6.841/18.405J: Advanced Complexity Theory	April 17, 2002
Lecture 16	
Lecturer: Madhu Sudan	*Scribe: Austin Che*

Recall $\mathrm{PCP}_{c,s}[r,q]$

- Verifier tosses $r(n)$ coins.
- Queries the proof oracle with $q(n)$ bits.
- Completeness $c(n)$. Omit if $c = 1$.
- Soundness $s(n)$. Zero subscripts means $c = 1$ and $s = \frac{1}{2}$.

Last time $\mathrm{NP} = \mathrm{PCP}_{\frac{1}{2}+\epsilon}[O(\log n), 3]$ [Håstad]

Proposition 1 $\mathrm{NP} = \mathrm{PCP}_{\frac{1}{2}+\epsilon}[O(\log n), 3] \implies$ *MaxSAT is hard to approximate to within* $\frac{15}{16} + \epsilon'$.

MaxSAT is the problem of satisfying as many clauses as possible.

Today A weaker statement: $\mathrm{NP} \subseteq \mathrm{PCP}_{1,\frac{1}{2}}[\text{poly}\log, \text{poly}\log]$

1. Set up an algebraic promise problem (GapPCS).
2. Show it is NP-hard (similar to IP=PSPACE proof).
3. Give a PCP verifier for this problem.

Constraint Satisfaction Problems

$x_1, \ldots, x_n$ (variables)
$c_1, \ldots, c_t$ (constraints)
Find an assignment to the n variables such that all (or many) of the constraints are satisfied.

Examples:

1. 3SAT. x_i boolean. $c_j = x_{i_1} \vee x_{i_2} \vee \overline{x_{i_3}}$.
2. 3COL. x_i tertiary (colors). $c_j =$ "$x_{i_1} \neq x_{i_2}''$

Polynomial Constraint Satisfaction Problems

F is a field and $|F|^m = n$. Also, have a degree parameter d and $|F| \gg d$.
In 3SAT, an assignment $A : [n] \to \{0, 1\}$.
Here, assignments will be functions $f : F^m \to F$. Variables will be vectors in F^m.
Constraints will be $c_j = (A_j, x_1^{(j)}, x_2^{(j)}, \ldots, x_k^{(j)})$. A_j is an algebraic circuit $F^k \to F$.
A constraint c_j is satisfied by f if $A_j(f(x_1^{(j)}), f(x_2^{(j)}), \ldots, f(x_k^{(j)})) = 0$.

GapPCS

We define a promise problem based on the polynomial constraint satisfaction problem.

- YES instances: $\exists f : F^m \to F$ that satisfies all constraints and f is a degree d polynomial.
- NO instances: $\forall f : F^m \to F$ that are of degree d, at least 90% of constraints are unsatisfied by f.

Hardness of GapPCS

To show that GapPCS is NP-hard, we reduce SAT on N variables to GapPCS in time $|F|^m$ with:

- $k, d = (\log N)^3$
- $m \geq \frac{\log N}{\log \log N}$
- $|F| \approx (\log N)^{10}$
- $t \approx |F|^m$

Then the reduction is done in polynomial time in N:

$$|F|^m = \left((\log N)^{10}\right)^{\frac{\log N}{\log \log N}} = 2^{\frac{10 \log N \log \log N}{\log \log N}} = 2^{10 \log N} = N^{10}$$

PCP Verifier for GapPCS

1. Expect to be given a proof oracle $f : F^m \to F$.
2. Verifier tests that f is close to some degree d polynomial $p : F^m \to F$ (low degree testing).
3. Build an oracle computing $p : F^m \to F$" from oracle for $f : F^m \to F$ (self correction).
4. Pick random j and verify c_j is satisfied by p (not f).

We note that

- Self correction can be done in time poly(m, d)
  (Contrast with the number of coefficients of p: $(\frac{d}{m})^m \leq \#$ coeffs $\leq d^m$)
- Low degree testing can also be done in time poly(m, d).
- To verify c_j, the verifier needs to make poly $\log N$ queries.
- We need $\log t$ random bits to select j, for low degree testing, we need $O(m \log |F|) = O(\log n)$ bits, and for self correction, we need $m \log |F|$ bits. This shows that the verifier uses $O(\log N)$ random bits.

Self Correction

- Given oracle $f : F^m \to F$ such that there exists a polynomial $p : F^m \to F$ of degree d and

$$\Pr_x[f(x) \neq p(x)] \leq \delta$$

- Also given $a \in F^m$
- Compute $p(a)$. For all a, should be computing $p(a)$ correctly with high probability over internal randomness. We cannot just use $f(a)$ as for some fraction of a, $f(a)$ may be incorrect.

Algorithm

1. Pick $r \in_R F^m$.
2. Take the line $l(t) = (1 - t)a + tr$ and we'll look at p along this line.
3. Let $\tau_1, \tau_2, \ldots, \tau_{d+1}$ be distinct and non-zero elements from F. These do not need to be random.

4. Compute coefficients of $h : F \to F$ of degree d such that $h(\tau_i) = f(l(\tau_i))$.

5. Output $h(0)$.

Claim 2 *Self correction outputs $p(a)$ with probability $\geq 1 - (d+1)\delta$.*

Proof $l(\tau_i)$ is a random point in F^m over random choice of r for all non-zero τ_i.

$$\Pr_r[f(l(\tau_i)) \neq p(l(\tau_i))] \leq \delta$$

By the union bound,

$$\Pr_r[\exists i \in \{1, \ldots, d+1\} f(l(\tau_i)) \neq p(l(\tau_i))] \leq (d+1)\delta$$

If the above event does not occur, then for all i, $h(\tau_i) = f(l(\tau_i)) = p(l(\tau_i))$.
So with probability $1 - (d+1)\delta$, $h = p|_l$ and $h(0) = p(l(0)) = p(a)$. ■

The above is due to [Beaver, Feigenbaum] and [Lipton].
They were interested in how to compute a function $f(a)$ without revealing a.

Low Degree Testing

- Given oracle $f : F^m \to F$
- Completeness: if $f = p$ of degree d, then must accept with probability 1.
- Soundness: if $\forall p$ of degree d,

$$\Pr_x[f(x) \neq p(x)] > \delta \implies \text{must reject with high probability}$$

Algorithm

1. Repeat many times
   (a) Pick $a \in_R F^m$ at random.
   (b) Use the self correction algorithm to find $p(a)$ and verify $p(a) = f(a)$.

Theorem 3 (Rubinfeld-Sudan, ALMSS) *Soundness of above algorithm. $\exists \delta_0$ such that if f is δ-far from any polynomial p then f is rejected with probability $min\{\frac{\delta}{2}, \delta_0\}$.*

[Rubinfeld, Sudan] showed $\delta_0 = O(\frac{1}{d})$ and [ALMSS] showed that $\delta_0 = 10^{-3}$.