

• Algo $2^{O(f(n))}$ $\text{poly}(n)$ $\Rightarrow \exists c, d, k \text{ t.q. algo} \leq 2^{d f(n)} n^k$

• $g(n) \in o(f(n))$ si $g(n) \in O(f(n))$ mais $f(n) \notin O(g(n))$
 $\Rightarrow g(n)$ est "meilleure" que $f(n)$

$$n^2 \in o(n^3)$$

$\Rightarrow g(n)$ est $o(f(n))$ si $\forall \varepsilon > 0$, $g(n) \leq \varepsilon f(n)$ pour n assez grand

ETH: il n'y a pas d'algo qui décide 3-SAT en temps $2^{o(n)}$ $\text{poly}(n)$, où $n = \#$ de variables.
 \Leftrightarrow D'autrement, $\exists c$ t.q. tout algo pour 3-SAT est en temps $\geq 2^{c n}$ $\text{poly}(n)$.

Un algo $10000^{\overbrace{n}^{1000}} n^{10000}$ viendrait montrer que ETH est fausse
 $\leq 2^{\overbrace{100}^{100} n} \text{poly}(n) \rightarrow 2^{o(n)} \text{poly}(n)$

Origine: il existe un algo $\leq 1.84^n \text{poly}(n)$ pour 3-SAT.

algo 3sat(φ) $C_1 \wedge C_2 \wedge \dots \wedge C_m$ $C_i = (x_1 \vee \bar{x}_2 \vee x_3)$
 \triangleright si φ a 0 clauses, return true; si φ a 1 vars, return false;
prendre une clause $C_i = (x_1 \vee x_2 \vee x_3)$

brancher récursivement sur 3 façons de sat. C :

- $n-1$ vars \rightarrow L fixer $x_1 = \text{True}$, appeler 3sat($\varphi \setminus \{C_i, x_1\}$)
- $n-2$ vars \rightarrow L fixer $x_1 = \text{False}$, $x_2 = \text{False}$, appeler 3sat($\varphi \setminus \{C_i, x_1, x_2\}$)
- $n-3$ vars \rightarrow L fixer $x_1 = \text{False}$, $x_2 = \text{True}$, $x_3 = \text{True}$, appeler 3sat($\varphi \setminus \{C_i, x_1, x_2, x_3\}$)

