Temps exponentiel

- ETH$_{VI}$: Exponential Time Hypothesis

  Il n'y a pas d'algo en temps $2^{o(n)} \cdot poly(n)$ pour décider 3-SAT.

  n = nombre de variables.

  - $poly(n)$ = une fonction $c \cdot n^k$, $c, k \in \mathbb{N}$
  - $o(f(n))$: une fonction $g(n) \in o(f(n))$ si $f(n) \notin O(g(n))$ mais $g(n) \in O(f(n))$.

    ex: $\sqrt{n} \in o(n)$ $\quad \log n \in o(n)$

    $n^{9/10} \in o(n)$

    si $\lim_{n \to \infty} \frac{g(n)}{f(n)} = 0$, alors $g(n) \in o(f(n))$

- Un algo $2^{1000 \cdot \sqrt{n}} n^{10000}$ réfuterait la ETH.

- 2^n n'est pas une borne inférieure stricte pour 3SAT

  $2^{c \cdot n}$ est permis $\quad c \in \mathbb{R}, c > 0$

ex: On peut décider 3-SAT en temps $O(1.84^n poly(n))$.

Ceci ne réfute pas ETH car $1.84^n = 2^{c \cdot n}$

3sat(φ)

Soit $\varphi = C_1 \wedge C_2 \wedge C_3 \wedge \dots \wedge C_m$ une instance de 3-SAT.

ex: $C_i = (x_1 \vee \bar{x}_2 \vee x_3)$ $\qquad C_i = (F \vee x_4 \vee F)$ $\qquad C_i : x_4 \vee \bar{x}_5$

Prendre une clause C_i, avec disons variables $x_1 \vee \bar{x}_2 \vee x_3$

Brancher récursivement sur les façons de satisfaire C_i

- fixer $x_1 = T$, $res_1 = 3sat(\varphi \setminus \{C_i, x_1\})$ $\quad t(n-1)$
- fixer $x_1 = F, x_2 = F$, $res_2 = 3sat(\varphi \setminus \{C_i, x_1, x_2\})$ $\quad t(n-2)$
- fixer $x_1 = F, x_2 = T, x_3 = T$, $res_3 = 3sat(\varphi \setminus \{C_i, x_1, x_2, x_3\})$ $\quad t(n-3)$

return $res_1 \vee res_2 \vee res_3$ // un des appels a réussi à satisfaire

return $res_1 \vee res_2 \vee res_3$ // un des appels a réussi à satisfaire

Complexité: par rapport à n = #variables

soit $t(n)$ le temps de l'algo sur n variables.

On a $t(n) = poly(n) + t(n-1) + t(n-2) + t(n-3)$

traiter φ

$\Rightarrow$ Ce temps est $\leq 1.84^n \cdot poly(n)$ quand même $2^{c \cdot n} poly(n)$

Meilleur alg pour 3-SAT : $\approx 1.34^n poly(n)$

- Comment utiliser ETH pour bornes inférieures algorithmique

ETH 3-SAT $\leq_?$ CLIQUE

$\varphi \xrightarrow{f} \langle G, k \rangle$

CLIQUE n'admet pas d'algo $2^{o(n)} poly(n)$

Idée: Si on avait un un tel algo, il permettrait de résoudre 3-SAT en temps $2^{o(n)} poly(n)$.

$\Rightarrow$ Si ETH vraie, ne peut pas exister.

(v2)

ETH: on ne peut pas résoudre 3-SAT en temps $2^{o(n+m)} poly(n)$

où n = #vars, m = #clauses

équivalent: 3-SAT restreint aux formules φ avec n vars et $\leq c \cdot n$ clauses, [illegible] $2^{o(n)} poly(n)$

équivalent: 3-SAT restreint aux formules φ avec n vars et $\leq c \cdot n$ clauses, ne peut pas être résolu en $2^{o(n)} \text{poly}(n)$.

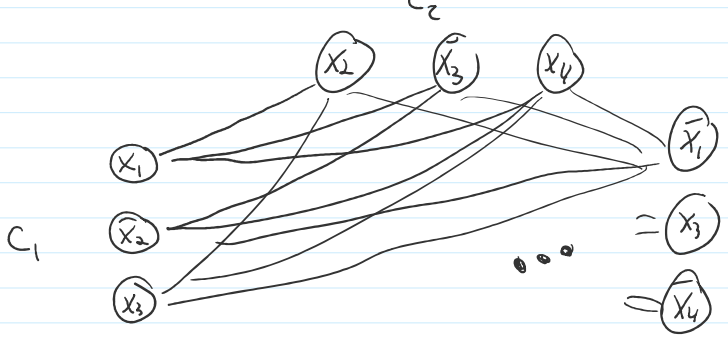
Thm: CLIQUE ne peut pas être résolu en temps $2^{o(|V(G)|)} \cdot \text{poly}(|V(G)|)$ si ETH est vraie.

