INFR11102: Computational Complexity 26/11/2019

Lecture 20: Interactive protocol for the permanent

Lecturer: Heng Guo

1 Interactive protocol for the permanent

Recall that the permanent of a n-by-n matrix A is defined as:

$$\operatorname{per}(A) := \sum_{\pi \in S_n} \prod_{i=1}^{n} A_{i,\pi(i)}.$$

Computing the permanent is a #P-complete problem, even if A is a $\{0,1\}$ matrix. Moreover, Toda's theorem [Tod91] states that $\mathtt{PH} \subseteq \mathtt{P}^{\#\mathtt{P}}$.

Recall the class IP, which is the private coin interactive proof protocol for polynomially many rounds. It feels like "merely" an interactive extension of NP. However, Lund, Fontnow, Karloff, and Nisan [LFKN92] showed a surprising result.

Theorem 1. $\operatorname{per}(A)$ *can be computed by an interactive proof protocol in polynomial rounds.*

Here we present an interactive proof system for the permanent over the ring of integers. The input is an n-by-n $\{0,1\}$-matrix $A = (a_{ij})_{1\le i,j\le n}$.[1] The prover will present an integer N such that the verifier accepts N with high probability if and only if $\operatorname{per}(A) = N$.

Instead of working over the integer ring, we will choose a finite field $\mathrm{GF}(p)$ where $p > n! \ge \operatorname{per}(A)$ is a prime. Thus working in $\mathrm{GF}(p)$ does not lose any information, and the advantage is that every element now has a multiplicative inverse. At the start of the protocol, the prover will choose this prime p and send it to the verifier, and the verifier then use the randomized primality test [Mil76, Rab80] (or the deterministic one [AKS04]) to make sure that p is a prime. The rest computation is all in modulo p.

We utilize the self-reducibility of the permanent. In other words, first observe that

$$\operatorname{per}(A) = \sum_{i=1}^{n} a_{1i} \operatorname{per}(A_{1,i}),$$

where $A_{1,i}$ is the $(n-1)$-by-$(n-1)$ submatrix of A obtained by removing the first row and the ith column of A. Thus, to compute $\operatorname{per}(A)$, we only need to compute the permanent for $(n-1)$-by-$(n-1)$ matrices.

[1] The protocol still works for arbitrary integers without much modification, but we will stick to the $\{0,1\}$ case for simplicity.

The protocol resolves around the following polynomial. Define

$$F(x) := \sum_{i=1}^{n} \frac{\prod_{1\le j\le n,\, j\neq i}(x-j)}{\prod_{1\le j\le n,\, j\neq i}(i-j)} A_{1,i}.$$

The coefficients of this polynomial matrices $(A_{1,i})$. It has an interesting property, namely that $F(i) = A_{1,i}$ for all $1 \le i \le n$. To see this, if $x = i$, then only the ith term evaluates to $A_{1,i}$, and all others evaluate to 0. Indeed, the construction of $F(x)$ is the matrix analogue of Lagrange interpolation. Instead of viewing $F(x)$ as a polynomial with matrix coefficients, we will view it as a $(n-1)$-by-$(n-1)$ matrix, where each entry is a univariate polynomial of degree $\le n-1$.

Using this view, define

$$g(x) := \mathrm{per}(F(x)).$$

and define $B_r := F(r)$ for $1 \le r \le p$. Notice that $B_r = A_{1,r}$ if $1 \le r \le n$. By definition, $g(r) = \mathrm{per}(B_r)$. Moreover, each single entry of $F(x)$ (which is a matrix) is a univariate polynomial of degree $\le (n-1)$. Thus, $g(x)$ is a polynomial in x of degree $\le (n-1)^2$.

After agreeing with the prime p, The prover's move is to send $\widehat{g}(x)$ and N to the verifier, claiming that $\widehat{g}(x) = g(x)$ and $N = \mathrm{per}(A)$. Of course the prover might be lying, and thus we need to verify it. We first do a simple consistency check:

$$N = \sum_{i=1}^{n} a_{1i} \cdot \widehat{g}(i).$$

If $N \neq \mathrm{per}(A)$, then the prover cannot send the true $g(x)$. Thus, in that case, the prover must lie and send $\widehat{g} \neq g$. We now assume that the consistent check has passed.

The next step is crucial, to build the protocol by induction. The verifier randomly choose $r \in [p]$ and evaluate $\widehat{g}(r)$ and B_r. Notice that $F(x)$ can be easily computed, and so is $B_r = F(r)$.

Now we have a $(n-1)$-by-$(n-1)$ matrix B_r and a claimed value of $\widehat{g}(r) = \mathrm{per}(B_r)$ to verify. We continue by induction.

What is the probability of the verification to succeed? There are two cases.

1. If $\widehat{g} = g$, then of course $\widehat{g}(r) = \mathrm{per}(B_r)$, and $N = \mathrm{per}(A)$ is verified.

2. If $\widehat{g} \neq g$, then since $g(x)$ has degree at most $(n-1)^2$,

$$\Pr_{r\in[p]}[\widehat{g}(r) = g(r)] \le \frac{(n-1)^2}{p} = \exp(-O(n)),$$

since $p \ge n!$. If the prover were to lie successfully, then he has to pass all of the checks above, each of which happens with exponentially small probability. (Otherwise the recursion goes down to a 1-by-1 matrix whose permanent is easy to compute.) The induction goes down n steps, and thus the total probability for the prover to lie successfully is still exponentially small.

