Russell Impagliazzo — February 5

Derandomizing Computation in Time/Space

How many random bits are needed for randomized algorithms.

Algorithm $A(x,r)$ - input x, random r.

Derandomization: estimate $\text{Prob}_r\, A(x,r)$

Generic derandomization: find a list $r_1, \ldots, r_k$; try all $A(x,r_i)$

Given truly random $i_1, \ldots, i_b \in \{0,1\}$, (a b-bit "seed")

$G(i_1 \ldots i_b) = r_i$ is a pseudorandom generator.

Defn $G: \{0,1\}^b \to \{0,1\}^T$ is an ε-prg, for algorithms in class C

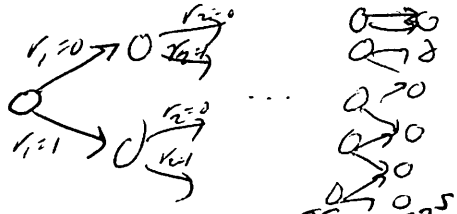
if $\forall x, \forall A \in C$,

$$\left| \text{Prob}_{r \in \{0,1\}^T}[A(x,r)] - \text{Prob}_{i \in \{0,1\}^b}[A(x, G(i))] \right| \leq \varepsilon$$

Class C for time/space tradeoff

Read-once, oblivious branching program, aka OBDD.

For each fixed input x: $A(x,-)$ defines an OBDD;



Space S branching program has $\leq 2^S$ many nodes (configurations) at each layer

Time T " " has T layers, last layer has two nodes "accept" and "reject".

Space = log (width)

Time = # of random bits.

Now work with class C = OBDD's with Time T and Space S.

= (S,T)-OBDD.

An ε-prg for C gives an ε-prg for $TISP(T,S)$.

Goal: Construct $\mathcal{E}$ prg with small b, for (S,T)-OBDD's.

Want b small, since derandomized algorithm uses time $T = 2^b T$

Also want G to be computable in small space S'.

Nisan-Zuckerman — this talk.

Ajtai-Komlos-Szemeredi, Babai-Nisan-Szegedy } earlier work

Nisan

In (S,T)-OBDD: Let N = # of random bits so far (= layer)

When $N >> S$, each state (at layer N) is reached in about $2^N/2^S$ ways ($\frac{2^N}{2^S} \simeq 2^N$ if $N >> S$.)

Given N bit string, and current state

Entropy of state is S

Entropy of random bits is N

Intuition: Use the unused entropy is $\simeq N-S$.

Alternate intuition: Use "poor quality" randomness to create only (uniformly) random bits.

Extractor: will use a small amount of true randomness

$$E: \{0,1\}^N \times \{0,1\}^\ell \longrightarrow \{0,1\}^M$$

N - "flawed" randomness
ℓ - true randomness
M - extracted randomness

Defn E is ε $(k, \varepsilon, N, \ell, M)$ extractor iff $\forall\, X \subseteq \{0,1\}^N$, $|X| \geq 2^k$, the statistical difference is less than ε:

$$E(x,s) \sim_\varepsilon U_M$$

$x \in_u X$, $s \in_u \{0,1\}^\ell$

where statistical distance equals L_1-distance or total variation. (or $\frac{1}{2}$ times it?)

Claim: $\exists$ construction $(k, \varepsilon, N, \ell, M)$-extractors
provided $M = \Omega(k)$, $\ell = O(\log N + \log(1/\varepsilon))$

Strong extractor: $(E(x,s), s) \sim_\varepsilon U_m \times U_\ell$

Claim: Same claim holds for strong extractor.

Let $N = c \cdot S$ $(N = O(S), c$ constant$)$

$k = N - 2S$

$\varepsilon = 1/\text{poly}(T)$

$M = \Omega(S)$

$\ell = O(\log S + \log T)$

We'll first use $N + \frac{T}{N}\ell$ bits to simulate T random bits.
and apply this construction recursively.

Results in PRG has seed length $\ell' = O\left(S \cdot \frac{\log T}{\log S}\right)$

Top level of recursion:

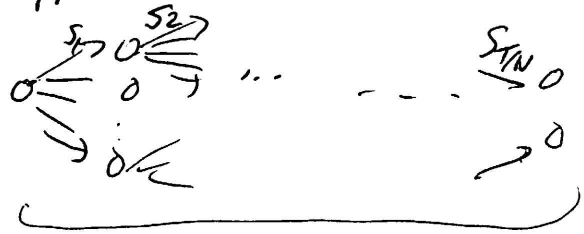
$$G_1(\underbrace{R}_{N\text{-bits}}, \underbrace{s_1, \ldots, s_{T/M}}_{\ell \text{ bits each}})$$

Random bits: $R \circ \underbrace{E(R, s_1)}_{M \text{ bits}} \circ \underbrace{E(R, s_2)}_{M \text{ bits}} \cdots \circ \underbrace{E(R, s_{T/M})}_{M \text{ bits}}$

(to make the analysis easier, don't use R as randomness)

Intuition: results of pseudorandom bits are close in distribution to truly random T-bits.

Recursive application



~~Use the~~ $\frac{T}{M}\ell$ bits of randomness

Handle this recursively.

