U.C. Berkeley
CS294: Pseudorandomness and Combinatorial Constructions
Professor Luca Trevisan

Handout N16
October 20, 2005
Scribe: Anand Kulkarni

Notes for Lecture 16

1 Proof of the Impagliazzo-Widgerson Theorem

1.1 Recap

We are near the end of our proof of the Impagliazzo-Widgerson result that every exponential time problem has a polynomial-time transformation to an equivalent exponential-time problem which is hard on average. To recap from last time:

Suppose L is a language decidable in time $2^{O(n)}$. Let L_n be the characteristic function of L on inputs of length n, $L_n : \{0,1\}^n \to \{0,1\}$. Choose an arbitrarily small constant $\gamma > 0$, a field $\mathbb{F}$ such that $|\mathbb{F}| = 2^{c\gamma n}$, and a set $H \subseteq \mathbb{F}$ with $H = 2^{\gamma n}$.

We can define the polynomial $p : \mathbb{F}^m \to \mathbb{F}$ with degree$(p) \leq |H|m = 2^{\gamma n + O(1)}$, $m = \frac{1}{\gamma}$, that agrees with L_n on $H^m \approx \{0,1\}^n$.

There exist absolute constants c and c' such that if c is a circuit of size $\leq 2^{\gamma n}$ that computes p, ie,

$$\mathbf{Pr}_x[c(x) = p(x)] \geq \frac{1}{2^{\gamma n}} \tag{1}$$

then there is also a circuit for L_n of size $2^{c'\gamma n}$.

Then, p is computable in time $2^{O(n)}$.

So far we have reached this point: suppose L is decidable in time $2^{O(n)}$ and not solvable by circuits of size $\leq 2^{\delta n}, \delta > 0$. Suppose we do this entire construction and pick $\gamma = \frac{\delta}{c'}$ and $2^{c'\gamma n} \leq 2^{\delta n}$. Then we can construct a Boolean function that has exponential average case complexity – that is, the probability of success is exponentially small.

So we are in the right ballpark for our result, but we want a single-bit Boolean function that is very hard on average, instead of this n-bit Boolean function. This is where we begin today.

1.2 Remainder of proof

Definition 1 (Inner product on strings) *For strings a and b, $\langle a, b\rangle = \sum_i a_i b_i \mod 2$.*

Claim 2 *Define a function $f : \{0,1\}^{cn+c\gamma n} \to \{0,1\}$ as $f(x,y) = (p(x), y)$. Suppose there is a circuit of size $\leq S$ that computes f with $\mathbf{Pr}[c(z) = f(z)] \geq \frac{1}{2} + \epsilon$. The existence of this circuit contradicts the hardness of L.*

Proof:

$$\mathop{\mathbf{Pr}}_{x,y}[c(x,y) = f(x,y)] \geq \frac{1}{2} + \epsilon \Rightarrow \mathop{\mathbf{Pr}}_{x}\left[\mathop{\mathbf{Pr}}_{y}[c(x,y)) = f(x,y)] \geq \frac{1}{2} + \frac{\epsilon}{2}\right] \geq \frac{\epsilon}{2} \tag{2}$$

Call this latter condition "x is good". Now, fix some "good" x. We have that $\mathbf{Pr}_y[c(x,y) = \langle p(x), y\rangle] \geq \frac{1}{2} + \frac{\epsilon}{2}$. But this means that the output of c has some nontrivial agreement with the linear function mapping $y \to \langle p(x), y\rangle$.

Recall that there is a limit to the number of linear functions that a given function can closely agree with. Specifically, our function can have agreement $\geq \frac{1}{2} + \frac{\epsilon}{2}$ with at most $\frac{1}{\epsilon^2}$ linear functions. We could now use the Goldreich-Levin algorithm to enumerate these linear functions, but there's no need. Because the number of functions is so small, we can simply use brute force to enumerate them in subexponential time. Given c and x, we can find in time $2^{c\gamma x}\text{size}(c) = 2^{\gamma n}S$ all linear functions $a \in \{0,1\}^{c\gamma n}$ such that the function mapping $y \to c(x,y)$ has agreement $\geq \frac{1}{2} + \frac{\epsilon}{2}$ with the function mapping $y \to \langle a, y\rangle$. If x is good, then the correct value of $p(x)$ will appear somewhere in our list. Here is an algorithm for the enumeration.

