6.841 Advanced Complexity Theory April 9, 2007

Lecture 16

Lecturer: Madhu Sudan *Scribe: Amanda Redlich, Shubhangi Saraf*

1 Overview

In today's lecture we will cover the following

- Permanent: worst case $\leq$ average case
- Permanent $\subseteq$ IP
- towards IP $\supseteq$ PSPACE

2 The Permanent

We begin be recalling the definition of the permanent of a matrix. It is a function of n^2 integers.

$$\text{Permanent}\left((x_{ij})_{i,j=1}^{n}\right) = \sum_{\Pi \in S_n} \prod_{i=1}^{n} x_{i\Pi(i)}$$

We observe that the permanent of an $n \times n$ matrix is a degree n polynomial in n^2 variables. It is a "low-degree multivariate polynomial," and this turns out to be a very interesting feature of the permanent, as we'll see in the rest of the lecture.

3 Random Self-Reduction for Low-Degree Polynomials

This problem was first studied by Beaver–Feigenbaum and Lipton. The basic question is as follows: We are given a polynomial $f(x_1, \ldots, x_m)$ of degree d over some finite field $\mathbb{F} = \mathbb{Z}_p$ (say). Given an algorithm that computes f on random instances, can we compute $f(a_1, \ldots, a_m)$ for any worst-case $a_1, a_2, \ldots, a_m$?

There are two ways of understanding what it means to have an algorithm that computes f on random instances. When each input x_i; for $1 \leq i \leq m$ is chosen independently and uniformly

from $\mathbb{Z}_p$, either

- for most instances, say for $\left(1 - \frac{1}{n^2}\right)$ fraction, the algorithm outputs the right answer, and for $\frac{1}{n^2}$ fraction it gives the wrong answer, or

- the algorithm runs in expected polynomial time for the given distribution, and always outputs the right answer.

The first notion is weaker than the second, since we know that we can easily convert an algorithm of the second kind to the first kind. We'll assume the weaker notion when we talk about an algorithm that computes f on random instances.

4 Worst Case to Average Case Reduction for the Low Degree Polynomials

Let f be an m-variate polynomial, with degree $\leq d$. Then f has $\binom{d+m}{m} = \binom{d+m}{d}$ coefficients. Since this number is exponentially large, it is not tractable to write down all the terms of the polynomial explicitly. To get around this problem, we employ the "dimension-reduction" trick.

Let's say that we have a poly-time algorithm A that evaluates f correctly on most inputs. Assume that for $x \in \mathbb{Z}_p^m, f(x) = A(x)$, unless $x \in B$, where $\frac{|B|}{p^m} \leq \delta = \frac{1}{n^2}$.

The goal is to have an algorithm that computes f at a point $\bar{a} = (a_1, \ldots, a_m)$, with high probability. The idea for doing this is as follows: The algorithm picks a random line through $\bar{a}$, by picking a random point $\bar{b}$ uniformly from $\mathbb{Z}_p^m$, and drawing the line through $\bar{a}$ and $\bar{b}$. It them samples the line at $d+1$ points, and if it is lucky each time, i.e., at each of those points, A equals f, then those values uniquely determine f on the entire line. We can interpolate to find the function explicitly on the entire line, and thus evaluate the function at $\bar{a}$.

We analyze the above scheme. Let $\bar{a} = (a_1, a_2, \ldots, a_m)$; and $\bar{b} = (b_1, b_2, \ldots, b_m)$. Call the line $l_{\bar{a},\bar{b}} = \{l_{\bar{a},\bar{b}}(t) := (a_1 + tb_1, a_2 + tb_2, \ldots, a_m + tb_m) | t \in \mathbb{Z}_p\}$.

When we restrict f to the line, we get $f_{\bar{a},\bar{b}}(t) = f(\bar{a} + t\bar{b})$ is a polynomial in one variable, and of degree $\leq d$.

