| | |
|---|---|
| **6.841 Advanced Complexity Theory** | April 18, 2007 |
| Lecture 18 | |
| *Lecturer: Madhu Sudan* | *Scribe: Nadia Benbernou* |

# 1 Probabilistically Checkable Proofs (PCP)

The goal of a probabilistically checkable proof is to verify a proof by looking at only a small number of bits, and probabilistically decide whether to accept or reject. The two resources which PCPs rely on are randomness and queries. A Restricted $(r(n), q(n), a(n))$ PCP verifier is a probabilistic polynomial time verifier with oracle access to a proof $\pi$ that

1. uses at most $r(n)$ random bits.
2. makes $q(n)$ queries to $\pi$.
3. expects answers of size $a(n)$.

**Definition 1** *A language $L$ is in $\mathrm{PCP}_s[r(n), q(n), a(n)]$ if there exists a restricted $(r(n), q(n), a(n))-\mathrm{PCP}$ verifier $V$ with soundness parameter $s$ such that:*

$$\begin{aligned} x \in L &\Rightarrow \exists\, \pi \ s.t.\ \Pr[V^{\pi}(x) \ accepts\ ] = 1 \\ x \notin L &\Rightarrow \forall\, \pi, \qquad \Pr[V^{\pi}(x) \ accepts\ ] < s. \end{aligned}$$

**Theorem 2 (PCP Theorem)** *There exists a global constant $Q$ such that $\forall L \in NP$ there is a constant $c$ such that $L \in \mathrm{PCP}_{1/2}[c\,log\ n, Q, 2]$*

The PCP Theorem was first proved by Arora, Safra, Arora, Lund, Motwani, Sudan, and Szegedy, then later by Dinur.

It is easy to see that $\mathrm{NP} = \cup_{c\in\mathbb{N}}\mathrm{PCP}_0[0, n^c, 2]$, since the proof of membership to a language in NP is polynomial in size, so can just query entire proof and then accept or reject. We also have $\mathrm{NP} = \cup_{c\in\mathbb{N}}\mathrm{PCP}_{1/2}[c\log n, n^c, 2]$, to see the forward inclusion that $\mathrm{NP} \subseteq \cup_{c\in\mathbb{N}}\mathrm{PCP}_{1/2}[c\log n, n^c, 2]$ can simulate $\log n$ bits of randomness (just enumerate over all random strings, count the number of accepting and rejecting configurations, and then output decision).

# 2 MAX-k-SAT

**Definition 3** *The promise problem* MAX-k-SAT *is given by:*

$\Pi_{Yes} = \{\phi \,|\, \phi$ *is a k-cnf formula and all clauses can be simulaneously satisfied*$\}$

$\Pi_{No} = \{\phi \,|\, \phi$ *is a k-cnf formula and any assignment satisfies* $< 1 - \epsilon$ *fraction of the clauses*$\}$

**Corollary 4** *There are constants $k, \epsilon > 0$ such that MAX$-k-$SAT is NP-hard to approximate within $1-\epsilon$.*

Applying the PCP-theorem to SAT. There is a proof system $\pi$ such that verifier can query this string in several locations, and if it says one all the time, then $\phi$ is satisfiable, and if it says zero at least 1/2 the time, then $\phi$ is unsatisfiable. Given a formula $\phi$ as input, we'd like to output a MAX $-\ k\ -$ SAT instance that is in $\Pi_{\text{Yes}}$ if $\phi$ is satisfiable and is in $\Pi_{\text{No}}$ if the formula is unsatisfiable. Enumerate all random strings of length $r(n)$: $(r_1, \ldots, r_{2^{r(n)}})$. On random string $r_i$, the verifier queries some locations of $\pi$ and computes some function $f_i$ of them, and outputs 0 or 1. The function $f_i$ depends only on a constant number of variables $Q$ (the number of queries to $\pi$).

For each $i$, take $f_i$ and write it as a $Q$-cnf formula $\psi_{f_i}$. $f_i$ is a function of $Q$ variables, so there are at most $2^Q$ clauses in $\psi_{f_i}$. Combining these formulae over all $i$ into the expanded boolean formula $\psi = \psi_{f_1} \wedge \cdots \wedge \psi_{f_{2^{r(n)}}}$, there are at most $2^{r(n)} \cdot 2^Q$ clauses in $\psi$. These clauses are in the variables of the proof. If $\phi$ is satisfialbe, then there is a proof $\pi$ for which all of the $f_i$'s will accept, and hence each clause in $\psi$ is satisfied. And if $\phi$ is not satisfiable then for all proofs $\pi$, at least 1/2 of the $f_i$'s will reject. Hence for at least half of the $f_i$, there is at least one clause out of the $2^Q$ clauses in $\phi_{f_i}$ which is false. Hence $\forall\pi$, at least $\frac{1}{2} \cdot \frac{1}{2^Q}$ fraction of the clauses of $\psi$ are unsatisfied. Thus we have an algorithm which given a formula $\phi$ as input outputs a MAX $-\ Q\ -$ SAT instance.

