1 Techniques for Parallelization

Lecture 6: May 21 and 28, 2003

Our goal is to prove:

Theorem 1 *For every integer $k \geq 1$, there are constants c_1, c_2, c_3, positive integer p (that will be larger than k) and $e < 1$ such that there is a*

$$(k, r = c_1 \log n, p, c = O(\log^{c_2} n), a = O(\log^{c_3} n), e)$$

inner verifier.

In doing so we shall also modify the proof from Lectures 3 and 4 that $NP \subset PCP(\log n, \text{polylog } n)$ so that the number of locations probed in the PCP witness is only a constant. We remark that if our goal is merely to minimize the number of probes, then they can be reduced to two as done in [4, 3]. These works predated [2, 1], and not much attention is given there to minimizing the number of random bits used, which is $O(\log^c n)$ rather than $O(\log n)$. It may be an interesting research question (with potential connections to simplification of the proof of the PCP theorem, or improved NP-hardness of approximation results for set cover) to check now in hindsight whether two probes suffice for $NP \subset PCP(\log n, \text{polylog } n)$, without using the full machinary of composing inner verifiers.

The reader is encouraged to refresh his/her memory regarding the construction of the PCP in Lectures 3, and the requirements from inner verifier in Lecture 5.

1.1 The code E - a degree-h multinomial

Recall that for inner verifiers we required that the contents of locations in the outer PCP be encoded using some error correcting code E. The specific code that we use (or rather, the honest prover uses) is the degree-h extension given in Lecture 3. Note that we already used it there in order to encode the satisfying assignment held by the PCP prover.

1.2 The degree-h test revised

In Lecture 3 we presented a degree-h test (that was analysed in Lecture 4) on a table A. The basic test calls for picking an aligned line and a random point

y on the line. On this aligned line, we find the unique degree-h polynomial p that agrees with the values given by A to the first $h+1$ points on the line, and then reject unless $p(y) = A(y)$. (Namely, the value given by the table A in location y is the same as the value predicted by the polynomial, which with some abuse of notation we denote by $p(y)$.)

As described, the basic test requires probing A at $h+2$ locations. As our goal here is to reduce the number of probes to a constant, we slightly modify the PCP. We add to it a table B that contains for every possible aligned line the respective degree-h polynomial. In the modified degree-h test, the verifier makes one probe to B and one probe to A.

The use of the auxiliary table B may potentially offer the prover more ways to cheat. The polynomials written in this table need not be the same as the respective polynomials one would get by interpolating from the first $h+1$ values on the respective aligned line. However, it is not hard to verify that the proof given in Lecture 4 works virtually without change also for this modified degree-h test.

1.3 The total degree test

The proof given in Lecture 4 only shows that application of the degree-h test has probability $\Omega(1/mh)$ of detecting that the function in table A is far from being of degree-h. To get constant probability of success, we proposed to repeat the basic degree-h $O(mh)$ times. But this would defeat our purpose of keeping the number of probes constant.

There may be several approaches in order to get around this problem. We shall present here the approach used in [1]. Rather than test that A is a multinomial that has degree at most h in each variable, we test that the total degree of A is at most $d = mh$. By total degree we mean maximum over all monomials of the sum of the degrees appearing in the monomial. For example, $(x_1)^2 x_2 + x_1(x_2)^2$ is of degree 2 and total degree 3.

To test that A is close to a function of total degree d, we pick two random points $x, y \in \mathcal{F}^m$, and consider the parametric line $x + ty$. If A is a multinomial of degree d, then the values of A for points on this line form a univariate polynomial in t of degree d. Note that every line contains $|\mathcal{F}|$ points, hence there are $|\mathcal{F}|(|\mathcal{F}|-1)$ choices of x and y that give the same line. For the test, the prover needs to prepare a table B, where each entry in B is the polynomial associated with one line, and each line is represented only once (e.g., by the lexicographically first x', y' for which the line $l(t) = x' + ty'$ contains the same set of points as the line $x + ty$). The verifier picks two random points $x, y \in \mathcal{F}^m$, probes B for the polynomial $p(t)$ that corresponds to the respective line $l(t) = x' + ty'$, and rejects unless $p(t_x) = A(x)$, where t_x is the unique value for t such that $l(t_x) = x$.

Let $\Delta_d(f)$ denote the hamming distance of function A from the closest multivariate polynomial of degree total d. The following theorem follows from [6, 2, 1, 5].