Split into N/ℓ steps

Recurse $h = \frac{\lg T}{\lg(M/\ell)}$ times. Random bit string $R_1, \ldots R_h$

Last level uses $R_1 \ldots R_h \; S_1^{(h)} \ldots S_{M/\ell}^{(h)}$

Traverse recursion tree to extract bits at level $\emptyset$,

Space to compute: (height of tree) M,

Total random bits $= O(Nh) = O(S \frac{\lg T}{\lg S})$.

Russell - lecture #2, February 12

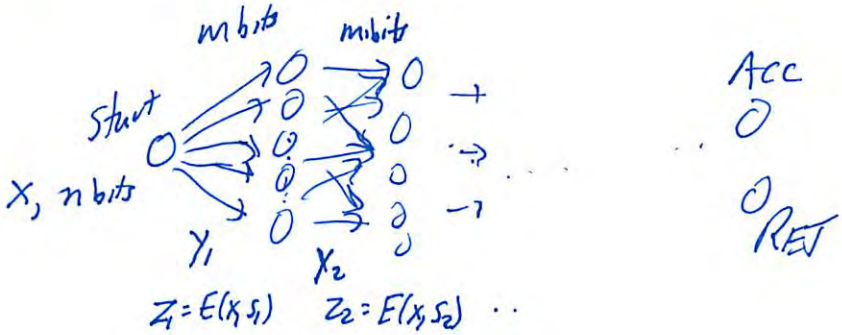
Extractor

Get randomness out of a partially random into an approximately uniformly random string.

~~If~~ $E(x,s): \{0,1\}^n \times \{0,1\}^r \to \{0,1\}^m$

"If x has k bits of randomness" and $s \in_u \{0,1\}^r$,

then $E(x,s) \sim_u \{0,1\}^m$



$Z_1 = E(x,s_1)$ $Z_2 = E(x,s_2)$ ..

Y_1, Y_2 .. are random bits that would be used in the actually (uniformly probabilistic) computation.

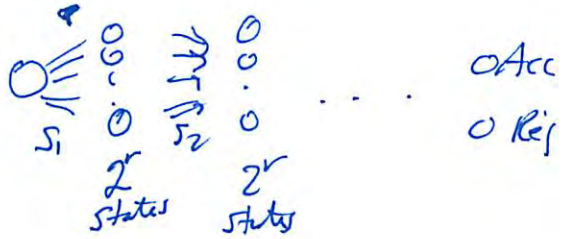
$Z_1, Z_2, \ldots$ replace the y_i's $Z_i' = E(x,s_i)$

$s_1, s_2, \ldots$ randomly chosen seeds, each r bits.

New randomness $= n + \left(\frac{T}{m}\right) r$

New space $= S$ (S is also the original space).

Apply recursively:



Reblock + apply recursively, in a "tree" of randomness.

Lemma If $E(x,s)$ is an $(n, r, m, k, \varepsilon)$-extractor, then

$$\left| \text{Prob}[\text{original computation accepts}] - \text{Prob}[\text{the extractor based algorithm accepts}] \right|$$

$$\leq O(\varepsilon T + T \cdot 2^{-\Omega(n-s-k)})$$

$$= O((\varepsilon + 2^{-\Omega(n-s-k)}) T).$$

$$= O((\varepsilon + 2^{-\Omega(n-s-k)}) T).$$

Intuition: We need k bits of "randomness" at each stage.
x starts off with n bit of (true) randomness.
But after the i^{th} level, only have $n-s$ bits of randomness.
So need $k < n-s$.

Pf Hybrid argument

$D_0 = Y_1 \cdots Y_{T/m}$ — P_0 - acceptance probability

$D_{T/m} = x\, z_1 \cdots z_{T/m}$ — $P_{T/m}$ " "

$D_i' = x\, z_1 \cdots z_i\, y_{i+1} \cdots y_{T/m}$ — P_i " "
$\quad\quad s_1 \quad s_i$

Need to show $|P_i' - P_{i+1}| \leq O(\varepsilon + 2^{-\Omega(n-s-k)})$.

$D_{i+1} = x\, z_1 \cdots z_i\, (z_{i+1})\, y_{i+2} \cdots y_{T/m}$ - change in one place from D_i'.

Can fix $s_1 \cdots s_i$, and $y_{i+2} \cdots y_{T/m}$ to maximize the difference $|P_i' - P_{i+1}|$.

Let $F(x)$ be the state reached after 1^{st} i stages
(using the fixed $s_1 \cdots s_i$)

$F(x): \{0,1\}^n \to \{0,1\}^s$

Let q be any state where $|F^{-1}(q)| < 2^k$
(this is the "bad" case, since $x \in F^{-1}(q)$ does not have k bit of randomness)

For a "bad" g,

$$\text{Prob}_x\left[F(x)=g\right] < 2^k/2^n$$

$$\text{Prob}_x\left[\exists g \; |F^{-1}(g)| < 2^k \text{ and } F(x)=g\right) < 2^k \cdot 2^s/2^n = \frac{2^k}{2^n}\cdot 2^s = 2^{-(n-s-k)}.$$

Let $\delta_i := |P_i - P_{i+1}|$

So $\exists$ fixed g s.t. $|F^{-1}(g)| > 2^k$ and

$$\left|P_i|_g - P_{i+1}|_g\right| > \delta_i - 2^{-(n-s-k)}.$$

"Distinguishing probability" conditioned on reaching configuration g.