# 3 Generalized Graph Coloring (GGC)

A *GGC* instance is a 4-tuple $(V, E, \Sigma, \{c_e\}_{e \in E})$ where $V$ is set of vertices, $E$ is set of edges, $\Sigma$ is set of colors, and $c_e : \Sigma \times \Sigma \to \{\text{TRUE}, \text{FALSE}\}$ is a constraint on edge $e$. For example, in a 3-coloring of a graph, the constraint $c_e$ on each edge $e = (v_i, v_j)$ would just be $A(v_i) \neq A(v_j)$ where $A : V \to \{0, 1, 2\}$ is the color assignment to the vertices.

Let $G$ be a GGC instance. $G$ is satisfiable if $\exists A : V \to \Sigma$ such that $\forall e = (v_i, v_j) \in E$, $c_e(A(v_i), A(v_j)) = \text{TRUE}$. The interesting parameter for these graphs is the unsatisfiability of $G$. Define

$$\text{UNSAT}(G) := \min_{A:V\to\Sigma} \frac{\#\text{ of edges } (v_i, v_j) \text{ s.t. } c_e(A(v_i), A(v_j)) = \text{FALSE}}{|E|}.$$

Suppose exists a polynomial transformation $T$ that takes Boolean formulae to GGC($a$) instances (where $a$ is the number of colors) such that:

- $\phi \in \text{SAT} \Rightarrow T(\phi)$ is satisfiable (i.e. $\text{UNSAT}(T(\phi)) = 0$)
- $\phi \notin \text{SAT} \Rightarrow \text{UNSAT}(T(\phi)) > \epsilon$

Then $\text{NP} \subseteq \cup_{c \in \mathbb{N}} \text{PCP}_{1-\epsilon}(c \log n, 2, a)$. To see why this is true: the proof is the coloring $A$. The verifier picks a random edge $(v_i, v_j)$ and queries the 2 elements $A(v_i)$ and $A(v_j)$, then checks whether this assignment $(A(v_i), A(v_j))$ satisfies the constraint $c_{(v_i, v_j)}$.

# 4 Dinur's Main Theorem

**Theorem 5** *There is a polynomial time transformation $T : GGC(16) \rightarrow GGC(16)$ and an $\alpha > 0$ such that:*

- *If $G$ is satisfiable, then $T(G)$ is satisfiable.*
- *If $UNSAT(G) < \alpha$, then $UNSAT(T(G)) > 2 \cdot UNSAT(G)$.*
- *$Size(T(G)) = O(|G|)$.*

Initially $G$ may have $\text{UNSAT}(G) \in 0, 1/n^2$ (i.e. there is a coloring $A$ for which all constraints are satisfied, or for all colorings $A$ there is at least one constraint out of the $n^2$ constraints which is not satisfied–note that $n^2$ is an upper bound on the number of constraints since number of edges is at most $n^2$). Apply $T$ to $G$ once we amplify this gap to $\text{UNSAT}(T(G)) \in \{0, 2/n^2\}$. Apply $T$ a second time to get $\text{UNSAT}(T(T(G))) \in \{0, 4/n^2\}$, a third time to get $\text{UNSAT}(T \circ T \circ T(G)) \in \{0, 8/n^2\}$, ..., a logarithmic number of times to get $\text{UNSAT}(T \circ \cdots \circ T(G)) \in \{0, \epsilon\}$ for a constant $\epsilon$.

**Definition 6** *A hypergraph is $q$-*uniform *if each hyperedge involves exactly $q$ vertices.*

**Lemma 7** *There exists a transformation*

$$T_1 : GGC(c \text{ colors}, q - \text{uniform}) \rightarrow GGC(2 \text{ colors}, [\log c] \cdot q - \text{uniform})$$

*such that*

$$UNSAT(T_1(G)) = UNSAT(G).$$

**Proof** For a vertex $v_i$ of a hyperedge in $G$ with $A(v_i) \in \{1, 2, \ldots, c\}$, map it

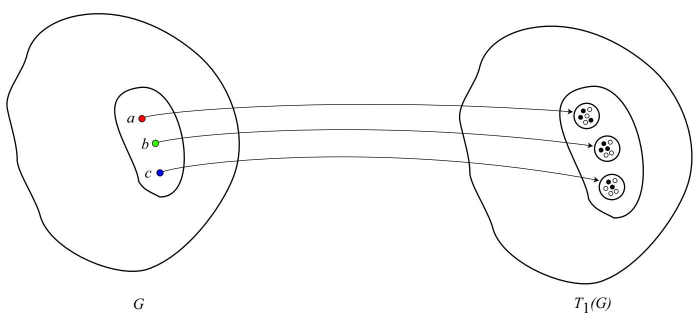


**Figure 1**:

to $x$ vertices $(v'_{i_1}, \ldots, v'_{i_x})$where $x = x_1 \ldots x_{\log A(v_i)})$ is the binary representation the color $A(v_i)$ (hence $x$ is at most $\log c$) and assign each vertex $v'_{i_j}$ the color corresponding to its bit $x_j$ in the binary representation of $c$. See Fig. 1. Hence a $q$-hyperedge which is $c$-colored maps to a hyperedge containing $o(\log c) \cdot q$ vertices which are 2-colored. ■

