

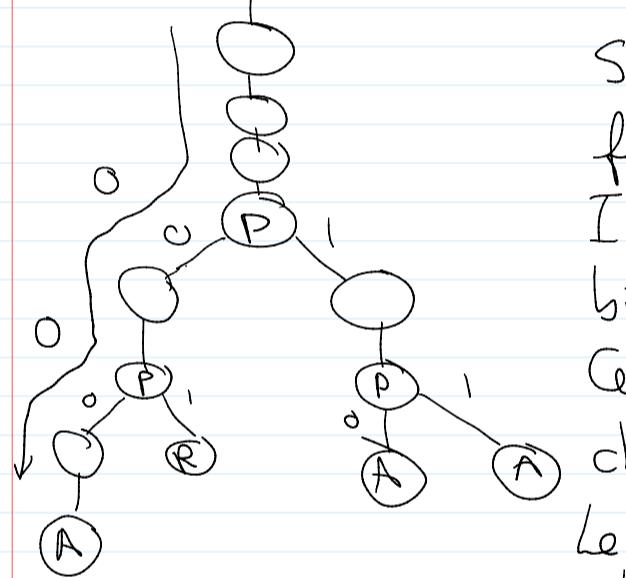
Si $G \in \text{HAMCYCLE} \Rightarrow \text{OPT}(G') = n$

Si $G \notin \text{HAMCYCLE} \Rightarrow \text{OPT}(G') > n^{1000}$

Si on a une c-approx M pour MinTSP, on peut décider $\neg G \in \text{HAMCYCLE}$ en temps poly

- transformer G en G' , G'
- rouler M sur G' , soit r le résultat
- si $r \leq c \cdot n$
- Accepter
- si non
- Rejeter

MT probabiliste M



Supposons que M fait toujours $f(n)$ lancer de pièces.

Il y a $2^{f(n)}$ combinaisons de bits aléatoires utilisées par M.

Ces bits aléatoires sont des chaînes dans $\{0,1\}^{f(n)}$.

Le chemin de calcul de M est entièrement déterminé par w et $r \in \{0,1\}^{f(n)}$ de l'oracle.

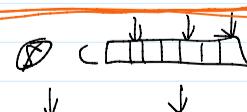
Vérificateur prob. V pour un langage L

- en temps poly par rapport à w . ($O(|w|^k)$)
- $w \in L \Rightarrow \exists c \text{ t.q. } \Pr[V \text{ accepte } \langle w, c \rangle] = 1$ arbitraire
- $w \notin L \Rightarrow \forall c \quad \Pr[V \text{ accepte } \langle w, c \rangle] \leq \frac{1}{2}$

Vérificateur PCP

- Idée: Limiter le # de lancers de pièce et limiter le # de bits lus du certificat

Def : vérificateur PCP



Un langage L a un vérificateur $(r(n), g(n))$ -PCP

si il existe une MT prob. V qui satisfait:

① sur entrée $\langle w, c \rangle$, V prend un temps $O(|w|^k)$

② V utilise au plus $r(n)$ lances de pièce et $g(n)$ bits

n'importe quelle pos.

- (1) sur entrée $\langle w, c \rangle$, V prend un Temps $\mathcal{O}(|w|^r)$
- (1) V utilise au plus $r(n)$ lancers de pièces et accède à au plus $q(n)$ bits de c .
- (2) si $w \in L$, $\Pr[V \text{ accepte } \langle w, c \rangle] = 1$ pour un c
- (3) si $w \notin L$, $\Pr[V \text{ accepte } \langle w, c \rangle] \leq \frac{1}{2}$ $\forall c$

Déf: On dit que $L \in \text{PCP}(r(n), q(n))$ si il existe des constantes α et β telles que L admet un vérificateur $(c \cdot r(n), d \cdot q(n))$ -PCP

LE théorème PCP: $\text{NP} = \text{PCP}(\log n, 1)$

$\mathcal{O}(\log n)$ pièces - $\mathcal{O}(1)$ bits lus

- Preuve > 100 pages ~90 (Arora & al.)
- Preuve ≤ 25 pages ~2000 (Dinur)