We observe that for any fixed $\bar{a}$ and non-zero t, $\bar{a}+t\bar{b}$ is distributed uniformly in $\mathbb{Z}_p^m$ if $\bar{b}$ is distributed uniformly in $\mathbb{Z}_p^m$. Hence,

$$Pr_{\bar{b}}[f(\bar{a} + t\bar{b}) \neq A(\bar{a} + t\bar{b})] \leq \delta.$$

By the union bound,

$$Pr_{\mathbf{b}}[\exists t \in \{1, 2, \ldots, d+1\} | f(\bar{a} + t\bar{b}) \neq A(\bar{a} + t\bar{b})] \leq (d+1)\delta.$$

Assuming the above event does not occur, if we compute a polynomial $h(t)$ of degree $\leq d$, such that $h(t) = A(\bar{a} + t\bar{b})$ for every $t \in \{1, 2, \ldots, d+1\}$. Then, $\forall t$, $h(t) = f_{\mathbf{a},\bar{b}}(t)$. Once we have this, we can compute $h(0)$, which gives us the value of f at $\bar{a}$.

In this manner, using a subroutine algorithm that computes the permanent on most points, we can get an algorithm that evaluates the permanent on worst case inputs with high probability.

5 Restriction of f to a Curve

Say we are interested in the value of the function f at 5 points. Just as in the case of 1 point, where we drew a line, we draw a curve in the space through those 5 points, and through a randomly chosen sixth point.

We know that a line $l_{\bar{a},\bar{b}} : \mathbb{F} \to \mathbb{F}^m$ is a concatenation of m functions - $l^{(1)}_{\bar{a},\bar{b}}, l^{(2)}_{\bar{a},\bar{b}}, \ldots, l^{(m)}_{\bar{a},\bar{b}} : \mathbb{F} \to \mathbb{F}$, each of which is a univariate polynomial of degree 1.

Similarly, a curve $C : \mathbb{F} \to \mathbb{F}^m$ is a concatenation of m functions - $C^{(1)}, C^{(2)}, \ldots, C^{(m)} : \mathbb{F} \to \mathbb{F}$, each of which is a univariate polynomial, and degree$(C) = \max_{1 \le i \le m}\{\text{degree}C^{(i)}\}$.

Given $k+1$ points, $\exists$ a degree k curve through them.

If $f : \mathbb{F}^m \to \mathbb{F}$ is of degree d, then $f|_c : \mathbb{F} \to \mathbb{F}$ is of degree $\le kd$, where $f|_c(t) = f(c(t))$.

We now put these ideas together to give an interactive proof for the permanent.

6 Permanent $\subseteq$ IP

We can use the same general type of idea to calculate the permanent. Let the permanent language L_{perm} be defined as follows.

Definition 1 *$L_{perm} = \{(M,a) | a = perm(M)\}$, where M is a matrix in $\mathbb{Z}_p^{n\times n}$ for some n and p prime.*

We want to show this language is in **IP**. That is, we have to describe an algorithm by which the prover can convince the verifier a pair (M,a) is indeed in L_{perm}. Let M_i be the i^{th} minor of M. Then we know that

$$perm(M) = \sum_{i=1}^{n} a_{1i} perm(M_i)$$

So instead of giving an answer for the $n \times n$ permanent of M, we can check all n of the $(n-1)\times(n-1)$ permanents of the minors. Just like before, the prover can compute a curve C going through $M_1 \ldots M_n$ in $\mathbb{Z}_p^{(n-1)\times(n-1)}$. The curve can be described using $(n-1)^2$ functions in one variable (one for each coordinate), each of which has degree n.

So, the prover sends to the verifier C, which is a function from $\mathbb{Z}_p$ to $\mathbb{Z}_p^{(n-1)\times(n-1)}$, and P, which is a function from $\mathbb{Z}_p$ to Z_p. The prover claims that C is in fact a curve through M_i and P is the permanent restricted to that curve, i.e. $P(i) = perm(M_i)$. Now the verifier has to check that this is true. The verifier can easily check if $C(i) = M_i$ (since that depends on $(n-1)^2$ polynomials, each of degree n) and if $a = \sum_{i=1}^{n} a_{1i} p(i)$. If either of these is false, the verifier rejects.