**Lemma 8** *There exists a transformation*

$$T_2 : GGC(2\ colors, q - uniform) \rightarrow GGC(2\ colors, 3 - uniform)$$

*such that:*

$$UNSAT(G) = 0 \Rightarrow UNSAT(T_2(G)) = 0,$$

$$UNSAT(T_2(G)) > \frac{1}{2^{q+2}} \cdot UNSAT(G).$$

**Lemma 9** *There exists a transformation*

$$T_3 : GGC(c\ colors, q - uniform) \rightarrow GGC(c^q\ colors, 2 - uniform)$$

*such that:*

$$UNSAT(G) = 0 \Rightarrow UNSAT(T_3(G)) = 0,$$

$$UNSAT(T_3(G)) > \frac{1}{q} UNSAT(G).$$

**Proof** Given a $c$-coloring of a $q$-uniform hypergraph $G$, we create a graph $T_3(G)$ with vertices consisting of $V(G)$ and an additional vertex corresponding to each hyperedge of $G$. We join each hyperedge vertex $e_h$ to the $q$ vertices involved in the hyperedge $h = (v_1, \ldots, v_q)$. So the number of edges in $T_3(G)$ is $q \cdot |E|$, where $|E|$ is number of edges in $G$. See Fig. 2

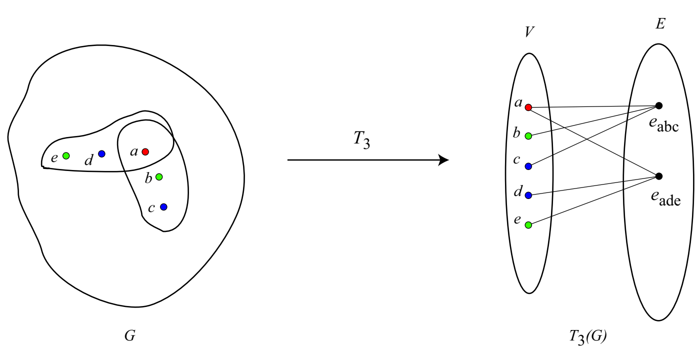


**Figure 2**:

And we assign "vertex set" vertices one of $c$ colors, and hyperedge vertices $e_h$ a $q$-tuple of colors where each position of the tuple can take on one of $c$ colors. Let $A' : V \rightarrow [c] \times \cdots \times [c] = [c]^q$ be a $c^q$ coloring of $T_3(G)$.

For an edge $(v_i, e_h)$ in $T_3(G)$ where $h = (v_1, \ldots, v_i, \ldots, v_q)$, define the edge constraint for $T_3(G)$ to be:

1. $A'(v_j)$ is the color corresponding to the $j^{th}$ position of $A'(e_h)$.

2. The $q$-tuple of colors assigned to $e_h$ satisfy the hyperedge constraint for $G$ (i.e. $c(A'(e_h)) =$ TRUE).

The first constraint ensures that for an edge $(v_j, e_h)$ we have $A'(e_h)$ of the form $(A(v_1), \ldots, A(v_j), \ldots, A(v_q))$ where color assigned to the $j^{th}$ position of $A'(e_h)$ matches the color assigned to $v_j$ $A(v_j)$.

Let

$$\text{UNSAT}(G) = \min_{A:V \to [c]} \frac{\#\text{of unsatisfied hyperedges in } G}{|E|} = \epsilon.$$

Suppose for contradiction that

$$\text{UNSAT}(T_3(G)) < \frac{\epsilon}{q}.$$

We have

$$\text{UNSAT}(T_3(G)) = \min_{A':V \to [c]^q} \frac{\#\text{of unsatisfied edges in } T_3(G)}{q \cdot |E|} < \frac{\epsilon}{q}.$$

Hence the #unsatisfied edges in $T_3(G)$ < #unsatisfied edges in $G$. If we apply the coloring of the "vertex set" vertices in $T_3(G)$ to the vertices of $G$, then each unsatisfied edge in $T_3(G)$ will yield at most one unsatisfied edge in $G$ (namely if the second condition of $T_3(G)$ is violated then the hyperedge in $G$ will not be properly colored). And all other hyperedges of $G$ will be satisfied since satisfied edges $(v_i, e_h)$ in $T_3(G)$ correspond to satisfied hyperedge $e_h$ in $G$. But this induces a $c$-coloring of $G$ with

$$\text{UNSAT}(G) \leq \frac{\#\text{of unsatisfied edges in } T_3(G)}{|E|} < \epsilon,$$

contradictng the fact that $\text{UNSAT}(G) = \epsilon$ ■