Idée: On a déjà vu une réduction de 3-SAT vers CLIQUE.

$\varphi \xrightarrow{f} \langle G, k \rangle$ tels que $\varphi \in$ 3-SAT $\Leftrightarrow \langle G, k \rangle \in$ CLIQUE

ex: $(x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee x_4) \wedge (\bar{x}_1 \vee x_3 \vee \bar{x}_4)$

C_1 C_2 C_3



Ce graphe contient $|V(G)| = 3m$ sommets

m = # clauses

(a)—(b) si a, b clauses différentes et a ≠ négation de b

Si on pouvait résoudre CLIQUE en temps $2^{o(|V(G)|)} \text{poly}(n)$, on pourrait résoudre 3-SAT comme suit:

sur φ:

- calculer $\langle G, k \rangle = f(\varphi)$ // poly(n)
- résoudre $\langle G, k \rangle$ en temps $2^{o(|V(G)|)} \text{poly}(|V(G)|) = 2^{o(3m)} \text{poly}(3m) = 2^{o(n+m)} \text{poly}(n+m)$
- accepter $\Leftrightarrow \langle G, k \rangle$ a été accepté

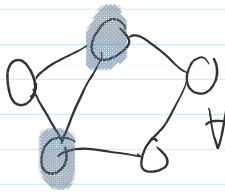
Donc on pourrait décider 3-SAT en $2^{o(n+m)} \text{poly}(n+m)$ ce qui réfuterait la ETH.

- Supposition implicite $o(3m) \rightarrow o(n+m)$

- Supposition implicite $o(3m) \rightarrow o(n+m)$
  n variables et $m \leq c \cdot n$ clauses

| $o(3m) = o(3m + n$ "importe quoi"$)$
| $o(3m) = o(3m+n) = o(n+m)$

- IND-SET n'a pas d'algo $2^{o(n)}\text{poly}(n)$
- SET-COVER n'a pas d'algo $2^{o(n)}\text{poly}(n)$, n = # ensembles
- ex: ENSEMBLE-DOMINANT NP-complet sur graphe planaire



↳ $X \subseteq V(G)$ tel que
$\forall v \in V(G) \setminus X$, v a un voisin dans X

$\exists$ algo $2^{\sqrt{n}}\text{poly}(n)$

SETH: Strong Exponential Time Hypothesis

- ETH: $2^{o(n)}\text{poly}(n)$ pour 3-SAT
- SETH: CNF-SAT ne peut pas être résolu en temps $2^{\delta n}\text{poly}(n)$, $\forall \delta < 1$.

CNF-SAT $= \{\varphi : \varphi \in$ SAT et φ est en forme conjonctive normale$\}$

Autre formulation: CNF-SAT n'a pas d'algo $(2-\varepsilon)^n \text{poly}(n)$
$\forall \varepsilon$, avec $0 < \varepsilon < 2$

Principe général: Montrer qu'un problème X n'est pas résoluble en temps $O(n^{k-\varepsilon})$, $\forall \varepsilon > 0$

Démarrer de SETH: $\exists \delta < 1$ tel que $2^{\delta n} \text{poly}(n)$ pour CNF-SAT.

Sur instance φ de CNF-SAT

transformer φ en $f(\varphi)$ instance de X

telle que:
- $\varphi \in$ CNF-SAT $\iff f(\varphi) \in X$
- f s'exécute en temps $2^{\delta n} \text{poly}(n)$, $\delta < 1$
- $|f(\varphi)| \in O(2^{n/k})$

- Argument: si on pouvait résoudre X en temps $O(n^{k-\varepsilon})$, n = taille de l'instance, on pourrait décider CNF-SAT comme suit:

sur φ:

construire $f(\varphi)$ // $2^{\delta n}\text{poly}(n)$ $\delta < 1$

décider $f(\varphi)$ en temps $O(|f(\varphi)|^{k-\varepsilon})$

$$= O((2^{n/k})^{k-\varepsilon}) = O(2^{\frac{n\cdot(k-\varepsilon)}{k}})$$

$$= O(2^{\frac{nk-\varepsilon n}{k}})$$

$$= O(2^{\frac{nk}{k}-\frac{\varepsilon n}{k}})$$

$$= O(2^{n-\varepsilon n/k})$$

$$= O(2^{n(1-\varepsilon/k)})$$

$= O(2^{n \cdot \delta})$ où $\delta < 1$

ce qui réfute SETH.