Pseudorandom generators (prg)

- Expanders helped in derandomizing a specific problem in RL.

- Prgs are objects to derandomize more general randomized algorithms.

Definition: • A distribution R over $\{0,1\}^m$ is (s, ε)-pseudorandom if $\forall$ circuits C of size $\leq s$,

$$\left| \Pr_{x \in R}[C(x)=1] - \Pr_{x \in U_m}[C(x)=1] \right| < \varepsilon.$$

(U_m ← uniform distribution)

(This measures how well can C distinguish R from U_m. It has cryptographic origins.)

• Let $S: \mathbb{N} \to \mathbb{N}$ be a function. A $2^{O(n)}$-time computable function $G: \{0,1\}^* \to \{0,1\}^*$ is an → S-prg if $\forall \ell \in \mathbb{N}$,

$G: \{0,1\}^\ell \to \{0,1\}^{S(\ell)}$ &

$G(U_\ell)$ is $(S(\ell)^3, 0.1)$-pseudorandom.

S is the stretch

Prgs derandomize classes

$S, \ell : \mathbb{N} \to \mathbb{N}$ are poly-time computable & nondecreasing.

Lemma: If there exists an S-prg then $\forall$ function ℓ, $\text{BPtime}(S \circ \ell(n)) \subseteq \text{Dtime}(2^{O(\ell(n))} \cdot S \circ \ell(n))$.

Proof:

- Idea is to use an S-prg G as the source of (pseudo-) random bits in the randomized algorithm.

- A language $L \in \text{BPtime}(S \circ \ell(n))$ if $\exists$ algorithm M that on input $x \in \{0,1\}^n$ uses $m = O(S \circ \ell(n))$ random bits r & runs for time $O(S \circ \ell(n))$ s.t.

$$\Pr_{r \in \{0,1\}^m} [\, M(x,r) = L(x) \,] \geq 3/4.$$

- The derandomization idea is to use an S-prg G to produce r's:
- On input x, our deterministic algorithm B will go over all $z \in \{0,1\}^{\ell(n)}$, compute $M(x, G(z))$ & output the majority vote.

- We claim that $\Pr_{z \in U_{\ell(n)}}[M(x, G(z)) = L(x)] \geq$

$\frac{3}{4} - 0.1 > 1/2$, thus, B correctly solves L.

- Suppose not, then $\Pr_{z \in U_{\ell(n)}}[M(x, G(z)) = L(x)]$

$< \frac{3}{4} - 0.1$.

$\Rightarrow \left| \Pr_{z \in U_{\ell(n)}}[M(x, G(z)) = L(x)] - \Pr_{r \in U_m}[M(x, r) = L(x)] \right|$

$> \left| \left(\frac{3}{4} - 0.1\right) - \left(\frac{3}{4}\right) \right| = 0.1$

- Now consider the circuit C_x that on input $y \in \{0,1\}^{s_0\ell(n)}$ outputs 1 iff $M(x, y) = L(x)$.
- Since M is $O(s_0\ell(n))$-time we get a (lazy) size bound of $O(s_0\ell(n))^2$ for C_x. (Exercise)

$\Rightarrow$ C_x distinguishes $G(U_{\ell(n)})$ from $U_{s_0\ell(n)}$ well, contradicting the fact that G is a S-prg.

$\Rightarrow$ B is a det. algo. solving L in $O(2^{\ell(n)} \cdot s_0\ell(n))$-time

□

- By picking various stretch fns S, we get the following <u>conditional</u> derandomizations:

<u>Corollary:</u> (i) If $\exists\ 2^{\varepsilon \ell}$-prg, for some $\varepsilon > 0$, then BPP = P.
↑ exponential stretch

(ii) If $\exists\ 2^{\ell^\varepsilon}$-prg, for some $\varepsilon > 0$, then
↑ subexponential stretch
$BPP \subseteq Dtime(2^{poly\text{-}lg(n)}) =: QuasiP.$

(iii) If $\forall c > 1$, $\exists\ \ell^c$-prg then $BPP \subseteq \bigcap_{\varepsilon > 0} Dtime(2^{n^\varepsilon}) =: Subexp.$
↑ polynomial stretch

<u>Proof:</u>

(i) Apply the lemma on $S: \mathbb{N} \to \mathbb{N}$; $n \mapsto 2^{\varepsilon n}$ & $\ell: \mathbb{N} \to \mathbb{N}$; $n \mapsto c \cdot \lg n$, for $c \in \mathbb{N}$.

(ii) On $S: \mathbb{N} \to \mathbb{N}$, $n \mapsto 2^{n^\varepsilon}$ & $\ell: \mathbb{N} \to \mathbb{N}$, $n \mapsto c \cdot (\lg n)^{1/\varepsilon}$, $c \in \mathbb{N}$.

(iii) On $S: \mathbb{N} \to \mathbb{N}$, $n \mapsto n^c$ & $\ell: \mathbb{N} \to \mathbb{N}$, $n \mapsto n^\varepsilon$ for $c \in \mathbb{N}$ & $\varepsilon \in (0,1)$.

□

- How do we construct these prgs?
The only known way is to exploit the <u>hardness</u> (conjectured) of problems!

<u>Hardness & prgs</u>

- We define two types of hardness of boolean functions.

<u>Definition</u>: • For $f: \{0,1\}^* \to \{0,1\}$, the <u>average-case hardness $H_{avg}(f)$</u> is the largest $S(n)$ s.t. $\forall$ circuit $C_n \in Size(S(n))$,

$$\Pr_{x \in U_n}[C(x) = f(x)] < 1/2 + 1/S(n).$$

• <u>Worst-case hardness $H_{wrs}(f)$</u> is the largest $S(n)$ s.t. $\forall$ circuit $C_n \in Size(S(n))$,

$$\Pr_{x \in U_n}[C(x) = f(x)] < 1.$$

▷ $H_{avg}(f) \leq H_{wrs}(f) < 2^{2n}$.

- Counting methods show that "usually" $H_{wrs}(f) = 2^{\Omega(n)}$.

  But we do not know of "natural" f with <u>super-polynomial</u> hardness!

- The conjectured f, of cryptographic significance, are:
  (1) $H_{wrs}(3SAT) = 2^{\Omega(n)}$?
  (2) $H_{avg}(Int\text{-}Fact) = n^{\omega(1)}$?

- We will later prove that a worst-case hard function gives rise to an average-case one.

  The tool would be local list decoding of linear error-correcting codes.

- For now, we relate average-case hardness to derandomization.