Notation: $\text{PCP}(r(n), \text{poly}(n)) = \bigcup_k \text{PCP}(r(n), n^k)$

$\text{PCP}(\text{poly}(n), q(n)) = \bigcup_k \text{PCP}(n^k, q(n))$

Ex: $\text{PCP}(0, \text{poly}(n)) = \text{NP}$

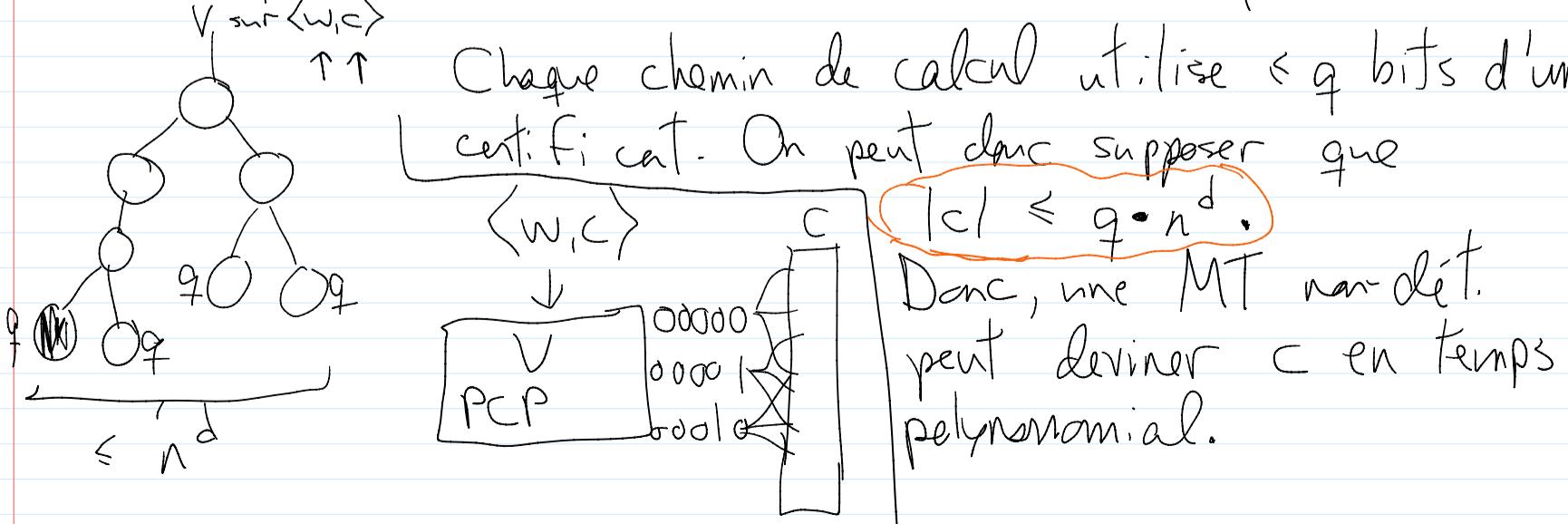
Idee: si on a un vérif. PCP qui utilise 0 bits aléatoires, roule en temps $\mathcal{O}(n^k)$, et consulte $\text{poly}(n)$ de bits de certificat, alors ce vérificateur PCP est aussi un vérificateur NP "ordinaire".

Ex: $\text{PCP}(\log n, 1) \subseteq \text{NP}$

Soit $L \in \text{PCP}(\log n, 1)$, et soit V son vérificateur $\text{PCP}(\log n, q)$.

V peut prendre un chemin différent pour chaque $r \in \{0, 1\}^{d \log n}$.

Il y a $2^{d \log n}$ chemins possibles, et $2^{d \log n} = (2^{\log n})^d = n^d (\text{poly})$.



On peut donc concevoir une MT non-déterm. pour L comme suit:

On peut donc concevoir une MT non-déterm. pour L comme suit:

- sur entrée $w \in \{0,1\}^*$

 └ deviner le certif. c que V utilisera t

 └ pour chaque $r \in \{0,1\}^{d \cdot \log n}$

 └ simuler V sur entrée $\langle w, c \rangle$ lorsqu'elle fait les choix r.

