Lecture 24

In this lecture, we continue the construction of a PCP-verifier that uses $O(\log n)$ random bits and looks at a constant number of places in the proof. We also give a description of the verifier that reads only a constant number of bits but uses polynomial number of random bits.

1 Verifier that accesses to a constant number of places

Last time, we introduced the use of curves to encode the evaluations of the polynomial. We review the lemma from the last lecture.

Lemma 1

Given 3SAT formula ϕ, one can construct a sequence of constraints of the form $(x_1, \dots, x_c, T)$ where

$x_1, \dots, x_c \in \mathcal{F}$,
$T : \mathcal{F}^c \to \{0,1\}$,
$|\mathcal{F}| = O(\log^3 n)$,
$m = \log n / \log\log n$,
$m' = 3m + 3$,
$h = \log n$, *and*
$d = 2hm' + m' + 2$,

such that

- *if ϕ is satisfiable, there is a polynomial $p : \mathcal{F}^{m'+1} \to \mathcal{F}$ of degree d such that for every constraint $(x_1, \dots, x_c, T)$, $T(p(x_1), \dots, p(x_c)) = 1$.*
- *if ϕ is not satisfiable, for every polynomial $p : \mathcal{F}^{m'+1} \to \mathcal{F}$ of degree d such that for at least half of the constraints $(x_1, \dots, x_c, T)$, $T(p(x_1), \dots, p(x_c)) = 0$.*

1.1 The proof

The proof consists of 3 parts, namely, f, L, and $Curve$. The polynomial f is the proof, the table L is for checking that if f is close to some polynomial p, and the table $Curve$ is for evaluating p in many points by only looking at a constant number of places. Each of the entries of the proof with the claim from the prover is described below.

- $f : \mathcal{F}^{m'+1} \to \mathcal{F}$.

  Claim: $f \equiv p$ where p is the degree d polynomial that satisfies all constraints.

- $L : (\mathcal{F}^{m'+1} \times \mathcal{F}^{m'+1}) \to \mathcal{F}^{d+1}$.

  Claim: for all $\bar{x}, \bar{y} \in \mathcal{F}^{m'+1}$, $L(\bar{x}, \bar{y})$ describes a degree-d univariate polynomial q such that $q(t) = f(\bar{x} + t\bar{y})$.

- $Curve : (\mathcal{F}^{m'+1})^{c+1} \to \mathcal{F}^{cd+1}$.

  For all $\bar{x}_0, \ldots, \bar{x}_c \in \mathcal{F}^{m'+1}$, let $C : \mathcal{F} \to \mathcal{F}^{m'+1}$ be a curve passing through $x_0, \ldots, x_c$ at $t_0, \ldots, t_c$.

  Claim: $Curve(x_0, \ldots, x_c)$ describes a degree cd univariate polynomial q such that $q(t) = f(C(t))$.

1.2 The verifying procedure

The verifier proceeds as follows.

Verifying procedure:
  repeat $O(1)$ times *{for testing if $f \approx p$}*
    pick random $\bar{x}, \bar{y} \in \mathcal{F}^{m'+1}$ and $t \in \mathcal{F}$.
    let $q = L(\bar{x}, \bar{y})$, reject if $q(t) \neq f(\bar{x} + t\bar{y})$.
  repeat $O(1)$ times *{for retrieving the evaluations of p}*
    pick a random constraint $(x_1, \ldots, x_c, T)$.
    pick a random $x_0 \in \mathcal{F}^{m'+1}$.
    let $q = Curve(x_0, x_1, \ldots, x_c)$.
    let C be a degree-d curve passing through $x_0, \ldots, x_c$.
    pick a random $t \in \mathcal{F} \setminus \{t_1, \ldots, t_c\}$.
    reject if $q(t) \neq f(C(t))$.
    reject if $T(q(t_1), \ldots, q(t_c)) = 0$.
  accept.

Essentially, the step that checks if the formula is satisfiable is the last one that uses the constraint $(x_0, \ldots, x_c, T)$. The other parts of the procedure are for verifying the consistency of the proof and for retrieving the evaluations of the polynomial at c places. Clearly, if the polynomial q in the procedure represents the correct polynomial as described in the first part of the proof, using lemma 1, the test in the last step will correctly verify the proof.