Theorem 2 *If $|\mathcal{F}|$ is large enough (e.g., $|\mathcal{F}| > d^4$) and $\Delta_d(A) \geq \delta$ then the total degree test rejects with probability at least $\delta/2$.*

Using the analysis in [5], $|\mathcal{F}|$ can be taken to be linear in d.

The proof of this theorem will probably not be given in the course, though Lecture 4 should convince the reader that the statement of the theorem is reasonable. Part of the proof may be given as homework.

Let us note in passing that every two total degree d multinomials that differ in at least one point, differ in at least a $d/|\mathcal{F}|$ fraction of the points. This can be proved by considering all lines that go through the point on which they differ.

The fact that we switched from a degree-h to a total degree $d = mh$ test requires a minor change in the sum check protocol of Lectures 2 and 3. For the proof of correctness, all we can assume (after performing the total degree test) is that the prover is performing the sum check protocol on a multinomial of total degree d. Hence every variable might have degree d. In the sum check protocol, the prover sends back univariate polynomials of degree $O(h)$, and we need now to argue that they disagree with every degree $O(d)$ polynomial on a fraction of roughly $1 - 1/m$ of the points. This may requires $|\mathcal{F}| = \Omega md = \Omega(m^2 h)$, rather than our previous requirement of $|\mathcal{F}| = \Omega(mh)$.

1.4 Sampling A at a nonrandom location

Let A be a table that supposedly represents a multinomial of total degree d, and let B be a table giving for each line the degree d univariate polynomial that agrees with A. If $\Delta_d(A) = \Delta(A, p)$ is small (with p being the closest multinomial of total degree d), then probing A at a random location $x \in \mathcal{F}^m$ is similar to probing p at x. Sometimes we will need to probe A at a nonrandom location x. We would still like to say that the answer that we get is the same that we would get had we probed p at x. To achive this we do as follows.

Error detecting probe.

1. Choose $y \in \mathcal{F}^m$ and $t_r \in \mathcal{F} - \{0\}$ at random.
2. Retrieve from B the degree d univariate polynomial $q(t)$ that correspond to the line $x + ty$.
3. Retrive $A(x)$ and $A(x + t_r y)$ from A.

4. Reject unless $q(0) = A(x)$ and $q(t_r) = A(x + t_r y)$.

5. Output $A(x)$.

A truthful prover need never fail in the above.

Lemma 3 *If for some multinomial p of total degree d we have that $\Delta(A, p) = \epsilon$, then for every table B and every x for which $A(x) \neq p(x)$, the probability that the probing procedure rejects is at least*

$$1 - \epsilon - \frac{d}{|\mathcal{F}| - 1}$$

Proof: Assume that $A(x) \neq p(x)$. Then on every line through x we have one of two cases:

1. $q(t)$ is different from the q that really corresponds to p.

2. $q(t)$ agrees with p of this line, and then the test $q(0) = A(x)$ rejects. For simplicity, when the test $q(0) = A(x)$ rejects, let us replace $q(t)$ by a different polynomial on which this test does not reject (and then the new q disagrees with p).

When q differs from the polynomial induced by p on the respective line, these two degree d polynomials agree on at most d points out of the $|\mathcal{F}|$ points on the line. Note that the lines through x form a disjoint cover of $\mathcal{F}^m$. Let $Q(y)$ be the function on $\mathcal{F}^m$ whose value at $y \in \mathcal{F}^m$ equals the value one gets for y by considering the polynomial q for the line joining x and y. Then q and p agree on a fraction of at most $d/(|\mathcal{F}| - 1)$ of $\mathcal{F}^m$. By the triangle inequality, Q and A agree on a fraction of at most $\epsilon + d/(|\mathcal{F}| - 1)$ of the points. As the "error detecting probe" compares Q and A on a random point, the probability of it failing when $A(x) \neq p(x)$ is at least $1 - \epsilon - d/(|\mathcal{F}| - 1)$. □

Remark: In [1], the test $q(0) = A(x)$ is removed, saving one probe (to $A(x)$), but slightly complicating the analysis. There is a certain twist on the "error detecting probe" of this section, that may be viewed as an "error correcting probe". It works when A is equal to a degree d multinomial almost everywhere, and on input x we have only access to A, but wish to compute $p(x)$ rather than $A(x)$. Then we can take a random line through x, query $d + 1$ points on this line, interpolate a degree d polynomial from it (that supposedly agrees with p), and recover $p(x)$ from q. This idea can be extended to retrieve $p(x)$ even if A and p differ on almost half the points, by picking degree 2 curves through x rather than lines, sampling more than $d+1$ points on the curve, and using the Berklekamp-Welsh procedure to retrieve the closest degree-d polynomial even when there are some errors.