 └ si V a accepté sur tous les chemins, accepter

 └ sinon, rejeter

MT non-déterm. dont le langage est L . Donc $L \in \text{NP}$. \blacksquare

Ex: $\text{PCP}(0, \log n) = P$

① $P \subseteq \text{PCP}(0, \log n)$

Soit $L \in P$, une MT M qui décide L. Donc, M utilise 0 bits aléatoire et décide si $w \in L$ correctement avec Prob 1 peu importe les bits d'un certificat.
Donc, M est un vérificateur $\text{PCP}(0, \log n)$

② $\text{PCP}(0, \log n) \subseteq P$.

Soit $L \in \text{PCP}(0, \log n)$, et V un vérificateur $(0, d \cdot \log n)$ -PCP pour L.

Soit M qui, sur entrée w

- pour chaque $c \in \{0,1\}^{d \cdot \log n}$

 └ simule V sur $\langle w, c \rangle$

 s: V accepte, accepter

- rejeter

 } quantité polynomiale

$$2^{d \cdot \log n} = n^d$$

Si $w \in L$, $\exists c$ t.q. $\Pr[V \text{ accepte } \langle w, c \rangle] = 1 \Rightarrow M \text{ accepte } w$

Si $w \notin L$, $\forall c \quad \Pr[V \text{ accepte } \langle w, c \rangle] \leq \frac{1}{2}$

Puisque V utilise 0 bit aléatoires, le # de chemins = 1,

$\Rightarrow \Pr[V \text{ accepte } \langle w, c \rangle] \in \{0, 1\}$

$\Rightarrow \Pr[V \text{ accepte } \langle w, c \rangle] = 0 \quad (\text{car } \leq \frac{1}{2})$

$\Rightarrow M$ rejette w.

Liens l'approximation

Thm 1: $\text{NP} = \text{PCP}(\log n, 1)$

Problème: MAX-q-CSP (constraint satisfaction problem)

Entrée: variables x_1, \dots, x_n
 formules bool $\varphi_1, \varphi_2, \dots, \varphi_m$ (forme
 chacune utilisant $\leq q$ variables arbitraire)

Sortie: assignation qui maximise le # de φ_i satisfaites

Thm 2: $\forall L \in NP$, il existe f de Σ^* en une instance $\{\varphi_1, \dots, \varphi_m\}$ de MAX- q -CSP telle que
 $w \in L \Rightarrow$ on peut satisfaire tous les φ_i [OPT = m]
 $w \notin L \Rightarrow$ on peut satisfaire $\leq \frac{1}{2} m$ des φ_i [OPT $\leq \frac{m}{2}$]
 Donc, pas de $\frac{1}{2}$ -approx pour MAX- q -CSP. (si $P \neq NP$)

Thm 3: Thm 1 \Rightarrow Thm 2

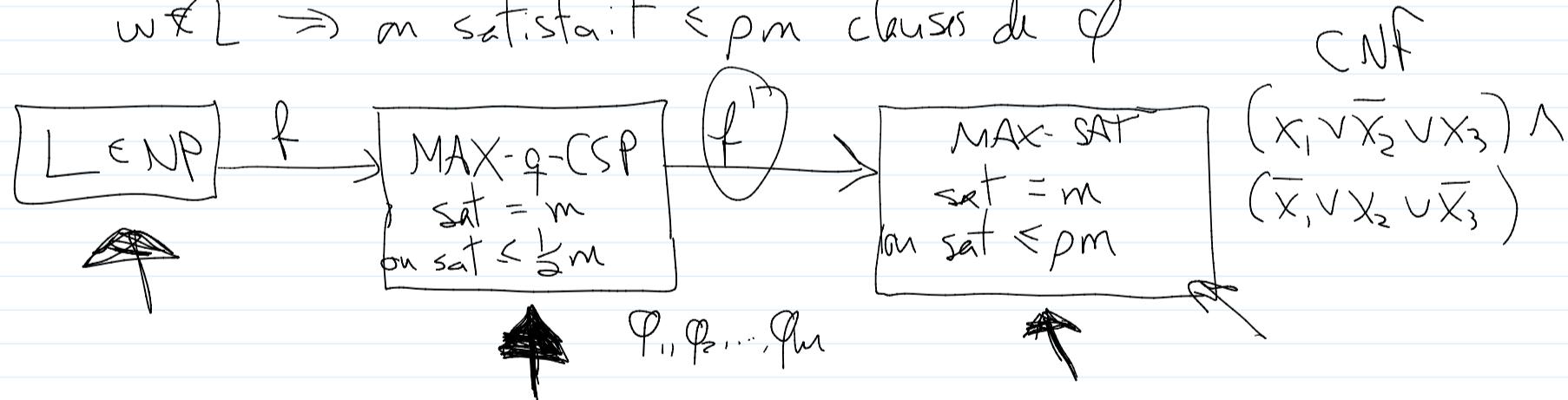
i.e. $NP = PCP(\log n, 1) \Rightarrow$ MAX- q -CSP n'a pas de $\frac{1}{2}$ -approx

