Probabilistically Checkable Proofs

Def A **verifier** is a poly-time deterministic algorithm, that receives an input x and a proof π and accepts/rejects.

The language defined by the verifier \( V \) is \( \{ x \in \Sigma^* \mid \exists \pi \ V(x, \pi) = \text{accept} \} \).

**Completeness** \( x \in L \rightarrow \exists \pi \ V(x, \pi) = \text{acc} \)

**Soundness** \( x \in L \rightarrow \forall \pi \ V(x, \pi) = \text{rej} \)

**NP** - The class of all languages \( L(v) \) for some \( V \).

**Remark** asymmetry between complexity of proof and complexity of verification.

Q What happens if we add randomness to the verifier?

Def An \((r, \xi_x)\) restricted verifier

(If \( \Sigma = \{0, 1\} \) we sometimes omit \( \Sigma \) ) is a poly-time algo.

1. Reads \( x \) and \( r \).
2. Queries \( q \) locations in \( \Pi \).
3. \text{acc/rej}

The lang. defined by the verifier \( V \) is \( L(r, \xi) \).

**Completeness** \( x \in L \rightarrow \exists \pi \ P(V(x, \pi) = \text{acc} ) = 1 \) \( (c) \)

**Soundness** \( x \in L \rightarrow \forall \pi \ P(V(x, \pi) = \text{acc} ) < \frac{1}{2} \) \( (s) \)

\( \text{PCP}^{c, \xi}[r, q]_\xi = \text{class of all lang. with } (r, q, \xi)_\xi \text{-restricted verifier w/ comp. } c \text{ and soundness } \xi. \)

\( \bigcup_c \text{PCP}[0, n^c] = \text{NP} = \bigcup_c \text{PCP}[0(\log n), n^c] \)

\( \text{PCP}[0, 0] = \text{P} = \text{PCP}[0(\log n), 0] \)

[1]
Easy $PCP[O\log n, O(1)] \subseteq NP$

Then $PCP[O\log n, O(1)] = NP$

We'll start by showing $PCP[\text{polylog}, \text{polylog}] \not\subseteq NP$.

**Hardness of Approximation**

An optimization problem maps a set of solutions for each input. Each solution has a value.

**Goal**: Find a solution with optimal value.

- **Max-Clique**: instance = graph $G$, sol's: cliques, value: size
- **Min-Set-Cover**: instance = $U, S_1, \ldots, S_n$, sol's: covers, value: size
- **Max-CSP (constraint satisfaction problem)**

**Def**: Let $V = \{v_1, \ldots, v_n\}$ be variables over alphabet $\Sigma$, $q \in N$.

A $q$-ary constraint is $(\varphi, i_1, \ldots, i_q)$ s.t. $i_1, \ldots, i_q \in [n]$, $\varphi: \Sigma^q \to \{0, 1\}$.

It is satisfied by $a: V \to \Sigma$ iff $\varphi(a(v_{i_1}), \ldots, a(v_{i_q})) = 1$.

Denote $\text{Sat}_{\varphi}(\text{const.})$ - fraction of const. sat. by $a$.

**Max-CSP[\Sigma]$$q$$**: instance = vars $V = \{v_1, \ldots, v_n\}$, $q$-ary constraints $\varphi_1, \ldots, \varphi_m$ over $\Sigma$.

sol's: assign $a: V \to \Sigma$, value: $\text{Sat}_{\varphi}(C)$

Generalizes Max-3SAT, Max-Cut, Max-3COL.