1.5 Comparing the inner and outer verifiers

Recall that the inner verifier expects to see k tables, that we denote by $A_1, \ldots, A_k$, where each table A_i is supposedly the list of values of the encoding $E(X_i)$ for some character X_i that the outer verifier would have read. In addition, there is a PCP witness π. The inner verifier needs to check that π contains an encoding of the same $X_1, \ldots, X_k$.

The encoding E that we use for $A_1, \ldots, A_k$ is via m-variate multimonials of total degree d. More precisely, $E(X_i)$ is the degree-h multinomial extention of X_i. In π we also include $A_1, \ldots, A_k$. The approach used is as follows. We add the variable x_{m+1} as an indicator random variable. We then construct an $m+1$-variate multinomial A of degree h in each of the first m variables and degree $k-1$ in the x_{m+1}, such that for $x_{m+1} = i$, A is identical to A_i. For uniformity, we can allow A to be of degree $h > k-1$ also in x_{m+1}. Hence A is an $(m+1)$-variate multinomial of total degree $(m+1)h$. Note that if A_i is $E(X_i)$, then we have that A is really the encoding that we would get for the concatenation of all X_i.

Assume that the prover gives us tables $A_1, \ldots, A_k$ and A. How do we check that they are consistent? First, we use auxiliary tables $B_1, \ldots, B_k$ and B to check that each one of the tables is close to some polynomial of low total degree. Denote these (unknown) closest polynomials by $p_1, \ldots, p_k$ and p. Now we need to check that p_i and p agree. For this we pick a random point $x \in \mathcal{F}^m$. We query $A_i(x)$ and compare it to $A(x, i)$. However, as (x, i) is not a random point in $\mathcal{F}^{m+1}$ (because i is not random), we do not know that with high probability $A(x, i) = p(x, i)$. To remedy this problem, we probe $A(x, i)$ using the error detecting probe of Section 1.4.

Repeating the test for every i we have that with high probability, the restriction of p on $x_{m+1} = i$ is identical to p_i.

More careful inspection shows that there is no need for the auxiliary tables $B_1, \ldots, B_k$ and the test that $A_1, \ldots, A_k$ are close to being of low total degree. It suffices to perform a total degree test on A, and then the comparison between A and the A_i implies that also the A_is are close to multinomials of low total degree.

1.6 Parallelization using curves

Suppose that we have a table A that supposedly represents the value of a total degree d multinomial p over $\mathcal{F}^m$. We wish to obtain the values of $A(x_i)$ for points $x_1, \ldots, x_l$. Naively, this takes l probes to A. We present here a procedure that regardless of the value of l, uses only only four probes (provided that $|\mathcal{F}|$ is large enough). This procedure uses two additional tables, B and C. The table B supposedly gives for every line in $\mathcal{F}^m$ the univariate polynomial of degree d that describes A on this line. The table

C does the same for curves of degree $l-1$, which is a generalization of lines (lines being curves of degree 1). Formally, a degree k curve in $\mathcal{F}^m$ can be expressed as $c(t) = a_0 + a_1 t + \ldots + a_k t^k$, where for every $0 \leq i \leq k$, $a_i \in \mathcal{F}^m$. For every $t \in \mathcal{F}$, $c(t)$ gives a point in $\mathcal{F}^m$. Given arbitrary points $y_i \in \mathcal{F}^m$ for $1 \leq i \leq l$, there is a unique degree $l-1$ curve $c(t)$ for which $c(i) = y_i$. It can be constructed coordinate by coordinate, where in coordinate j the values of the coefficients $a_0, \ldots a_{l-1}$ are chosen so that the univariate polynomial definied by them agrees with the values of the jth coordinate of the y_i.

The value of the total degree d multinomial p over the points of a curve $c(t)$ of degree k is a degree dk polynomial in the parameter t. (This is true for each monomial separately, and hence for the sum of monomials.)