$t(n)$ = temps de cet algo

$$t(n) = \text{poly}(n) + t(n-1) + t(n-2) + t(n-3)$$

//, magie analyse récurrence

$$|n| = \text{poly}(n) + |n-1| + |n-2| + |n-3|$$

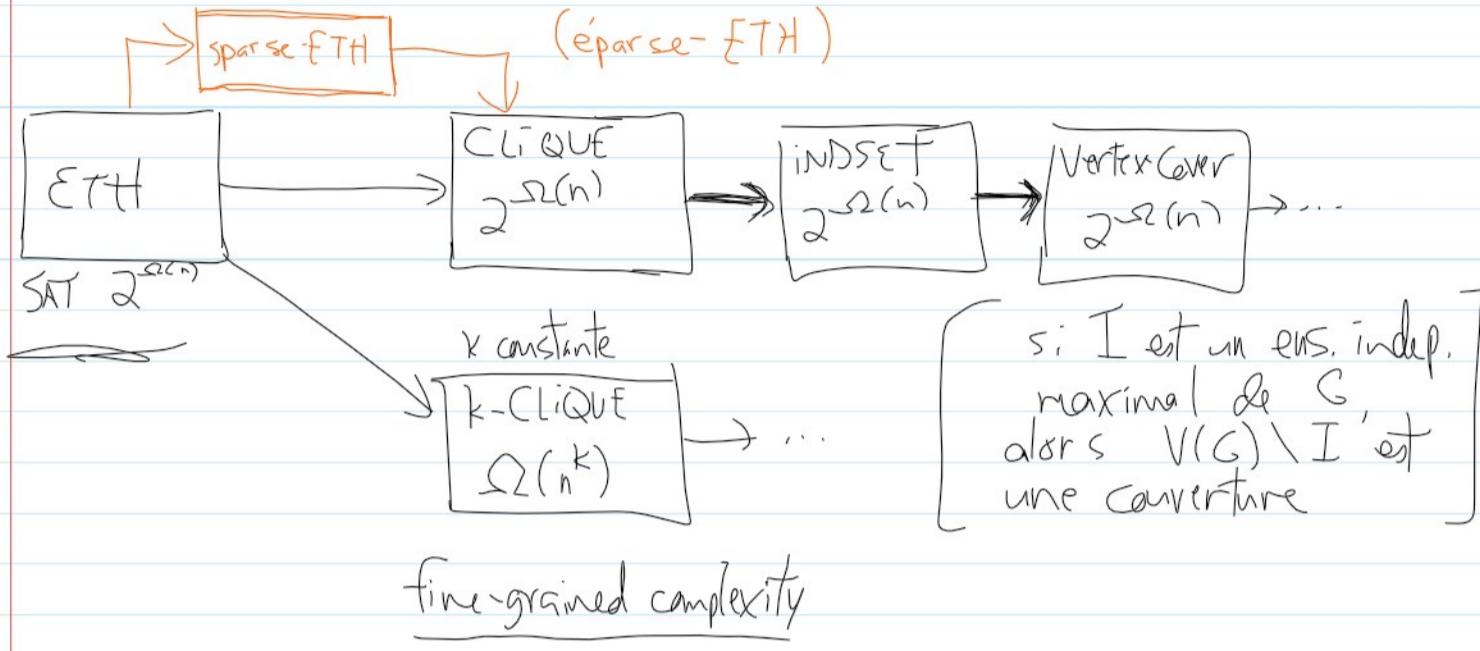
↓ magie analyse récurrence

$$t(n) \in \underline{\mathcal{O}(1.84^n \text{poly}(n))}$$

Meilleur connu pour 3-SAT: $1.31^n \text{poly}(n)$

$$1.84^n = 2^{cn}$$

$$1.31^n = 2^{cn}$$



Idée des réductions de 3-SAT \rightarrow problème X

si on avait un algo $2^{o(f(n))} \text{poly}(n)$ pour X, alors

on aurait un algo $2^{o(f(n))} \text{poly}(n)$ pour 3-SAT.

Sparse-ETH: Un algo qui décide 3-SAT en $2^{o(n)} \text{poly}(n)$

même sur les instances "éparses", sparse, i.e.

sur n vars et $\leq cn$ clauses, constante

$$\text{ETH} \xrightarrow{(2)} \text{Sparse-ETH} \xrightarrow{(1)} \text{CLIQUE et } 2^{\Omega(n)} \text{poly}(n)$$

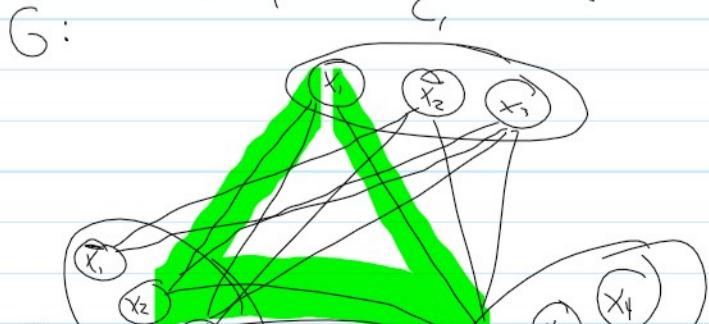
$n = |V(G)|$

① Si Sparse-ETH est vraie, alors CLIQUE est $2^{\Omega(n)} \text{poly}(n)$.

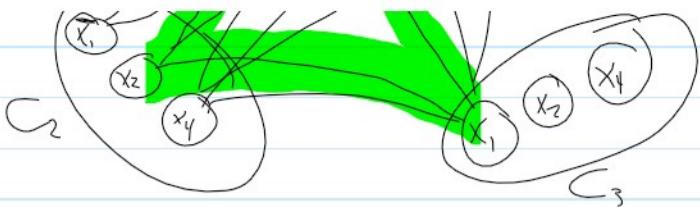
• Soit $\varphi = C_1 \wedge C_2 \wedge \dots \wedge C_{cn}$ une instance de 3-SAT avec cn clauses.

On génère $\langle G, \text{con} \rangle$ une instance de CLIQUE avec la réduction classique.

ex: $\varphi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4) \wedge (x_1 \vee x_2 \vee \bar{x}_4)$



$$\left| \begin{array}{l} \varphi \in 3\text{-SAT} \\ \iff \\ \langle G, \text{con} \rangle \in \text{CLIQUE} \end{array} \right.$$



Cette réduction génère un graphe G avec

$3 \cdot c_n$ sommets.

