03684155: On the P vs. BPP problem. 27/11/16 – Lecture 5a

Non-uniformity - Some Background

Amnon Ta-Shma and Dean Doron

1 Derandomization vs. Pseudorandomness

In the last lecture we saw the following conditional result: If there exists a language in E (which is a uniform class) that is worst-case hard for $\mathsf{SIZE}(s(n))$ for some parameter $s(n)$, then BPP has some non-trivial derandomization, depending on $s(n)$.

We now want to distinguish between two notions:

- Derandomization.
- Pseudorandomness.

We say a language in BPP can be *derandomized* if there exits *some* algorithm placing it in P. For example, primality testing was known to be in BPP, but the best deterministic algorithm for it was quasi-polynomial, until Agrawal, Kayal and Saxena [1] showed a polynomial time algorithm for it. The AKS algorithm is a derandomization of a specific probabilistic polynomial time algorithm.

The conditional result we saw last time does much more than that. If we have a language in the uniform class E not in the non-uniform class $\mathsf{SIZE}(s(n))$ then a PRG exists. I.e., there is a way to systematically replace random bits with pseudorandom bits such that no BPP algorithm will (substantially) notice the difference. Thus, not only it shows that *all* languages in BPP are in fact in P (or whatever deterministic class we get) but it also does that in an oblivious way. There is one recipe that is always good for replacing random bits used by BPP algorithms! One recipe to rule them all.

Now we take a closer look at the assumption behind the conditional result. It says that if we have a uniform language hard for a non-uniform class we are good. The regression to non-uniform classes is non-aesthetic to begin with, and also thought-provoking. It is a qualitative issue rather than a quantitive issue (unlike the question of parameters, and, to a lesser extent, the question of worts-case vs. average-case hardness). Is it just an artifact of our limited abilities or should it really be there?

The first thing that comes to mind is that this is an artifact of the construction. We really used non-uniformity, but where? There are several places we fix bits (sometimes we call it hardwiring). For example in the NW generator, we "fix" all the bits in the advice (that depend on the function and the distinguisher, not on the input to f). There are also places in the STV reduction where we fix bits. Yet, one gets to wonder, because in many of these places one can pick a string at random instead of the hardwiring of the fixed string. Is there a way to avoid the fixing and get a uniform result?

A little more thought reveals that there is another, perhaps deeper, issue. Suppose L is in BPP solved by a machine $M(x,y)$ running in P, where $x \in \{0,1\}^n$ is the input and $y \in \{0,1\}^m$ is the random bits string. We want to replace y by $PRG(U_\ell)$ for a suitable pseudo-random generator

$PRG : \{0,1\}^{\ell} \to \{0,1\}^{m}$. Since we replace the random bits in an oblivious way (that does not depend on M and also does not depend on the input x), the machine that we want to cheat is not one but any of the machines in the collection $\{M(x,\cdot)\}$. In particular, we can think of x as a non-uniform advice string given to M. This is a reason (perhaps, *the* reason) why we need a PRG against non-uniform classes. The other uses of non-uniformity we have are probably just because it is more convenient to simply fix unnecessary bits, since we already have to cheat non-uniform classes, so who cares.

The interested ones among you are referred to the paper "Randomness vs. Time: De-randomization under a uniform assumption" by Impagliazzo and Wigderson [3]. They prove that if $\mathsf{BPP} \neq \mathsf{EXP}$, then every problem in BPP can be solved deterministically in sub-exponential time on almost every input (on every samplable ensemble for infinitely many input sizes). This is similar to what we have seen last time, but entirely in the uniform world. It is still a conditional statement that assumes $\mathsf{BPP} \neq \mathsf{EXP}$, but now the assumption is entirely in the uniform world and compares the power of a probabilistic class (BPP) against a deterministic one (EXP), which is a natural comparison:"Randomness vs. Time". It says that if BPP is not everything (because clearly $\mathsf{BPP} \subseteq \mathsf{EXP}$) then it can be non-trivially derandomized. However, here the derandomization is cumbersome: it is only on average! When one says on average he/she should also say with respect to what, and IW say it can be with respect to any samplable distribution (a distribution that can be sampled in P). Thus, we get a natural conditional result (or a "gap" theorem that says unconditionally that BPP is either everything or has non-trivial derandomization) but the derandomization is only on average and not worst-case. This is exactly because of the problem that we described before that the input may serve as a non-uniform advice to M.

