03684155: On the P vs. BPP problem. 4/12/16 – Lecture 7

No non-uniform lower bounds implies PIT is hard and no derandomization

Amnon Ta-Shma and Dean Doron

Kabanets and Impagliazzo showed that derandomizing polynomial identity-testing (henceforth PIT) is essentially equivalent to proving non-uniform super-polynomial lower bounds on a uniform function. Specifically, we will prove that if PIT $\in \mathsf{P}$ then either $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$ or PERM $\notin \mathsf{nuANC}^2$, where nuANC^2 is the arithmetic version of non-uniform NC and will be defined soon.

We begin by some preliminaries.

1 Preliminaries

1.1 Arithmetic circuits

We consider arithmetic circuits. In this model, inputs are variables or constants from a field and the computation is performed using $+$ and $\times$. Formally:

Definition 1. *An arithmetic circuit C over the field $\mathbb{F}$ and the set of variables $X = x_1, \ldots, x_n$ is a directed acyclic graph as follows. The vertices of C are called* gates. *Every gate in C of in-degree 0 is labeled by either a variable from X or a field element from $\mathbb{F}$ (those are the input gates). Every other gate in C is labeled by either $+$ or $\times$ and has in-degree 2. The gate of out-degree 0 is called the output gate. The size of C is the number of edges in C.*

An arithmetic circuit computes a polynomial in the natural way. Also, every polynomial in $\mathbb{F}[X]$ can be computed by an arithmetic circuit.

We define VP, which is the algebraic analogue of NC.

Definition 2. *A family of polynomials $\{f_n\}$ over $\mathbb{F}$ is p-bounded if there exists some polynomial t such that for every n, the number of variables of f_n is n, the total degree of f_n is at most $t(n)$ and there is an arithmetic circuit of size at most $t(n)$ computing f_n. The class $\mathsf{VP}_{\mathbb{F}}$ consists of all p-bounded families over $\mathbb{F}$.*

An important family of polynomials is the determinant DET_n over $n \times n$ matrices. It is an exercise to show that indeed $\{\mathrm{DET}_n\} \in \mathsf{VP}$. In fact, the determinant is in some sense *complete* for the class VP. More accurately, for any family $\{f_n\} \in \mathsf{VP}$ there exists a function $t : \mathbb{N} \to \mathbb{N}$ satisfying $t(n) = n^{O(\log n)}$ such that f_n is a projection of $\mathrm{DET}_{t(n)}$. For the exact definitions, see [4, Chapter 1].

In the *Boolean* world we can think of IntDet as the problem of computing the determinant of an integer matrix. It is known that IntDet $\in \mathsf{NC}^2$ (following Csansky's algorithm [2]) and also that many important problems in linear-algebra are NC^1 Turing reducible to IntDet (the interested reader is referred to [1]). The class DET comprise the languages that are NC^1 Turing reducible to IntDet.

We also denote $\mathsf{AP/poly}$ for the class of non-uniform arithmetic P, i.e., the class of all function over $\mathbb{F}$ that can be solved by a non-uniform family of polynomial-size arithmetic circuits. Another name that fits the class is nuAP.

1.2 Polynomial identity testing

We will consider multivariate polynomials over some integral domain, e.g., the ring $\mathbb{Z}$ of integers. We say that a polynomial is identically zero if all its coefficients are zero. There are two seemingly similar problems we can consider:

- The problem of testing whether a given polynomial vanishes over a given domain, and,
- The problem of testing if a given polynomial is identically zero.

In spite of the external resemblance between the two problem they are in fact quite different. We are concerned with the latter.

Definition 3. *The problem* PIT *is defined as follows. Given as an input an arithmetic circuit C computing a polynomial $p(x_1, \ldots, x_n)$ over a field $\mathbb{F}$ (or a ring R), decide if $p \equiv 0$.*

1.3 The permanent

Recall that the permanent of an $n \times n$ matrix A of integers is defined as

$$Perm(A) \ = \ \sum_{\sigma \in Sn} \prod_{i=1}^{n} A_{i,\sigma(i)},$$

where S_n is the set of all permutations of $[n]$. We define Perm as the problem of computing the permanent of a matrix over the integers. Perm is #P-complete under polynomial-time reductions (this is Valiant's theorem [5]).

The (seemingly) hardness of the permanent is exemplified in the algebraic world as well: The permanent polynomial is a complete problem for the VNP (w.r.t. *uniform* projections). Again, we remark that the notation VNP is misleading as it suggests that VNP is the arithmetic analogue of NP. Instead, it is the arithmetic version of #P as is witnessed by the fact that the permanent function is complete for it. It is conjectured that $\mathsf{VP} \neq \mathsf{VNP}$. The corresponding conjecture in the Boolean world would be that $\mathsf{DET} \neq \mathsf{P}^{\#\mathsf{P}}$, which is almost as saying $\mathsf{NC}^2 \neq \mathsf{P}^{\#\mathsf{P}}$.