**Def**: An algorithm $A$ approximates problem $\mathcal{O}$ if given input $x$,
it outputs a solution whose value $A(x)$ satisfies

\[
2^{-\epsilon} \leq A(x) \leq \mathcal{OPT}(x) \quad \text{for maximization}
\]

\[
2^\epsilon \leq \mathcal{OPT}(x) \leq A(x) \leq 2 \cdot \mathcal{OPT}(x) \quad \text{for minimization}
\]

**Rem**: $\epsilon$ may be a function of $|x|$. 
Known eff. approx

- Max-Clique \( \frac{n}{\log^2 n} \)
- Min-Set-Cover \( \ln n \)
- Knapsack \( n^3 \cdot V^{\frac{3}{2}} \)
- Max-3SAT \( \frac{1}{8} \)

\[ \text{Def} \quad \text{gap-CSP}(q) \]

is the following problem:

Given an instance \( (V, C) \) of Max-CSP, decide between:

**YES:** \( \text{OPT}(V, C) \geq c \)

**NO:** \( \text{OPT}(V, C) < s \)

An algo. is said to solve a gap problem if says **YES** on **YES** inst, **NO** on **NO** inst.

Claim: If there is a reduction from NPC language to gap-CSP, mapping \( x \) to \( (V_x, C_x) \) s.t.

\[ x \in L \rightarrow \text{OPT}(V_x, C_x) \geq c \]
\[ x \notin L \rightarrow \text{OPT}(V_x, C_x) < s \]

Then it is NP-hard to approximate Max-CSP to within \( c \)

Proof: Assume that \( A \) is a \( \frac{c}{s} \)-approx. algo. for Max-CSP.

Eff. algo for \( L \): Given \( x \), run reduction to get \( (V_x, C_x) \).
Run \( A \) on \( (V_x, C_x) \), denote value by \( v \).

If \( v < s \), output **NO** (because if \( x \in L \), \( \text{OPT}(V_x, C_x) \geq c \) so \( v \geq \frac{c}{s} \cdot c = s \) )

If \( v > s \), output **YES** (because if \( x \notin L \), \( \text{OPT}(V_x, C_x) < s \), so \( v < s \) )
The following two statements are equivalent: \( (\exists c > s > 0 \text{ are constants}) \)

(i) \( \text{NP} \subseteq \text{PCP}_{c,s} [O(\log n), O(1)]_x \)

(ii) There exists a constant \( q \), s.t. \( \text{gap-CSP}_{c,s}(q)_x \) is \( \text{NP} \)-hard.

**Proof** (i) \( \Rightarrow \) (ii). Let \( \text{LENPC} \). We will show a reduction \( x \rightarrow (V_x, C_x) \)

- \( x \in L \rightarrow \text{OPT}(V_x, C_x) \geq c \)
- \( x \notin L \rightarrow \text{OPT}(V_x, C_x) < s \)

Let Ver be an \((c, q)-verifier\) for \( L \) \((q = O(1), r = O(\log n))\) given by (i).

Define \( V_x \) to be the \( \leq 2^q \) Ver's corr. to the proof locations accessible log Ver.

Define \( C_x \) as follows: for each \( p_i \in \text{PCP}_{c,s}[O(\log n), q]_x \) and on a predicate \( \psi(p_i) \in \{0, 1\} \),

Denote \( C = (\psi(p_i), i_1, \ldots, i_q) \) the corr. const.

\( C_x = \{ C \in \text{PCP}_{c,s}[O(\log n), q]_x \} \). This reduction clearly works.

(ii) \( \Rightarrow \) (i). Assume we have a reduction from \( \text{LENPC} \) to \( \text{gap-CSP}_{c,s}(q)_x \)

\( \text{s.t. } x \rightarrow \text{OPT}(V_x, C_x) \geq c \)

\( x \rightarrow \text{OPT}(V_x, C_x) < s \)

We need to prove \( \text{NP} \subseteq \text{PCP}_{c,s}[O(\log n), q]_x \).

Enough to show \( \text{LE} \subseteq \text{PCP}_{c,s}[O(\log n), q]_x \).

Let Ver work as follows: On input \( x \), run the reduction, getting \( (V_x, C_x) \).

Interpret the proof as assign. \( a: V_x \rightarrow \Sigma \). Read \( |C_i| \) random bits to select a random constraint \( \psi(C_x) \in \{0, 1\} \).

Read proof locations \( T_{i_1}, \ldots, T_{i_q} \), and check if they said \( x \).