So where do we stand? With a "uniform" assumption ($\mathsf{BPP} \neq \mathsf{EXP}$) we can non-trivially derandomize BPP but only on average. With a "non-uniform" assumption (E is not contained in $\mathsf{SIZE}(s(n))$ we can worst-case non-trivially derandomize BPP. We understand where the problem lies – the input may serve as a non-uniform advice. According to the picture that emerges,

- If EXP is not contained in $\mathsf{P/poly}$, there is some PRG and sone non-trivial oblivious derandomization, where random bits are replaced in a way oblivious to the input and algorithm.
- $\mathsf{EXP} \subseteq \mathsf{P/poly}$, in which case we do not have a uniform function hard for non-uniform circuits. We may still have specific derandomizations (e.g., of primality), we might even be able to derandomize the whole of BPP one algorithm at a time, but we do not expect to have a uniform PRG against non-uniform classes, because such a PRG would give rise to a uniform language hard for a non-uniform class. But, can it be that we can still get one PRG that fools all uniform machines in BPP worst-case?

The main goal of the coming lectures (that compose the second part of the course) is explaining the Impagliazzo-Kabanets result. It says that if a specific language, PIT (for polynomial identity testing) that is known to be in $\mathsf{coRP} \subseteq \mathsf{BPP}$, can be derandomized, then either:

- $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$, in which case there even exits a PRG against non-uniform classes, and a non-trivial derandomization of BPP, or,
- The permanent function does not have polynomial size arithmetic circuits. Valiant proved the permanent is #P-complete, so the permanent lies somewhere between PH and PSPACE. Still, it is a uniform function, an so if this option happens we have a uniform polynomial hard for *non-uniform* arithmetic circuits!

One consequence of this amazing result is that if we can derandomize PIT (and certainly if we can derandomize the whole of BPP, even on a language-to-language basis) we get a uniform function that is hard for a non-uniform model!

This is quite amazing. The discussion above explains why we should have suspected that non-uniformity has something to do with pseudorandomness, i.e., with the dream (or fantasy) that derandomization can be oblivious to the language and input. But now what we see is that not only pseudorandomness inherently implies a non-uniform lower bound on a uniform language, but also derandomization is, and even derandomization of a specific function, PIT.

Still, the Impagliazzo-Kabanets has its own little caveats, and mostly that if PIT can be derandomized then either we get a lower bound in the boolean world (of circuits) *or* in the arithmetic world of arithmetic circuits.

In what follows we will slowly work our way towards this and other related results. We begin with a short reminder of the non-uniform world, followed by Karp-Lipton type theorems that state that if a uniform class is contained in some non-uniform class, then this implies a collapse in the uniform world, e.g., $\mathsf{EXP} \subseteq \mathsf{P/poly}$ implies $\mathsf{EXP} = \mathsf{MA}$. We will also present the Impagliazzo-Kabanets-Wigderson result that $\mathsf{NEXP} \subseteq \mathsf{P/poly}$ implies $\mathsf{NEXP} = \mathsf{MA}$. We will then prove the IK results and possibly some more surprising results.

2 Recap: some results and notions from the complexity class

2.1 Advice, MA, IP = PSPACE

Definition 1 (Turing machines that take advice)**.** *A language $L \in \mathsf{DTIME}(t(n))/a(n)$ if there exists a Turing machine M and a family of strings $\{\alpha_n\}$ such that $|\alpha_n| \leq a(n)$ and on input x of length n, M runs in time $t(n)$ and satisfies $M(x, \alpha_{|x|}) = 1$ iff $x \in L$.*

It is not hard to show that $\mathsf{P/poly} = \bigcup_{a,b \in \mathbb{N}} \mathsf{DTIME}(n^a)/n^b$. We will also need the following lemma.

Lemma 2. *If $\mathsf{NEXP} \subseteq \mathsf{P/poly}$ then for every $a \in \mathbb{N}$ there exists $b = b(a) \in \mathbb{N}$ such that $\mathsf{NTIME}(2^{n^a})/n \subseteq \mathsf{SIZE}(n^b)$.*

Proof. As an exercise. □

We will use the notion of interactive protocols and the class IP. Those of you who are not familiar with it, please refer to Chapter 8 of [2]. We will also use the well-known fact that $\mathsf{PSPACE} = \mathsf{IP}$ ([5, 6], and also in Chapter 8 of [2]). An important case of one-round protocol is the class MA:

Definition 3. *A language $L \in \mathsf{MA}$ if there exists a polynomial-time Turing machine M and polynomials p and q such that for every input x of length n,*

- *If $x \in L$, $\exists z \in \{0,1\}^{q(n)} \,.\, \Pr_{y \in \{0,1\}^{p(n)}}[M(x,y,z) = 1] \geq \frac{2}{3}$.*
- *If $x \notin L$, $\forall z \in \{0,1\}^{q(n)} \,.\, \Pr_{y \in \{0,1\}^{p(n)}}[M(x,y,z) = 1] \leq \frac{1}{3}$.*

2.2 $\mathsf{NP} \subseteq \mathsf{P/poly}$ implies $\mathsf{PH} = \Sigma_2$

This is a classical result by Karp and Lipton.