To summarize, if the claimed value $N = \mathrm{per}(A)$, then the prover can convince the verifier with probability 1, by simply answering correctly at every step. If the claimed value $N \neq \mathrm{per}(A)$, then $\widehat{g}$ must be wrong, and thus the prover must lie for the next smaller matrix, and in the end he has only exponentially small probability to make up n consistent lies.

2 The power of interactive proofs

Theorem 1 implies that IP is more powerful than the whole PH. Indeed, shortly after Theorem 1 was found, Shamir [Sha92] settled the question regarding the power of IP.

Theorem 2. IP = PSpace.

The proof of Theorem 2 is an IP protocol for TQBF. The protocol is somewhat similar to the proof of Theorem 1, except that we need to first translate a totally quantified Boolean formula into a polynomial. This can be done via operations like replacing $x \wedge y$ with xy and $x \vee y$ with $x + y - xy$. Details can be found in [AB09, Section 8.3.3].

One more comment on the difference between IP and AM. Goldwasser and Sipser [GS86] have shown the following result.

Theorem 3. *Let k be a number computable in polynomial-time.* $\mathtt{IP}[k] \subseteq \mathtt{AM}[k+2] \subseteq \mathtt{IP}[k+2]$.

Thus, we see that $\mathtt{IP}[k] = \mathtt{AM}[2] = \mathtt{AM}$ for any fixed k (as $\mathtt{AM}[k] = \mathtt{AM}[2]$), and $\mathtt{IP} = \mathtt{IP}[poly] = \mathtt{AM}[poly]$. The number of rounds has a more essential role in the power of interactive proof systems than the difference between private vs. public coins. The proof of Theorem 3 is very similar to the AM protocol for GNI. Namely, we find a suitable set S and distinguish between Yes/No via the size of S.

3 Random self-reducibility of the permanent

An interesting aspect of the permanent is its random self-reducibility, shown by Dick Lipton [Lip89]. It implies that computing the permanent can be reduced to computing a random instance, and the idea to prove it is very similar to the LFKN protocol. (In fact, LFKN were inspired by Lipton's result.)

Let $\mathbb{F}$ be a finite field of size $> 4n$.

Theorem 4. *If there is an algorithm which computes the permanent of $1 - \frac{1}{4n}$ fraction of all n-by-n matrices over $\mathbb{F}$, then there is one which computes all matrices correctly with high probability.*

Proof. Let A be the input matrix. Pick a random matrix R from $\mathbb{F}^{n \times n}$, and let $B(x) = A + x \cdot R$. Thus, $\mathrm{per}(B(x))$ is a polynomial in x of degree at most n. For any fixed r, $B(r)$ is a random matrix, and thus the oracle computes $\mathrm{per}(B(r))$ correctly with probability at least $1 - 1/4n$.

Fix $n+1$ values $r_1, \cdots, r_{n+1} \in \mathbb{F}$. We can evaluate $\mathrm{per}(B(x))$ at $x = r_1, \cdots, r_{n+1}$ using the oracle. The probability such that one of them is wrong is $\frac{1}{4n}$. Thus, by a union bound, we compute all of $\mathrm{per}(B(r_i))$ correctly for all $i \in [n+1]$ with probability at least $1 - \frac{n+1}{4n} \geq \frac{2}{3}$ (when n is at least some constant). When this happens we can interpolate the polynomial of $\mathrm{per}(B(x))$ since its degree is at most n. Indeed, it is easy to see that

$$\sum_{i=1}^{n+1} \frac{\prod_{1 \leq j \leq n+1,\, j \neq i} (x - r_i)}{\prod_{1 \leq j \leq n+1,\, j \neq i} (r_i - r_j)} \mathrm{per}(B(r_i))$$

agrees with $\mathrm{per}(B(x))$ on $n+1$ points. Since the degree of $\mathrm{per}(B(x))$ is at most n, they must be identical. What we want to compute is simply $\mathrm{per}(A) = \mathrm{per}(B(0))$.

The above yields an algorithm that succeeds with probability at least $2/3$. To drive the error probability to arbitrarily low, we may run the algorithm multiple times (by taking more than one R) and take the most common one. (This is, once again, an application of the Chernoff bound.) □

Remark (Historical). Nisan first sent out an email about an multi-prover interactive protocol (denoted `MIP`) for the permanent in Nov 1989. Within a couple of weeks, Lund, Fontnow, and Karloff found the protocol for the permanent (Theorem 1). The published version [LFKN92] combined both results. After these two emails, Shamir [Sha92] showed Theorem 2 within another couple of weeks. Eventually, Theorem 2 was "scaled down" from `IP` to `NP`, which is now known as the PCP (Probabilistically Checkable Proof) theorem. It states that any `NP` language has a polynomial sized proof that can be checked by only $O(1)$ queries.

Remark (Bibliographic). Relevant chapters are [AB09, Chapter 8.6, 8.7].

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Ann. of Math. (2)*, 160(2):781–793, 2004.

[GS86] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *STOC*, pages 59–68, 1986.

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

[Lip89] Richard J. Lipton. New directions in testing. In *Distributed Computing And Cryptography*, volume 2 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 191–202. DIMACS/AMS, 1989.

[Mil76] Gary L. Miller. Riemann's hypothesis and tests for primality. *J. Comput. Syst. Sci.*, 13(3):300–317, 1976.

[Rab80] Michael O. Rabin. Probabilistic algorithm for testing primality. *J. Number Theory*, 12(1):128 – 138, 1980.

[Sha92] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.

[Tod91] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.