Parallel probe procedure. To read the value of A at l points $x_1, \ldots, x_l$, the verifier picks the degree $l-1$ curve $c(t)$ that passes through these points for $t = 1, \ldots, l$. Then he probes the table C for the polynomial $q(t)$ of degree $d(l-1)$ that supposedly describes A on $c(t)$. Then he picks a random $t_r \in \mathcal{F} \setminus \{1, \ldots, l\}$, and performs an *error detecting probe* for $A(c(t_r))$, and rejects if it differs from $q(t_r)$. (Recall that the error detecting probe probes A at the point $c(t_r)$ and on one more random point, and probes B on the line joining these two points. Hence altogether the parallel probe procedure make four probes. In [1] only three probes are used for this procedure, because the corresponding error detecting probe uses only two probes.)

Lemma 4 *Let $\Delta(A, p) = \epsilon$, where p is the total degree d multinomial closest to A. If $\epsilon = 0$ and B and C truly correspond to p then the parallel probe procedure correctly computes the value of A at the l query locations, and never rejects. If for at least one of the query locations, $A(x_i) \neq p(x_i)$, then the parallel probe procedure rejects with probability at least*

$$1 - \frac{d(l-1)}{|\mathcal{F}| - l} - \epsilon - \frac{d}{|\mathcal{F}| - 1}$$

The proof of this lemma is given as homework.

1.7 A PCP with a constant number of probes

In Lecture 3 we constructed a PCP showing $NP \subset PCP(\log n, (\log n)^{O(1)})$. We now explain how to modify it so that the verifier makes only a constant number of probes to the PCP witness, and in each probe reads a character that is represented by $(\log n)^{O(1)}$ bits.

Recall that the satisfying assignment was encoded in a table A using its degree-h extention. On A the verifier performed the degree-h test. We replace this degree-h test by a total degree $d = hm$ test. For this the prover needs to prepare an additional table B that contains for every line in $\mathcal{F}^m$ the degree d univariate polynomial that describes A on this line. Given the tables

A and B, the total degree d test of Section 1.3 uses only two probes to detect that A is far from being of total degree d. By repeating the total degree test independently (or in a pairwise independent way) a constant number of times, the verifier deduces with high probability that $\Delta_d(A)$ is very small (e.g., at most 1/100).

In Lecture 3, the sum check part of the protocol required m queries to the PCP witness, where for each query the reply is a degree $2h$ polynomial in one variable. Such a polynomial can be represented by $O(h)$ coefficients, each containing $\log|\mathcal{F}|$ bits. Hence altogether a polynomial can be viewed as a character in an alphabet, where each character can be represented by $O(\log n \log\log n)$ bits. Recall that the number of random bits that the verifier is using altogether is $O(\log n)$, so that there are at most $N = n^{O(1)}$ possible queries that the verifier can make.

To reduce the number of probes in the sum check protocol we do as follows. Let T be a table with all answers to all possible queries of the verifier. T contains N entries. Choose integers $h' = \Theta(\log n)$ and $m' = \Theta(\log n/\log\log n)$ such that $(h')^{m'} = N$. The prover prepares a table A' that represents the degree h' extension of T over $(\mathcal{F}')^{m'}$, where $|\mathcal{F}'|$ is large enough. In addition the prover prepares a table B' that gives for every line in $(\mathcal{F}')^{m'}$ the degree $d' = (h')m'$ polynomial that describes A' on the line. Finally, the prover prepares a table C' that for every curve of degree $m-1$, gives the degree $(m-1)d'$ polynomial that describes A' on this curve.

Using A' and B', the verifier performs a total degree d' test on A'. This takes only a constant number of probes. Having convinced himself that A' is indeed very close to a total degree d' multinomial, the verifier performs the sumcheck protocol using only four probes (to A', B' and C'). Note that the sum check protocol needs to read m characters from A'. Simply read them using the parallel probe procedure of Section 1.6.

We have successfully reduced the number of probes to a constant. We still need to check that the cost in terms of other parameters is not prohibitingly large. This requires a more careful inspection of the table C'. Naively, there are $(|\mathcal{F}'|^{m'})^m$ ways of choosing m points in A', which is $N^{\Theta(m)}$. As N is polynomial in N and $m = \Theta(\log n/\log\log n)$, this makes the size of the table C' superpolynomial. The way around this problem is to observe that the original verifier required only $O(\log n)$ random bits. Hence there are at most $n^{O(1)}$ choices for m entries in A' that he may ever wish to access. In other words, from the whole table C', only $n^{O(1)}$ entries have nonzero probability of being accessed. The prover never needs to write down any other entry in C', thus keeping the size of C' polynomial.