Theorem 4 ([4]). *If* $\mathsf{NP} \subseteq \mathsf{P/poly}$ *then* $\mathsf{PH} = \Sigma_2$.

Proof. For the polynomial hierarchy to collapse, it is enough to show that $\Pi_2 \subseteq \Sigma_2$. Consider the Π_2-complete language Π_2SAT consisting of all true formulas of the form

$$\forall u \in \{0,1\}^n \exists v \in \{0,1\}^n \varphi(u,v) = 1 \tag{1}$$

where φ is some unquantified formula.

If $\mathsf{NP} \subseteq \mathsf{P/poly}$ there is a polynomial p and a family of circuits $\{C_n\}$, C_n of size $p(n)$, such that for every boolean formula φ, $C_n(\varphi) = 1$ iff $\phi \in SAT$, i.e., iff there exists $v \in \{0,1\}^n$ such that $\varphi(v) = 1$.

Furthermore, using search to decision reduction, we actually know that there is a polynomial q and a family of circuits $\{C'_n\}$, each of size $q(n)$, such that for every boolean formula φ, if there is a string v such that $\varphi(u,v) = 1$, C'_n *outputs* it.

By letting $r(n) = \text{poly}(q(n))$ be the encoding size of C'_n, we obtain that if indeed formula (1) holds, then:

$$\exists w \in \{0,1\}^{r(n)} \forall u \in \{0,1\}^n \varphi(u, ENC(w)(\varphi(u,\cdot)) = 1. \tag{2}$$

where $ENC(w)$ is the circuit encoded by the string w. Also, if (2) is true, then in particular for every u there exists a witness v such that $\varphi(u,v) = 1$ and so (1) is true. As circuit evaluation can be done in polynomial time, evaluating the truth of formula (2) can be done in Σ_2, so $\Pi_2\text{SAT} \in \Sigma_2$ and $\Pi_2 \subseteq \Sigma_2$, as desired. □

2.3 $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$ implies $\mathsf{PSPACE} = \mathsf{MA}$

Proof. $\mathsf{MA} \subseteq \mathsf{PSPACE}$ is trivial. For $\mathsf{PSPACE} \subseteq \mathsf{MA}$, let $L \in \mathsf{PSPACE}$, so L has an interactive proof protocol, and moreover, the proof shows there exists an interactive protocol where the honest prover runs in PSPACE (for soundness, of course, we are still protected from any all-powerful malicious prover). Since we assume $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$,the honest prover can be replaced by a family of circuits $\{C_n\}$, each of size $\text{poly}(n)$.

The interaction between the prover and the verifier can now be carried in one round. Given input x of length n, a honest prover sends to the verifier C_n which is of size $\text{poly}(n)$. The verifier then simulates the interactive proof, getting answers from C_n instead of the prover. By the correctness proof of the interactive protocol, if $x \in L$ the verifier always accepts. If $x \notin L$, then no matter what the malicious prover sends, by the interactive proof property we know that no prover (no matter what strategy he uses or what circuit he sends) will be able to convince the verifier except for some negligible probability. Thus, $L \in \mathsf{MA}$. □

Similarly, we leave the following as an exercise:

Theorem 5. *If* $\mathsf{EXP} \subseteq \mathsf{P/poly}$ *then* $\mathsf{EXP} = \mathsf{MA}$.

The polynomial hierarchy and the permanent

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. The permanent, at first sight, seems to be very similar to the determinant. However, it is conjectured to be much harder to compute. Recall that #P is the problem of exactly counting the number of witnesses to an NP relation. Valiant proved:

Theorem 6 ([7])**.** Perm *is* #P*-complete under polynomial-time reductions.*

In fact, even computing the permanent of integer matrices with 0 or 1 entries is already #P-complete.

Toda proved:

Theorem 7. $\mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}}$.

References

[1] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of mathematics*, pages 781–793, 2004.

[2] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[3] Russell Impagliazzo and Avi Wigderson. Randomness vs. time: De-randomization under a uniform assumption. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 734–743. IEEE, 1998.

[4] Richard M Karp and Richard J Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 302–309. ACM, 1980.

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

[6] Adi Shamir. Ip= pspace. *Journal of the ACM (JACM)*, 39(4):869–877, 1992.

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