We assume that the proof f passes the low-degree test, described in the last lecture, in the first step; therefore, it is close to some polynomial p. Now, we will prove the correctness of this verifier by proving that the second part of the procedure that outputs the evaluations of the polynomial is correct. The following procedure describes the same procedure used by the verifier.

  pick random $x_0 \in \mathcal{F}^{m'+1}$.
  let $q = Curve(x_0, \ldots, x_c)$.
  let C be a curver of degree c passing through $x_0, \ldots, x_c$.
  pick random $t \in \mathcal{F} \setminus \{t_1, \ldots, t_c\}$.
  reject if $q(t) \neq f(C(t))$.
  output $q(t_1), \ldots, q(t_c)$.

The following lemma states that the procedure outputs the correct evaluations with large-enough probability.

LEMMA 2
If $f : \mathcal{F}^{m'+1} \to \mathcal{F}$ has agreement $(1-\delta)$ with a degree-d polynomial $p : \mathcal{F}^{m'+1} \to \mathcal{F}$, then for every $x_1, \ldots, x_c \in \mathcal{F}^{m'+1}$ and for every function $Curve : (\mathcal{F}^{m'+1})^{c+1} \to \mathcal{F}^{cd+1}$, the polynomial evaluation procedure above, conditioned on not rejecting, outputs $p(x_1), \ldots, p(x_c)$ with probability at least $1 - 2\sqrt{\delta} - \frac{cd}{|\mathcal{F}|-c}$.

PROOF: We will bound the probability that the procedure accepts and gives the wrong answer. Let $x_1, \ldots, x_c \in \mathcal{F}$ and $t \notin \{t_1, \ldots, t_c\}$ be fixed. Consider C a degree-d curve $C(t_i) = x_i$ for $i = 0, \ldots, c$. We claim without proof that $C(t)$ is uniformly distributed in $\mathcal{F}^{m'+1}$ for $t \notin \{t_1, \ldots, t_c\}$. Therefore,

$$\mathbf{E}_{x_0}[\text{fraction of } t \notin \{t_1, \ldots, t_c\} \text{ such that } f(C(t)) \neq p(C(t))] \leq \delta.$$

Using Markov's Inequality, we have

$$\mathbf{Pr}_{x_0}[(\text{fraction of } t \notin \{t_1, \ldots, t_c\} \text{ such that } f(C(t)) \neq p(C(t))) \geq \sqrt{\delta}] \leq \sqrt{\delta},$$

and hence

$$\mathbf{Pr}_{x_0}[\ f(C(t)) = p(C(t)) \text{ for at least } 1-\sqrt{\delta} \text{ fraction of } t \notin \{t_1, \ldots, t_c\}] \geq 1 - \sqrt{\delta}.$$

Now, consider the case that C is such that $f(C(t)) = p(C(t))$ for $1-\sqrt{\delta}$ fraction of $t \in \mathcal{F} \setminus \{t_1, \ldots, f_c\}$. We note that the polynomial q of degree cd is uniquely specified by $Curve(x_0, \ldots, x_c)$. Because q is of degree cd, either (1) $q(t) = p(C(t))$ or (2) $q(t)$ and $p(C(t))$ agrees on at most cd points. In case (1), the output of the procedure is correct.

Hence, suppose that we are in case (2). We have that $q(\cdot)$ and $p(C(\cdot))$ agree on less than $\frac{cd}{|\mathcal{F}|-c}$ fraction of $\mathcal{F} \setminus \{t_1, \ldots, t_c\}$. Also, $p(\cdot)$ and $f(\cdot)$ agree on $\geq 1 - \sqrt{\delta}$ fraction of $\mathcal{F} \setminus \{t_1, \ldots, t_c\}$, i.e., they disagree on less than $\sqrt{\delta}$ fraction. Therefore, $q(\cdot)$ and $f(\cdot)$ agree on less than $\sqrt{\delta} + \frac{cd}{|\mathcal{F}|-c}$ of $\mathcal{F} \setminus \{t_1, \ldots, t_c\}$.

