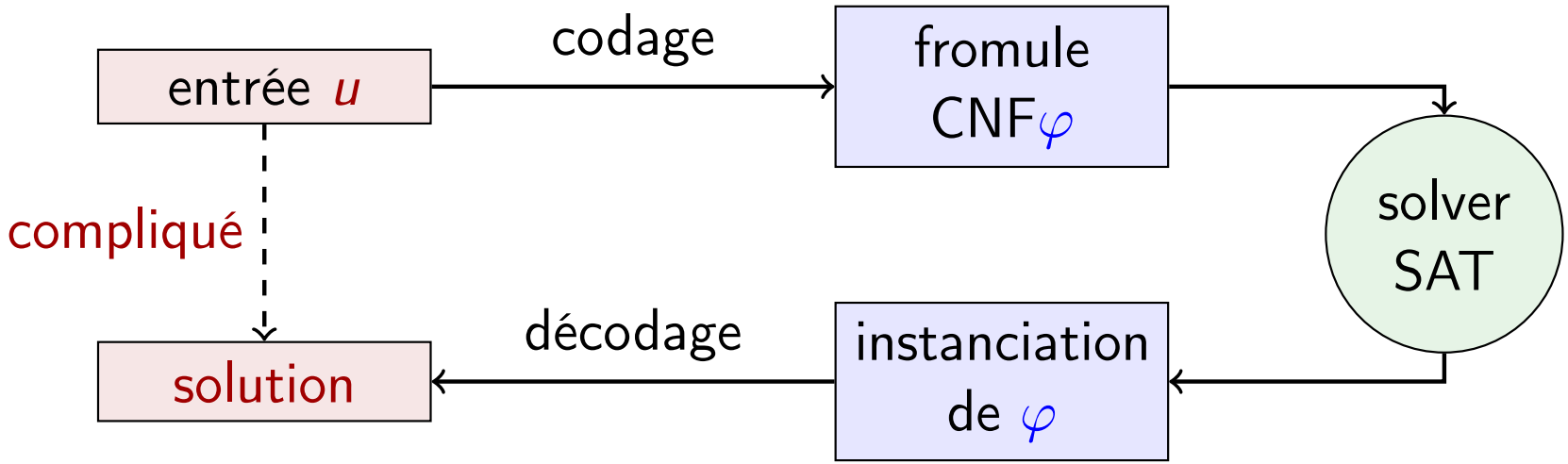
Résoudre un problème NP-complet par force brute (ou proche) demandera rapidement trop de ressources.

- Pour les problèmes de décision, utilisation de **SAT-solvers**.
- Pour les problèmes d'optimisation, utilisation de **solvers en nombres entiers**, d'approximation ou d'heursitiques, de méta-heuristiques.

Codage SAT

Codage SAT

Resoudre un problème par codage SAT, c'est trouver une fonction calculable en temps polynomiale qui, à chaque entrée u du problème d'entrée, associe une formule SAT en CNF φ qui est satisfiable si et seulement si u est une instance positive du problème.
De plus, on chercher à ce qu'une instanciation de la formule (qui rende son interprétation vraie) donne une solution au problème.



Suduku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

En testant toutes les possibilités par colonne (5 cases vides), on obtient $(5!)^9 \simeq 10^{18}$ possibilités. Plusieurs dizaines d'années de calcul.

Suduku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

En testant toutes les possibilités par colonne (5 cases vides), on obtient $(5!)^9 \simeq 10^{18}$ possibilités. Plusieurs dizaines d'années de calcul.

On va coder avec une formule SAT :

$X_{i,j,k}$ est vraie si la case (i, j) contient k.

Il y a n^3 variables (dans le cas classique $n = 9$).

Codage SAT du Suduku.

La case (i, j) a au moins une valeur :

$\bigwedge_{k=1}^{9} X_{i,j,k}$

La case (i, j) a au plus une valeur :

$\bigwedge_{k=1}^{9} \bigwedge_{k'=1,k'\neq k}^{9} \left(X_{i,j,k} \Rightarrow X_{i,j,k'}\right)$ ce qui s'écrit aussi
$\bigwedge_{k=1}^{9} \bigwedge_{k'=1,k'\neq k}^{9} \left(\neg X_{i,j,k} \vee X_{i,j,k'}\right)$

On le fait pour toutes les cases :

$\bigwedge_{i=1}^{9} \bigwedge_{j=1}^{9} \left(\bigwedge_{k=1}^{9} X_{i,j,k} \wedge \bigwedge_{k=1}^{9} \bigwedge_{k'=1,k'\neq k}^{9} \left(\neg X_{i,j,k} \vee X_{i,j,k'}\right)\right)$

Codage SAT du Suduku, Contrainte sur les colonnes

Le nombre 1 apparaît dans la première colonne

$\bigvee_{i=1}^{9} X_{i,1,1}$

Codage SAT du Suduku, Contrainte sur les colonnes

Le nombre 1 apparaît dans la première colonne

$$\bigvee_{i=1}^{9} X_{i,1,1}$$

Chaque nombre apparaît dans la chaque colonne

$$\bigwedge_{j=1}^{9} \bigwedge_{k=1}^{9} \bigvee_{i=1}^{9} X_{i,j,k}$$

Par un argument de cardinalité, cela suffit à garantir la contrainte sur les colonnes.