Other things that we need to check is that it suffices to have $|\mathcal{F}|$ and $|cF'|$ of size $O((\log n)^{O(1)})$ for the analysis of the total degree test to work, which is indeed shown in [1]. This is necessary in order to keep the number of random bits used by the verifier $O(\log n)$. Finally, we need to also check that the verifier works in a way that allows future composition. Namely, that before probing the PCP witness, it can prepare a list of (a constant number of) locations to probe and a circuit (or CNF formula) of size $O((\log n)^{O(1)})$ that will be evaluated on the answers to the probes. Thereafter, all the verifier does is perform the probes, substitute the answers in the circuit, and evaluate it. Indeed, this is the case here, because the original PCP was of this form, and the extra computational effort needed by the new verifier is only polynomial time computations (such as evaluating polynomials on certain points) on inputs of polylogarithmic size.

1.8 The inner verifier

We now treat the verifier of Section 1.7 as an outer verifier, and construct for it an inner verifier based on the principles of Lecture 5. In many respects, this inner verifier is similar to the outer verifier. We just highlight the differences here.

Let k (a constant) denote the number of probes made to the PCP witness by the outer verifier, and let $a = O((\log n)^{O(1)})$ denote the number of bits that are used in order to represent a single character read by the outer verifier. Each character X_i is replaced by the inner PCP prover by a table A_i that supposedly gives its degree-h' extension over $(\mathcal{F}')^{m'}$, where $h' = O(\log\log n)$, $|\mathcal{F}'| = O((\log\log n)^{O(1)})$, and $m' = O(\log\log n/\log\log\log n)$. In addition, for every random string r of the outer verifier, the inner prover prepares an inner PCP witness π_r, that in particular includes a table A_r that represents the degree h' extension of the concatination of $X_{r,1}, \ldots, X_{r,k}$, where $X_{r,i}$ denotes the answer given by the outer prover to the ith probe of the outer verifier, when the outer verifier is using random string r.

The combined outer and inner verifier works as follows. First, as an outer verifier, it uses a random string r of $O(\log n)$ bits to decide on k probe locations of the outer PCP, and on a circuit C of size $O((\log n)^{O(1)})$ that will be used in order to evaluate the answers to the probes. Then it switches role and becomes an inner verifier. The input to the inner verifier is k tables $A_{r,1}, \ldots, A_{r,k}$ and an inner PCP π_r that in particular includes a table A_r. The inner verifier uses a random string r' of length $O(\log\log n)$ to decide on p probe locations and on a circuit $C_{r,r'}$ of size $O((\log\log n)^{O(1)})$ that will be used in order to evaluate the answers to the probes. The p probe locations are of two types:

1. Probes to the inner PCP proof π_r to verify satisfiability of $C_{r,r'}$, in a way similar to Section 1.7.

2. Consistency checks between A_r and $A_{r,1}, \ldots, A_{r,k}$, as explained in Section 1.5.

As we need only a constant number of probes for each type of probes, the number of probes remains a constant.

In conclusion we obtain that

$$NP \subset RPCP(r = O(\log n), p, c = O((\log\log n)^{O(1)}), a = O((\log\log n)^{O(1)}), e)$$

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy. "Proof verification and hardness of approximation problems". *JACM 45(3): 501–555 (1998).*

[2] S. Arora, S. Safra. "Probabilistic checking of proofs: a new characterization of NP". *JACM 45(1): 70–122 (1998).*

[3] U. Feige, L. Lovasz. "Two-prover one-round proof systems: their power and their problems". *Procedings of the 24th Annual ACM Symposium on the Theory of Computing*, 733–744, 1992.

[4] D. Lapidot, A. Shamir. "Fully parallelized multi-prover protocols for NEXP-time". *JCSS* 54(2), 215–220, 1997.

[5] A. Polishchuk, D. Spielman. "Nearly-linear Size Holographic Proofs". *Proceedings of 26th ACM Symposium on Theory of Computing*, 194–203, 1994.

[6] R. Rubinfeld, M. Sudan. "Robust characterizations of polynomials with applications to program testing". *SIAM journal on Computing*, 25(2):252–271, 1996.