A (x)
  find list of a such that $y \to c(x,y)$ agrees $\geq \frac{1}{2} + \frac{\epsilon}{2}$ with $y \to \langle a, x\rangle$
  output random element of list

The probability of this algorithm succeeding is $\mathbf{Pr}_x[A(x) = p(x)] \geq \frac{\epsilon^2}{2} > \frac{1}{2^{\gamma n}}$. This holds even if we replace the random choice in the algorithm with an optimal fixed choice. Thus, the algorithm A can be implemented deterministically by a circuit of size $\leq S2^{c\gamma n}$.

Finally, if we apply the decoding algorithm for polynomials to A, we can construct a circuit of size $S \cdot 2^{c\gamma n} \cdot 2^{c''\gamma n}$ that computes p everywhere, and thus computes n everywhere, where c'' is some constant polynomial in $|\mathbb{F}|$.

Now choose $\epsilon = \frac{1}{2^{\Omega(n)}}$ and we can easily violate the hardness of L_n, giving a contradiction and resolving our claim. This concludes the proof of the Impagliazzo-Widgerson theorem. □

1.3 Summary of results

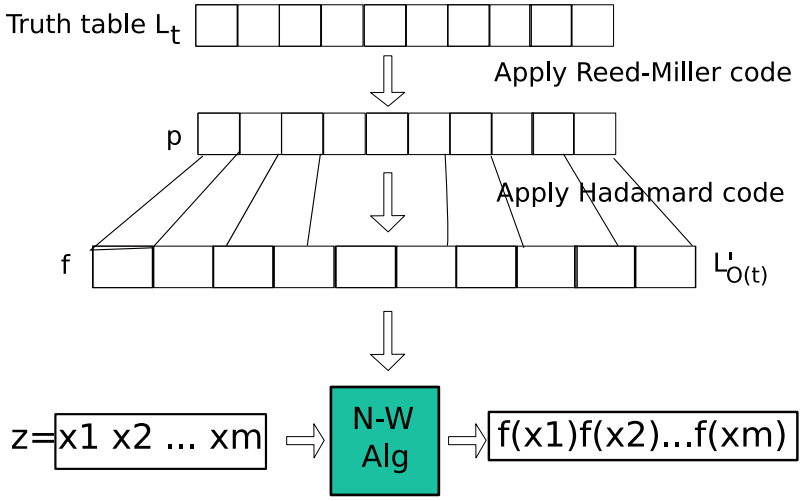
Theorem 3 *(Impagliazzo-Widgerson) If there is a language L and a constant $\delta > 0$ such that L is decidable in $\mathbb{R}^{O(n)}$ time and L_n is not solvable by circuits of size $\leq 2^{\delta n}$, then there exists a language L' and a constant $\alpha > 0$ such that L' is decidable in $2^{O(n)}$ time, and $\forall C$ of size $\leq 2^{\alpha n}$, $\mathbf{Pr}[C(x) = L'_n] \leq \frac{1}{2} + \frac{1}{2^{\alpha n}}$.*

Theorem 4 *(Nisan-Widgerson) If there exists a language L' and a constant $\alpha > 0$ such that L' is decidable in $2^{O(n)}$ time, and $\forall C$ of size $\leq 2^{\alpha n}$, $\mathbf{Pr}[C(x) = L'_n] \leq \frac{1}{2} + \frac{1}{2^{\alpha n}}$, then there is a generator*

$$G : \{0,1\}^{O(\log n)} \to \{0,1\}^n \tag{3}$$

that is $(n^2, \frac{1}{n})$-pseudorandom, and computable in polynomial time in n. Moreover, $\mathbf{P} = \mathbf{BPP}$.

The figure on the next page illustrates the use of such a language in the Nisan-Widgerson algorithm to build a strong pseudorandom generator. Let L be a language that is hard in the worst-case. For example, we could set the input to be a Turing Machine $\langle M\rangle$ and a string x, and let L be the set



of all such pairs where M accepts x in fewer than $2^{|x|}$ steps. We first define the truth table L_t for inputs of length t. Then we apply a Reed-Miller code to generate a polynomial p with $2^{O(t)}$ field elements. We apply a Hadamard code to each field element to get a function f equivalent to a language $L'_{O(t)}$. Then this will be the hard problem we use in the Nisan-Widgerson algorithm to make a generator. The seed to the generator will be $O(t) = O(\log n)$ evaluation points for f and the output will be the value of f at these points, yielding n bits.