$T(y) :=$ 'if start at g after i, do we accept using random bits $Y_{i+1} = y$.

For P_i, pick y at random

For P_{i+1}, choose $x \in_U F^{-1}(g)$, and choose $z = E(x, s_{i+1})$.

Then $\left|\text{Prob}[T(y)] - \text{Prob}[T(z)]\right| \le \varepsilon$

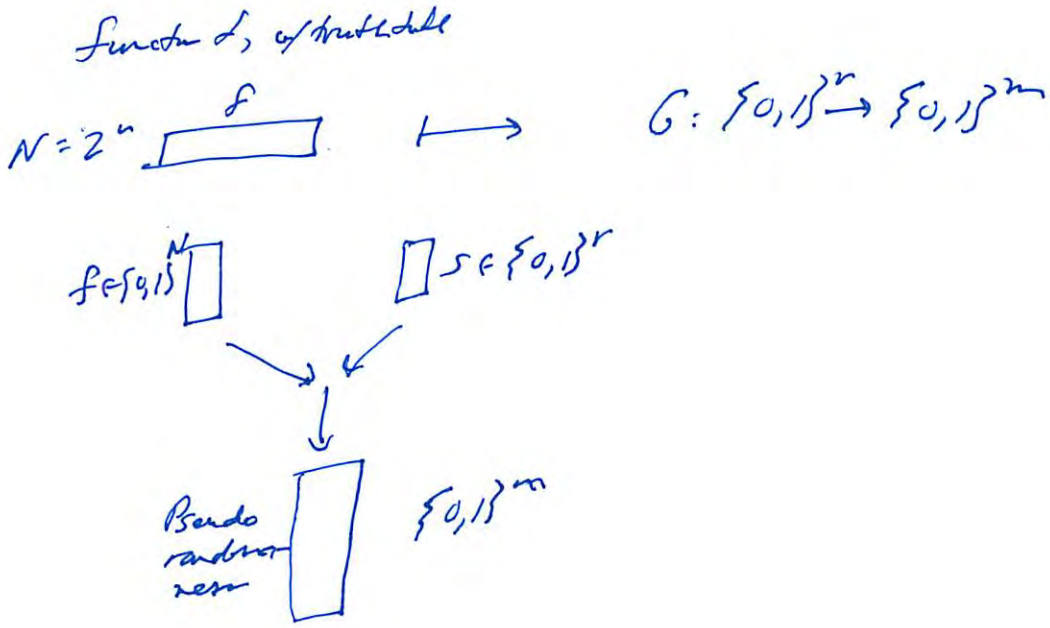
and $\delta_i - 2^{-(n-s-k)} \le \left|\text{Prob}[T(y)] - \text{Prob}[T(z)]\right|$

<u>gcd Lemma.</u>

<u>Rk</u> Big Ω not needed.

A pretty good extractor with a simple construction

[Trevisan]



If T is a test that distinguishes pseudo-random strings from truly random ones.

then $\exists$ circuit C^T that computes f, $|C^T| \le K$,

There are 2^K circuits of size k.

If f comes from a distribution $\omega > k$ bits of entropy, then for every test T, most f's are not equal to C^T, for each f.

If not, $T(G_f(r)) \sim_\varepsilon T(R)$.

Parameters with earlier argument

$n = N$

$K >>$ circuit size

$\varepsilon = \varepsilon$

$m =$ # number of bits per

Hardness to randomness construction

If f is worst-case hard, g is hard to predict w/ $> \frac{1}{2} + \delta$ advantage

[Basic idea: use error correcting codes of f.]

Using list-decoding codes from noise $\frac{1}{2} - \delta$.

Now use Nisan-Wigderson construction.

Design $S_1, \ldots, S_m$, $S_i \subseteq [\ell]$, $|S_i| = n'$.

.....

List decoding codes parameters

$$L = \Omega(1/\delta^2)$$

$|ECC(f)| = poly(N)$

$R = O(n')$

$\ell = \gamma \cdot n'$ - γ - constant

$m = 2^{\Omega(n')}$

$m = n^{O(1)}$.

$\ell = \gamma \cdot \log(N)$.

$k = 2^{\ell \cdot n} + \log(1/\varepsilon) = N^{\Omega(1)} + \log(1/\varepsilon)$

$n = k^{\Omega(1)}$.

$R = O(\log N)$, $\varepsilon = N^{-\gamma}$

$\varepsilon = 1/k$

$n = O(\max\{\log T, S\}) = O(S)$

$m = n^{\beta}$

$T \to T \cdot \frac{r}{m} = T \frac{\log S}{S^{\beta}}$

# of recursive iterations $= O\left(\frac{\log T}{\log S}\right)$

Total amount of randomness $= O\left(S \cdot \frac{\log T}{\log S}\right)$.