6.841 Advanced Complexity Theory April 13, 2005

Lecture 19

Lecturer: Madhu Sudan *Scribe: Kevin Matulef*

1 Today

- We finish the proof that NP $\subseteq$ PCP[poly $\log n$, poly $\log n$].
- We give another PCP construction, showing NP $\subseteq$ PCP[poly$(n), O(1)$].

2 Review: Graph Coloring PCP

Last time we began a PCP construction for graph 3-colorabilty. Recall that

- The vertices were labeled by elements of some sufficiently large field $\mathbb{F}$.
- The edges were given by a function $E : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$.

The prover writes down

- A coloring given by a polynomial $\chi : \mathbb{F} \to \mathbb{F}$ where degree $\chi \leq n$
- A polynomial $A : \mathbb{F} \to \mathbb{F}$ defined by $A(x) = \chi(x)(\chi(x) - 1)(\chi(x) - 2)$ so that $A|_V = 0$.
- A polynomial $B : \mathbb{F}\times\mathbb{F} \to \mathbb{F}$ defined by $B(x, y) = E(x, y) \prod_{j\in\{-2,-1,1,2\}}(\chi(x)-\chi(y) - j)$ so that $B|_{V\times V} = 0$

Last time we discussed how a verifier can check this proof. All we have left to show is how the verifier can check that the function tables for χ, A, B, etc. indeed represent low-degree polynomials.

3 Low Degree Test

Our task is thus: given an oracle $f : \mathbb{F}^n \to \mathbb{F}$, and an integer d, we want to sample f on only q locations, and then

- If $\mathbb{F}$ is an m-variate degree d polynomial $\Rightarrow$ accept wp 1.
- If $\mathbb{F}$ is δ-far from every such polynomial $\Rightarrow$ reject wp $\geq \epsilon$.

Here, by δ-far, we mean $\Pr_{x \in \mathbb{F}^m}[f(x) \neq g(x)] \geq \delta$

The number of queries, q, will depend on $m, d, |\mathbb{F}|, \delta$,and ϵ. In the univariate case, when $m = 1$, it is easy to see that the test must make more than d queries. If it makes fewer that that, then there is always a degree d polynomial consistent with the values it has seen. Thus, it will not have enough information to reject, even if the rest of the table is inconsistent with such a polynomial.

We will give a test that, for constant ϵ, δ (say both equal to 0.1), makes a number of queries q that depends polynomially on d. There is also a small dependence on m, but not enough to matter for our purposes, so we omit it.

It is easy to come up with such a test for univariate polynomials. We simply make $d+1$ queries, interpolate to get a polynomial, then test to see if that polynomial is consistent with the rest of the table by querying another point. More formally, here is our test for $m = 1$.

- Pick distinct $\alpha_1, \alpha_2, ..., \alpha_{d+1} \in \mathbb{F}$.
- Query $f(\alpha_1), f(\alpha_2), ..., f(\alpha_{d+1})$.
- Let $p(x)$ be a degree d polynomial st $p(\alpha_i) = f(\alpha_i)$ for all i.
- Pick $\alpha_{d+2} \in \mathbb{F}$ randomly.
- Query $f(\alpha_{d+2})$.
- Check if $p(\alpha_{d+2}) = f(\alpha_{d+2})$. If so, *accept*. Otherwise, *reject*.

To generalize this to m-variate polynomials, we can pick a random line, and then test that the points on this line are consistent with a univariate degree d polynomial. More formally, here is the test for arbitrary m:

- Pick $\beta, \gamma \in \mathbb{F}^m$.
- Let $\ell_{\beta,\gamma} : \mathbb{F} \to \mathbb{F}^m$ be defined by $\ell_{\beta,\gamma}(t) = \beta t + \gamma$.
- Let $f|_{\ell_{\beta,\gamma}} : \mathbb{F} \to \mathbb{F}$ be defined by $f|_{\ell_{\beta,\gamma}}(t) = f(\ell_{\beta,\gamma}(t))$.
- Verify that $f|_{\ell_{\beta,\gamma}}$ is a univariate polynomial of degree d using the univariate test given earlier.

Intuitively, this is a very natural extension of the univariate test. However, proving that it succeeds with sufficiently high probability is not easy, because it is hard to characterize those functions which are far from multivariate polynomials. Nonetheless, it can be shown that the test works. While we will not prove it here, we state the following (the full proof can be found in a paper by Arora, Lund, Motwani, Sudan, and Szegedy):

Lemma 1 *If* $\Pr[\textit{test rejects } f] \leq \epsilon$ *(for* $\epsilon \approx 0.1$*, and* $|\mathbb{F}| \geq \mathrm{poly}(\frac{d}{\epsilon\delta m})$*), then* f *is* 2ϵ*-close to some polynomial* p.

4 Reducing the Number of Queries

Now we have shown how to test that polynomials have degree d, by making poly(d) queries. However, in the PCP construction that we have given so far, we need to test that some polynomials have degree n. This requires making poly(n) queries into the proof. But this is silly, since 3-colorability has proofs of size $O(n)$. With poly(n) queries, we could just look at the whole proof. To fix this, we change the way the proof is encoded. Instead of encoding the vertices as elements of a field $\mathbb{F}$, we let $\mathbb{F}$ be a smaller field, and encode vertices as vectors in $\mathbb{F}^m$. Specifically:

- Let $|\mathbb{F}| \approx \log^3 n$.
- Let $H \subseteq \mathbb{F}$, $|H| = \log n$.
- Let $V \approx H^m$, so $m = \frac{\log |V|}{\log |H|} = \frac{\log n}{\log \log n}$.
- Now $E : H^m \times H^m \to \{0, 1\}$, and we extend to get $\hat{E} : \mathbb{F}^m \times \mathbb{F}^m \to \mathbb{F}$.