Let's see if this works. If the prover is honest and $perm(M) = a$, then the verifier won't reject. Suppose $perm(M) \ne a$. Then either the verifier will reject, or the prover will have to lie. The prover will have to give a P that is not equal to the permanent on C.

Suppose that happens. Certainly there does exist some P' that is equal to the permanent on C. If $P \neq P'$, then they differ almost everywhere (being degree n polynomials). So the verifier can pick a random $t_0 \in \mathbb{Z}_p$ and ask for proof that $(C(t_0), P(t_0))$ is in L_{perm}. This is the same type of problem as we started with, except that now it's about a matrix of size $(n-1)^2$ instead of size n^2. Iterating this process n times, eventually we reach a question about a 1×1 matrix's permanent, which is easy to calculate.

Question from the audience: We're assuming p is really big. What if it's smaller?

Well, if $p = 2$, then the permanent equals the determinant (since $-1 = 1$), and it'd not difficult to calculate. If $p = 3$, then it's essentially the same as $PARITY - SAT$. Also, in general, for small p we can use an extension field of characteristic p in the above algorithm.

Soundness analysis: If $(M, a) \notin L_{perm}$, then

$$Pr_{t_0}[(C(t_0), P(t_0) \in L_{perm}] \leq n^2/p$$

. That's just for one iteration. Since there are n iterations, the union bound gives

$$Pr[acceptance] \leq n^3/p$$

Choosing a p big enough makes that probability as small as you like.

7 Towards IP $\subseteq$ PSPACE

Here's a very abstract notion of a problem for which we want to prove things:

Definition 2 *Let n, d, p prime, and w be fixed. Suppose we have $p_1 \dots p_n$ such that, for all i, $p_i(x_1 \dots x_n)$ is a multivariate polynomial of n variables with degree $\leq d$. This is a polynomial computation sequence if, for all $i \geq 1$, p_i can be computed at any point by making $\leq w$ oracle calls to p_{i-1}, and if p_0 can be computed efficiently.*

Then the algorithm we described above for computing the permanent can also be used to prove interactively that $p_n(a_1 \dots a_n) = b$. All you'd have to do is ask the prover to give p_i restricted to C, where C is a curve through all w points needed for oracle calls (P_i is like the permanent of an $i \times i$ matrix and the w points are like the minors).

And guess what? $PSPACE$ can be described by rules like this. Suppose we're thinking of a space n computation. Let $F_i(a_1 \dots a_s, b_1 \dots b_s)$, where $\bar{a}$ and $\bar{b}$ are configurations, be 1 if it is possible to reach configuration $\bar{b}$ within 2^i steps, beginning with configuration $\bar{a}$, and 0 otherwise.

Since we're dealing with a space bounded computation, there are at most 2^s steps in any computation before it begins looping. So to check if a particular string is accepted by this computation, we need only calculate $F_i(\bar{a}, \bar{b})$, setting

$\bar{a}$ to be the starting configuration given that string, $\bar{b}$ to be the accepting configuration (WLOG we may assume there is exactly one accepting configuration), and $i = s$.

Furthermore, we note that $F_i(\bar{a}, \bar{b}) = 1$ if and only if there exists $\bar{c}$ such that $F_{i-1}(\bar{a}, \bar{c}) = F_{i-1}(\bar{c}, \bar{b}) = 1$. So in fact

Fact 3 $F_i(\bar{a}, \bar{b}) = \sum_{\bar{c} \in \{0,1\}^s} F_{i-1}(\bar{a}, \bar{c}) \cdot F_{i-1}(\bar{c}, \bar{b})$

It would be great if these were all low-degree multivariate polynomials. And they are! Note that the above recursion implies that for each a_j (b_j), the degree of a_j (b_j) in F_i is the degree of a_j (b_j) in F_{i-1}. Since the degree is bounded for $i = 0$, it is bounded in general.

We're almost all set. We now have a bunch of low-degree polynomials, just like before. But we're summing over all exponentially many possible $\bar{c}$. This could be a problem. In the next lecture, we'll see what do.