The procedure accepts in case (2) with probability at most $\sqrt{\delta} + \frac{cd}{|\mathcal{F}|-c}$. Taking into account the probability at most $\sqrt{\delta}$ for the case that $f(C(t))$ and $p(C(t))$ has too few agreements, the probability that the procedure accepts and outputs the wrong answer is at most $\sqrt{\delta} + \sqrt{\delta} + \frac{cd}{|\mathcal{F}|-c}$. The lemma follows. □

Note that the lemma above holds for any function $Curve$.

The verifying procedure in this section reads the proof 2 times in the first part and another 2 times in the second part. Therefore, the verifier reads $O(1)$ many places.

From here to $\mathbf{PCP}[O(\log n), O(1)]$... We have to encode the proof again, and verify that the encoding satisfies the same property.

2 Verifier that reads a constant number of bits

In this section we will give the first part of the proof that $\mathbf{NP} \subseteq \mathbf{PCP}[poly(n), O(1)]$. We show that we can construct the proof of exponential size and give a verifier that reads only a constant number of bits. The proof of correctness will be in the next lecture.

Our favorite problem, 3SAT, is not suitable for this reduction. We will use another **NP**-complete problem.

DEFINITION 1 *Given a degree-2 polynomial* $p_1(x_1, \ldots, x_n), \ldots, p_m(x_1, \ldots, x_n)$ *over* Z_2*, the problem* COMMON ROOT *is the problem of deciding if there are* $a_1, \ldots, a_n$ *such that*

$$p_i(a_1, \ldots, a_n) = 0,$$

for $i = 1, \ldots, m$.

The following lemma shows that COMMON ROOT is **NP**-complete.

LEMMA 3
COMMON ROOT *is* **NP**-*complete.*

PROOF: Given a 3CNF formula ϕ over $x_1, \ldots, x_n$ with m clauses, we can construct $2m$ polynomials p_i of degree 2 over $x_1, \ldots, x_n, y_1, \ldots, y_m$ such that ϕ is satisfiable iff these polynomials have a common root.

An example for such construction is as follows. For a clause $(x_1 \wedge x_2 \wedge \bar{x_3})$, two polynomials will be created: $y + (1 - x_1)(1 - x_2) - 1$ and $(x_3)(1 - y)$. □

Suppose there is a that satisfies $p_1, \ldots, p_m$, i.e., $p_j(a_1, \ldots, a_n) = 0$ for all $j = 1, \ldots, m$. We will construct the proof, which is an encoding of a, of exponential size. The proof consists of 2 parts:

- $A : \{0,1\}^n \to \{0,1\}$.
  claim: $A(x_1, \ldots, x_n) = \sum_i a_i x_i \bmod 2$.
- $B : \{0,1\}^{n \cdot n} \to \{0,1\}$.
  claim: $B(z_{11}, \ldots, z_{nn}) = \sum_{i,j} a_i a_j z_{ij} \bmod 2$.

The first part of the proof is all possible linear functions in a, and the second one is all possible degree-2 polynomials in a. This encoding is somewhat an opposite way of encoding polynomials. In polynomial codes, the code words are usually the evaluations of a polynomial on every points in the field. In this encoding, we fix a point an evaluate every polynomial at that point, instead. We note that B is not a linear code of a, i.e., B is of degree-2. However, B itself is a linear function.

The encoding described above is very useful. For example, if the encoding is correct, we can evaluate any degree-2 polynomial $p(y_1, \ldots, y_n) = c_0 + \sum_i c_i y_i + \sum_{i,j} c_{ij} y_i y_j$ by reading at 2 bits of the proof. Therefore, to check that the assignment a satisfies any polynomial p, one needs to verify that $c_0 + A(c_1, \ldots, c_n) + B(c_{11}, \ldots, c_{nn}) = 0$.

Moreover, checking the consistency of these two tables can be done with a constant number of accesses. If A and B are close to linear functions, i.e., $A(x_1, \ldots, x_n) = \sum a_i x_i$ for a large fraction of $x_1, \ldots, x_n$, and $B(z_{11}, \ldots, z_{nn}) = \sum b_{ij} z_{ij}$ for a large fraction of $z_{11}, \ldots, z_{nn}$, then we can check that $b_{ij} = a_i a_j$ for every i, j with $O(1)$ queries. Also, if A and B are not close to linear, we can check that with $O(1)$ queries.

We will present the verifier for this proof in the next lecture.