Accordingly, the prover changes the domain of the functions written down. So the prover writes:

- $\chi : \mathbb{F}^m \to \mathbb{F}$
- $A : \mathbb{F}^m \to \mathbb{F}$.
- $B : \mathbb{F}^m \times \mathbb{F}^m \to \mathbb{F}$

Note that $|\mathbb{F}^m| = ((\log n)^3)^{\frac{2 \log n}{\log \log n}} = n^6$, so the size of the proof is still polynomial. Now the verifier's task is to:

- verify the degrees of χ, A, B, etc. But notice now that the degree of each of these polynomials is $O(\text{poly} \log n)$ instead of n. Since the polynomials are m-variate instead of univariate, they needn't be of high degree to have the correct values on n or n^2 points.
- verify that χ is consistent with A, and that χ and E are consistent with B.
- verify that $A|_{H^m} = 0$ and $B|_{H^m \times H^m} = 0$. This can be done using the method outlined in the last lecture. Notice that $A(\overline{\alpha}) = 0$ for all $\overline{\alpha} \in H^m$, if and only if A can be written as $A(x) = \sum_{i=1}^m A_i(x) g(x)$ where $g(x) = \prod_{\beta \in H} (x - \beta)$ and $A_1, ..., A_m$ are polynomials.

  Thus, we modify the protocol so that the prover sends the additional polynomials $A_1, ..., A_m : \mathbb{F}^m \to \mathbb{F}$ and $B_1, ..., B_{2m} : \mathbb{F}^{2m} \to \mathbb{F}$. The verifier then tests the degrees of $A_1, ..., A_m$ and $B_1, ..., B_{2m}$. The verifier also tests that A is of the correct form by checking $A(\gamma) = \sum_{i=1}^m A_i(\gamma) g(\gamma)$ for some random γ. A similar test is done for B.

This completes the description of the PCP construction, showing that NP $\subseteq$ PCP$[\text{poly} \log n, \text{poly} \log n]$.

5 A Different PCP

Now we outline a different PCP construction where the verifier uses a polynomial amount of randomness but only makes a constant number of queries into the proof. To start, consider the following problem (it is an exercise to show that this is NP-hard). Given m degree-2 polynomials $P_1, ...P_m$ on n variables over $GF(2)$, find an assignment to $x_1, ...x_n \in GF(2)$ that makes all the polynomials simultaneously evaluate to zero.

Suppose $a_1, ...a_m$ is such an assignment. To convince the verifier of this, the prover writes down

- for every linear function $\ell : \mathbb{F}_n^n \to \mathbb{F}_2$, the value $\ell(\overline{a})$. Store all these values in a table T_1. There are 2^n linear functions (parity functions) so the table has length 2^n.
- for every quadratic function $q : \mathbb{F}_2^n \to \mathbb{F}_2$, the value $q(\overline{a})$. Store all these values in a table T_2. There are 2^{n^2} quadratic functions so the table has length 2^{n^2}.

The verifier then

1. verifies $\exists a$ st $T_1[\ell] = \ell(a)$ for most ℓ
2. verifies $\exists a$ st $T_2[q] = q(a)$ for most q.
3. Picks $r_1, ...r_m \in \{0, 1\}$ at random, then verifies that the polynomial $q(a) \triangleq \sum r_i P_i(a)$ is zero.

Step 1 is accomplished by choosing ℓ_1 and ℓ_2 at random, and checking that $T_1[\ell_1] + T_1[\ell_2] = T_1[\ell_1 + \ell_2]$. This is an application of the well-known linearity test by Blum, Luby, and Rubinfeld. We will not analyze it here.

Step 2 is accomplished with two separate checks. The verifier first chooses q_1 and q_2 randomly, then checks that $T_2[q_1] + T_2[q_2] = T_2[q_1 + q_2]$. Next the verifier checks that $T_1[\ell_1] \cdot T_1[\ell_2] = T_2[q_2 + \ell_1 \cdot \ell] - T_2[q_2]$.

Step 3 is accomplished as follows: the verifier picks a random quadratic function q_2 on n variables, and checks that $T_2[q_2 + q] = T_2[q_2]$. As long as T_2 is only corrupted a small amount, this test is likely to pass when a makes all the P_i's (and hence q) zero.

The total number of queries performed in all three steps is just 12, which is constant. Notice however that each query requires producing $O(n)$ or $O(n^2)$ bits simply to index into the tables T_1 or T_2. This shows that NP $\subseteq$ PCP$[\text{poly}(n), O(1)]$.

6 Further Reducing the Number of Queries

One way to reduce the number of queries further is to perform just one of the checks at random from the checks above. This reduces the number of queries to 4, at the cost of a higher error rate.

Hastad has given a PCP construction that goes even further and reduces the number of queries to a mere 3. We will attempt to give a sketch of his approach here (DISCLAIMER: I do not understand what follows. After this point I am mindlessly transcribing from my notes).

Conisder the following formulation of satisfiability. Given a function f on n variables over GF(2), find a such that $f(a) = 1$.

Now suppose $a_1, ..., a_n$ is a satisfying assignment. Let T be a table which holds the value $g(a)$ for every possible function g. Hastad performs a test that looks something like checking $T[g_1] + T_[g_2] = T[g_1 + g_2 + \eta]$ where η is chosen randomly. Actually, this is insufficient and the test is more like $T[g_1 \wedge f] + T_[g_2 \wedge f] = T[(g_1 + g_2 + \eta) \wedge f]$. To disallow a table of all zeros from passing the test, the prover is asked to only write down half of the functions g. The other half is implicity specified as the complement of the functions written down. Thus if the prover writes down zeros, the implicit complement functions will yield ones.