Si on avait un algo $2^{cn} \text{poly}(n)$ pour CLIQUE,

on pourrait décider 3-SAT comme suit:

- construire $f(\varphi) = \langle G, c_n \rangle$ ($\text{poly}(n)$)

- décider si $\langle G, c_n \rangle \in \text{CLIQUE}$ en $2^{cn} \text{poly}(n)$ ↗

\Rightarrow on décide 3-SAT en temps $\text{poly}(n) + 2^{cn} \text{poly}(n)$,

ce qui contredit la sparse-ETH \Rightarrow un tel algo ne peut pas exister si sparse-ETH est vraie.

$\text{ETH} \Rightarrow \text{sparse-ETH}$

$n \text{ vars}$

$n \text{ vars}$

$m \text{ clauses}$

$\leq c_n \cdot \text{clauses}$



Lemme de la sparsification [I, P, Z 2001]

Soit φ une instance 3-SAT. Alors $\forall \varepsilon > 0, \exists 2^{\varepsilon n}$

formules $\varphi_1, \varphi_2, \dots, \varphi_{2^{\varepsilon n}}$ en 3-CNF telles que

- $\exists c$ t.g. chaque φ_i a $\leq c \cdot n$

- $\varphi \in 3\text{-SAT} \Leftrightarrow \varphi_1 \vee \varphi_2 \vee \dots \vee \varphi_{2^{\varepsilon n}} \in \text{SAT}$ ↗

- $\varphi_1, \dots, \varphi_{2^{\varepsilon n}}$ peuvent être construites en $2^{\varepsilon n} \text{poly}(n)$

$\text{ETH} \Rightarrow \text{sparse-ETH}$

Si tout algo pour 3-SAT est $2^{cn} \text{poly}(n)$, alors $\exists c$ t.g.

tout algo pour 3-SAT $\geq 2^{cn} \text{poly}(n)$. Supposons que \exists algo M $2^{0(n)} \text{poly}(n)$ pour sparse-3-SAT. Donc $\forall \alpha > 0, M$ est

$\leq 2^{\alpha n} \text{poly}(n)$. On choisit $\alpha = \frac{\varepsilon}{8}$.

On utilise le lemme avec $\varepsilon = \frac{\varepsilon}{8}$.

On utilise le lemme avec $\varepsilon = \frac{1}{8}$.

$$\varphi \rightarrow \varphi_1 \vee \varphi_2 \vee \dots \vee \varphi_{2^{\lceil \varepsilon n \rceil}}$$

Soit l'algorithme \mathcal{Q} qui sur entrée φ :

- construit $\varphi_1, \varphi_2, \dots, \varphi_{2^{\lceil \varepsilon n \rceil}}$ selon le lemme

$$\left\} 2^{\lceil \varepsilon n \rceil}$$

- pour chaque φ_i (éparse)

décider si $\varphi_i \in \text{SAT}$ en temps $2^{\alpha n} \text{poly}(n)$

si oui, Accepter

Rejectte

$$\left\} 2^{\varepsilon n} \cdot 2^{\alpha n} = 2^{\frac{1}{8}n} \cdot 2^{\frac{1}{8}n} = 2^{\frac{1}{4}n}$$

$$2^{\frac{1}{8}n} + 2^{\frac{1}{4}n} \leq 2^{\frac{1}{2}n} < 2^n$$

On décide donc 3-SAT en temps $\leq 2^{\alpha n} \text{poly}(n)$, une contradiction de la ETH.

Sparse-ETH implique que 3-SAT n'est pas décidable en temps $2^{o(n+m)} \text{poly}(n)$. $n = \# \text{vars}$ $m = \# \text{clauses}$

Parce que les instances avec $m \leq cn$ sont $2^{o(n)} \text{poly}(n)$

$\Rightarrow 2^{o(n+m)} \text{poly}(n)$ pas possible (selon ETH)

$\Rightarrow 2^{o(n)} \text{poly}(n)$ pas possible

Strong-ETH (SETH) $\downarrow \downarrow$ $\text{poly}(n), \Omega(n), \dots$

- SAT pas décidable en $\leq 2^n \text{poly}(n)$

- Implications algos polynomiaux

e.g. diamètre d'un graphe $\Omega(n^2)$

alignement de séquence d'ADN $\Omega(n^2)$