Codage SAT du Suduku, Contrainte sur les colonnes

Le nombre 1 apparaît dans la première colonne

$$\bigvee_{i=1}^{9} X_{i,1,1}$$

Chaque nombre apparaît dans la chaque colonne

$$\bigwedge_{j=1}^{9} \bigwedge_{k=1}^{9} \bigvee_{i=1}^{9} X_{i,j,k}$$

Par un argument de cardinalité, cela suffit à garantir la contrainte sur les colonnes.

On fait de même pour les lignes et les carrés.

Codage SAT du Suduku, conditions initiales

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

$$X_{1,1,5} \wedge X_{1,2,3} \wedge X_{1,5,7} \dots$$

La formule obtenue a une solution si et seulement si la grille de Suduku en a une.
Une instanciation des $X_{i,j,k}$ qui rend la formule vraie donne une solution de la grille de suduku.

Coloriage

Pour un entier k fixé, on cosidère le problème suivant :

k-COLOR

Donnée : un graphe $G = (V, E)$
Question : est-il possible de colorier les sommets de G avec les couleurs de telle sorte que deux sommets voisins n'aient jamais la même couleur.

On admettra que 3-COLOR est **NP**-complet.

Exercice

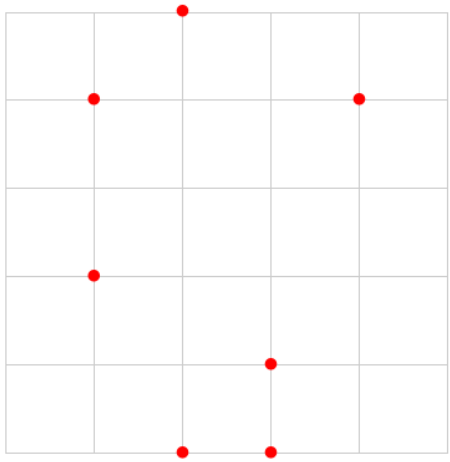
- Justifier que 2-COLOR est dans **P**.
- Proposer un codage SAT pour 3-COLOR.

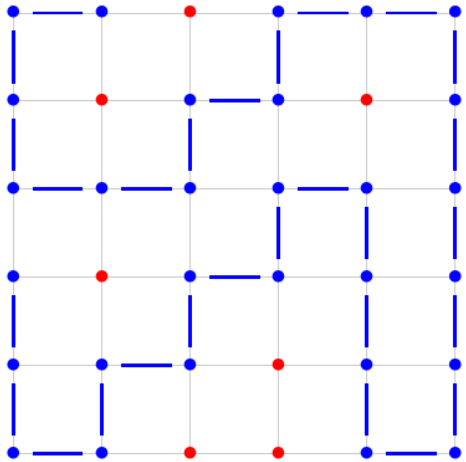
Cas pratique

Chemin autoevitant

Donnée : Un entier n et une liste ℓ de points du quadriage $n \times n$ et un entier $k \leqslant n^2$

Question : est-il possible de faire un chemin sur le quadrillage qui évite les points de ℓ, de longueur k, sans passer deux fois par le même point.





Cas pratique

Chemin autoevitant

Donnée : Un entier n et une liste ℓ de points du quadriage $n \times n$ et un entier $k \leqslant n^2$

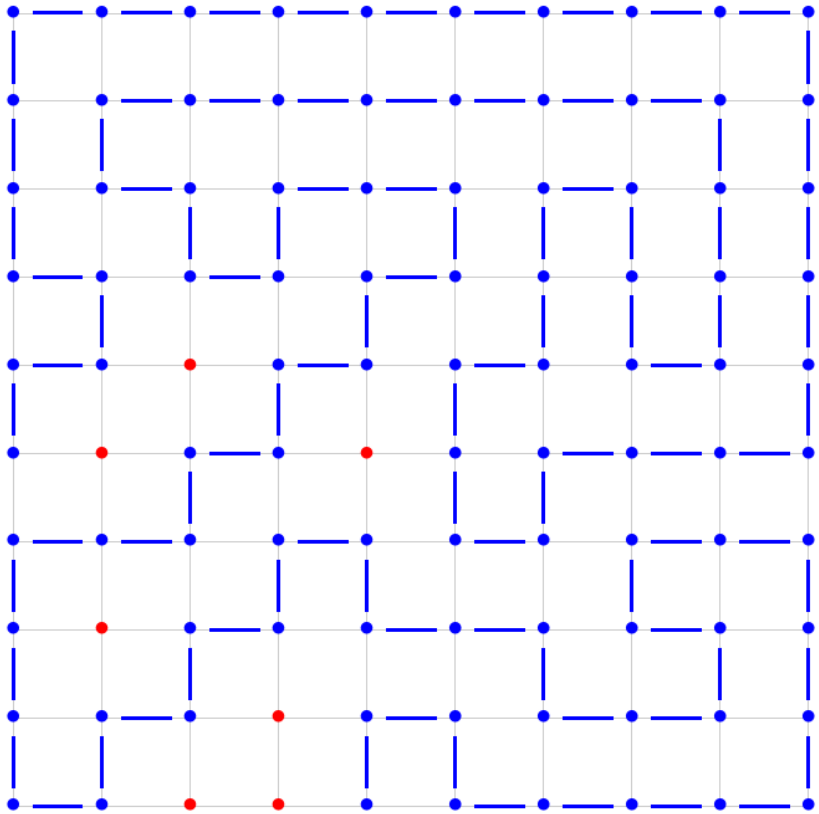
Question : est-il possible de faire un chemin sur le quadrillage qui évite les points de ℓ, de longueur k, sans passer deux fois par le même point.