Thm 2 \Rightarrow MAX-SAT n'a pas de p-approx pour une certaine constante p. (si $P \neq NP$)

On va montrer que $\forall L \in NP, \exists f$ qui fait $\Sigma^* \rightarrow \varphi$ de MAX-SAT t.f.

$w \in L \Rightarrow$ on satisfait m clauses de φ

$w \notin L \Rightarrow$ on satisfait $\leq pm$ clauses de φ



Soit $\{\varphi_1, \dots, \varphi_m\}$ une instance de MAX- q -CSP t.q.

- soit m des φ_i satisfiables
- ou $\leq \frac{1}{2} m$ des φ_i "

On va transformer chaque φ_i en forme CNF $\rightarrow \varphi'_i$ équivalente.

↳ On prend la table de vérité de φ_i

Pour chaque variables qui donnent Faux,

ajouter une clause qui fait la négation de ces variables.

$$\varphi'_i : x_1 \wedge x_2 \wedge x_3 \wedge \dots \wedge x_q$$

$$\varphi_1 = x_1 \wedge x_2 \wedge x_3 \dots \wedge x_q$$

$\left\{ \begin{array}{c} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{array} \right\} \quad \left\{ \begin{array}{c} T \\ F \end{array} \right\} \quad \left\{ 2^q \text{ rangées} \right\}$

$$\varphi'_1 = \underbrace{(x_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge x_4) \wedge (\bar{x}_1 \wedge \bar{x}_2 \wedge x_3 \wedge \bar{x}_4)}_{\text{clause}} \wedge \dots$$

$$\varphi'_1 = \underbrace{(x_1 \vee x_2 \vee x_3 \vee \bar{x}_4)}_{\text{clause}} \wedge \underbrace{(x_1 \vee x_2 \vee \bar{x}_3 \vee x_4)}_{\text{clause}} \wedge \dots$$

De plus, φ'_i a $\leq 2^q$ clauses (avec q constant).

On peut supposer que φ'_i a exactement 2^q clauses (avec des bidans)

$$\varphi_1 \varphi_2 \varphi_3 \dots \varphi_m$$

\downarrow
 $\varphi'_1 \wedge \varphi'_2 \wedge \varphi'_3 \wedge \dots \wedge \varphi'_m$
 $\underbrace{\text{CNF}}_{\text{CNF}} \quad \underbrace{\text{CNF}}_{\text{CNF}} \quad \dots \quad \underbrace{\text{CNF}}_{\text{CNF}}$

$$C_1 \wedge C_1' \wedge C_2 \wedge C_2' \wedge \dots \wedge C_{2^q} \wedge \dots \quad \left\{ m \cdot 2^q \text{ clauses} \right\}$$

- si on peut satisfaire chaque φ_i , on peut satisfaire chaque 2^q clauses de $\varphi'_1 \wedge \varphi'_2 \wedge \dots \wedge \varphi'_m$.

$$\text{OPT}(\varphi'_1 \wedge \varphi'_2 \wedge \dots \wedge \varphi'_m) = m \cdot 2^q$$

- si on peut satisfaire $\leq \frac{1}{2}$ des φ_i , on peut satisfaire les 2^q clauses pour la $\frac{1}{2}$ des φ'_i , et $2^q - 1$ clauses pour la $\frac{1}{2}$ des φ'_i .

Donc, on peut satisfaire au

$$\frac{1}{2}m \cdot 2^q + \frac{1}{2}m(2^q - 1) = \frac{1}{2}m(2^q + 2^q - 1)$$

$$= \frac{2 \cdot 2^q}{2} \cdot m - \frac{1}{2}m$$

$$= m \cdot 2^q - \frac{1}{2}m = p \cdot m 2^q$$

~~$f < 1$~~