2 Testing an arithmetic circuit for the permanent

We prove:

Theorem 4. *If* Perm $\in \mathsf{AP/poly}$ *and* PIT $\in \mathsf{P}$ *then* $\mathsf{P}^{\mathrm{Perm}} \subseteq \mathsf{NP}$.

The proof idea is as follows. First, since we assume Perm $\in \mathsf{AP/poly}$ there exists a small (polynomial size) arithmetic circuit for the permanent. We can therefore guess such a circuit. If we could find a way to verify that our guess is correct, we could have used the circuit to simulate the oracle calls to the Perm function and we are done. For the verification step we use the downward self-reducibility of the permanent function together with our assumption that PIT $\in \mathsf{P}$.

Proof. Let $L \in \mathsf{P}^{\mathrm{Perm}}$ and let M be the TM with oracle calls to Perm that decides it in time $p(n)$ on inputs of length n, for some polynomial p. Also, let q be the polynomial for which there are circuits of size $q(n)$ that solve the permanent of an $n \times n$ matrix.

We describe a nondeterministic TM M' that decides L. M' on input x of length n:

1. For $i = 1$ to $p(n)$, guesses circuits $\{C_i\}$ with size at most $q(i) \leq q(p(n))$, where each C_i has input size i.
2. Validate that every C_i solves the permanent of $i \times i$ matrices, in a way that we soon show.
3. If so, simulate M on x, evaluating each oracle call with the appropriate circuit. Note that as the running time of M is at most $p(n)$, it cannot make longer queries to the oracle.
4. Answer according to M.

We now show how to do (2) by induction, and this will conclude our proof. Verifying C_1 is trivial. For $i \geq 2$, assume that C_{i-1} is correct, and note that for a matrix A of size $i \times i$,

$$Perm(A) \;=\; \sum_{k=1}^{i} A_{1,k} \cdot Perm([A]_{1,k})$$

where $[A]_{1,k}$ is the $(1,k)$-th minor of A. Thus, it is enough to verify the following polynomial identity:

$$C_i(A) - \sum_{k=1}^{i} A_{1,k} \cdot C_{i-1}([A]_{1,k}) \;\equiv\; 0.$$

Given the circuits C_i and C_{i-1} of size at most $q(i)$, there is a circuit of size $\mathrm{poly}(q(i))$ that computes the l.h.s. of the equivalence. Hence, we can use the polynomial-time algorithm for PIT to verify the equation. $\square$

3 PIT $\in \mathsf{P}$ implies a non-uniform lower bound

The highlight of our lecture is the following theorem:

Theorem 5 ([3]). *If* PIT $\in \mathsf{P}$ *then either* $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$ *or* Perm $\notin \mathsf{AP/poly}$.

Proof. Assume towards contradiction that the following three conditions hold:

- PIT $\in \mathsf{P}$.
- $\mathsf{NEXP} \subseteq \mathsf{P/poly}$.
- Perm $\in \mathsf{AP/poly}$.

As $\mathsf{NEXP} \subseteq \mathsf{P/poly}$,

$$\mathsf{NEXP} = \mathsf{MA} \subseteq \Sigma_2 \subseteq \mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{P}^{\mathrm{Perm}},$$

where we have used Lecture 6 – that $\mathsf{NEXP} \subseteq \mathsf{P/poly}$ implies $\mathsf{NEXP} = \mathsf{MA}$, Toda's theorem that $PH \subseteq \mathsf{P}^{\#\mathsf{P}}$ and Valiant's theorem that Perm is $\#\mathsf{P}$-complete.

As we assume Perm $\in \mathsf{AP/poly}$ and PIT $\in \mathsf{P}$, by Theorem 4 we have that $\mathsf{P}^{\mathrm{Perm}} \subseteq \mathsf{NP}$. Thus,

$$\mathsf{NEXP} \subseteq \mathsf{P}^{\mathrm{Perm}} \subseteq \mathsf{NP}.$$

However, this contradicts the non-deterministic time hierarchy. $\square$

References

[1] Stephen A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64(13), 1985. International Conference on Foundations of Computation Theory.

[2] L. Csansky. Fast parallel matrix inversion algorithms. *SIAM Journal of Computing*, 5(6):618–623, 1976.

[3] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[4] Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends® in Theoretical Computer Science*, 5(3–4):207–388, 2010.

[5] Leslie G Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979.