- Environ 120 lignes de Python, $n^3 + 4n^2$ variables (pour $n = 8$, 768 variables).
- Cryptominisat [1]
- Trouve une solution pour $n = 8$ en 20 secondes.
- Montre qu'il n'y a pas de solution pour $n = 7$ en 2 minutes.

1. `https://www.msoos.org/cryptominisat5/`

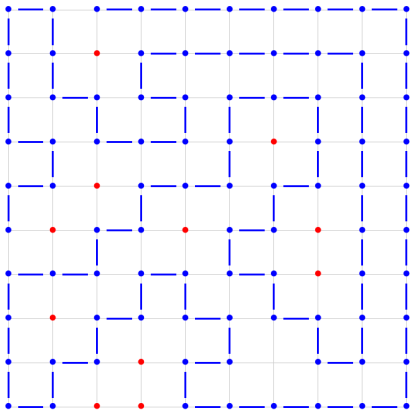
Cas pratique

- Solution trouvée pour $n = 9$ avec un sommet libre en 23s.
- Solution trouvée pour $n = 10$ avec un sommet libre en 2 minutes.
- Programme arrêté après 10h pour $n = 9$.
- Solution trouvée pour $n = 10$ en 20 minutes.

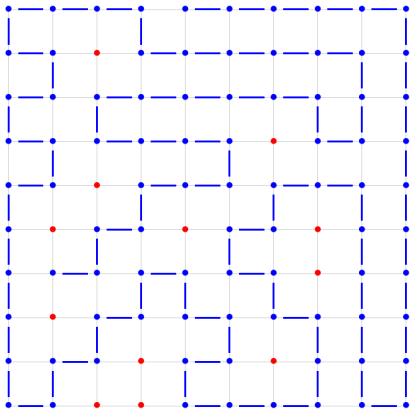


Cas pratique

- Solution trouvée pour $n = 10$ avec un sommet libre en 4 minutes.



- Solution trouvée pour $n = 10$ avec un sommet libre en 4 minutes.



Fichiers `.cnf`

Les solver SATs utilisent des fichiers `.cnf`.
$X_1 \wedge (\neg X_2 \vee X_3 \vee \neg X_4) \wedge (X_3 \vee X_4)$

```
p cnf 4 3
1 0
-2 3 -4 0
3 4 0
```

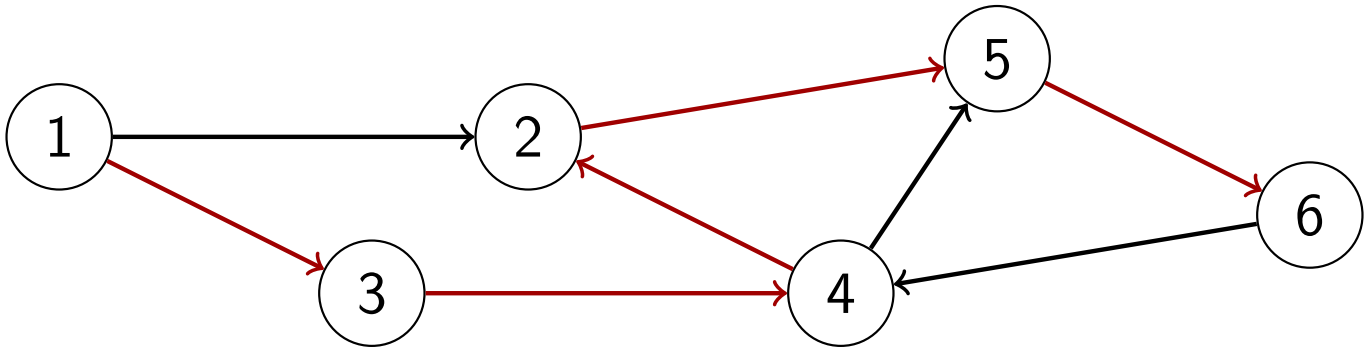
Certains solver autorisent des clause `xor`, on commence alors la ligne par un `x`.

Chemin Hamitlonien

Chemin Hamiltonien

Donnée : un graphe orienté $G = (V, E)$

Question : existe-t-il un chemin qui passe une et une seule fois par chaque sommet ?



Exercice

Proposer un codage SAT pour le problème de chemain Hamiltonien

Emploi du temps

Exercice

Proposer un codage SAT pour le problème de l'emploi du temps