1.4 Comments

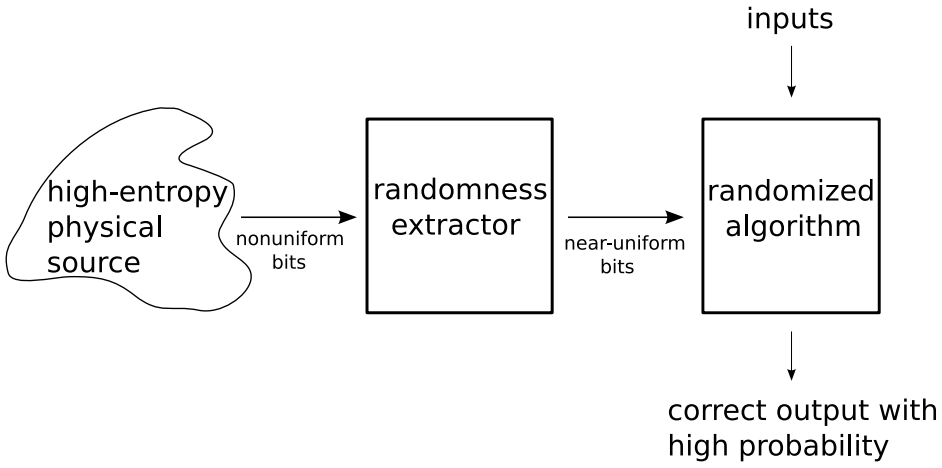
1. The one thing currently unknown in this system is the initial premise that there exists a hard language L

2. This is the best generator we can hope to get because the length of the seed is logarithmic in the nubmer of pseudorandom bits needed. In particular, this will be ϵ-biased against linear distinguishers.

3. We will go on to show how the Nisam-Widgerson machinery can be used in the construction of a randomness extractor.

2 Randomness Extractors

Suppose we have a physical source of bits which has high entropy but which is not uniformly distributed. We would like a procedure that takes as input a sample from such a distribution and returns a nearly uniform distribution of bits as output. We can then use these "random" bits wherever truly random bits are needed.



Example 1 *Von Neumann Randomness Extractor*

Let our source of randomness be a sequence of completely independent bits $x_1, ..., x_n$ *such that*

$$x_i = \begin{cases} 0 & \textit{with probability } p \\ 1 & \textit{with probability 1-p} \end{cases}$$

where p *is unknown.*

Then, for each subsequent pair of bits (x_{2n}, x_{2n+1}) *in the output string, replace* $(0,1)$ *by* 0*,* $(1,0)$ *by* 1*, and delete* $(0,0)$ *and* $(1,1)$*. An easy exercise shows that in the resulting distribution,* $P(0) = P(1)$*. On average, this turns* n *independent bits into* $2p(1-p)n$ *uniform bits.*

The shortcoming here is that truly independent bits as we assumed in the example are difficult to come by. There are impossibility theorems about extracting uniform distributions from inter-dependent bits – we cannot have a deterministic object which observes a nonrandom source and produces truly random output.

Let's suppose intead that our randomness extractor gets some small number t of truly random bits for its own operation and observes n bits with entropy $\geq k$, and outputs m near-uniform bits. This will be interesting if $t \ll m$, and very interesting if $t = O(\log(n))$.

Here are some definitions we will need to build the extractor next time:

Definition 5 (Statistical Distance) *If* X *and* Y *are distributions, then the statistical distance between* X *and* Y *is defined as*

$$\|X - Y\| := \max_{T:\{0,1\}^m \to \{0,1\}} \|\mathbf{Pr}[T(x) = 1] - \mathbf{Pr}[T(y) = 1]\| \tag{4}$$

where T *is an arbitrary statistical test and* $\mathbf{Pr}[T(x) = 1]$ *is the probability that* x *passes a given statistical test.*

Definition 6 (ϵ-close to uniform) *If* U_m *is the uniform distribution over* m *bits, then* X *is* ϵ*-close to uniform if*

$$\|X - U_m\| \leq \epsilon \tag{5}$$