Hardness vs. Randomness

Nisan and Wigderson '89

Presented by Daniel Lee

19, April 2021

Table of Contents

Table of Contents

How powerful is Randomness?

How powerful is Randomness?

- Randomness is provably useful in alternative computation models such as Communication, Query, and Streaming

How powerful is Randomness?

- Randomness is provably useful in alternative computation models such as Communication, Query, and Streaming
- What about in the general TM computation?

How powerful is Randomness?

- Randomness is provably useful in alternative computation models such as Communication, Query, and Streaming
- What about in the general TM computation?
  1. Historically believed to be powerful; interpolates between deterministic and nondeterministic computation

How powerful is Randomness?

- Randomness is provably useful in alternative computation models such as Communication, Query, and Streaming
- What about in the general TM computation?
  1. Historically believed to be powerful; interpolates between deterministic and nondeterministic computation
  2. But many (often surprising) instances of derandomization

Examples

Examples

- EQ (communication complexity)
- PCPs and IPs

Examples

- EQ (communication complexity)
- PCPs and IPs
- Quicksort
- Primality testing
- Undirected ST-reachability

Examples

- EQ (communication complexity)
- PCPs and IPs
- Quicksort
- Primality testing
- Undirected ST-reachability
- Polynomial Identity Testing ??

Most likely not very powerful

Main Corollaries (Informal)

Under...

- *Extremely reasonable* (think $P \neq NP$) hardness assumptions, $BPP \subseteq SUBEXP$

Most likely not very powerful

Main Corollaries (Informal)

Under...

- *Extremely reasonable* (think $P \neq NP$) hardness assumptions, $BPP \subseteq SUBEXP$
- *Still very reasonable* hardness assumptiions, $BPP \subseteq QuasiP$

Most likely not very powerful

Main Corollaries (Informal)

Under...

- *Extremely reasonable* (think $P \neq NP$) hardness assumptions, $BPP \subseteq SUBEXP$
- *Still very reasonable* hardness assumptiions, $BPP \subseteq QuasiP$
- *Moderately reasonable* hardness assumptions, $BPP = P$

Most likely not very powerful

Main Corollaries (Informal)

Under...

- *Extremely reasonable* (think $P \neq NP$) hardness assumptions, $BPP \subseteq SUBEXP$
- *Still very reasonable* hardness assumptiions, $BPP \subseteq QuasiP$
- *Moderately reasonable* hardness assumptions, $BPP = P$

("reasonableness" to be defined, and up to individual interpretation)

Most likely not very powerful

Main Theorem

If there exists a function $f \in EXPTIME$ with hardness $H_f(\ell) \geq S(\ell^c)$ for some $c > 0$, then there exists some $d > 0$ s.t. for all time constructable bounds $t : \mathbb{N} \to \mathbb{N}$, $BPTIME(S(t^d)) \subseteq DTIME(2^{O(t^d)})$

Table of Contents

Pseudorandom Generator

(Informal) Definition

A pseudorandom generator G is a *quickly computable* function which takes a (short) random string and outputs a *longer string* which is *indistinguishable* from random for every *polynomial* size circuit family.

Pseudorandom Generator

(Informal) Definition

A pseudorandom generator G is a *quickly computable* function which takes a (short) random string and outputs a *longer string* which is *indistinguishable* from random for every *polynomial* size circuit family.

- *quickly computable:* $G(x)$ runs in 2^{ℓ} steps. (where $\ell = |x|$)

Pseudorandom Generator

(Informal) Definition

A pseudorandom generator G is a *quickly computable* function which takes a (short) random string and outputs a *longer string* which is *indistinguishable* from random for every *polynomial* size circuit family.

- *quickly computable:* $G(x)$ runs in 2^ℓ steps. (where $\ell = |x|$)
- *longer string:* Let $S : \mathbb{N} \to \mathbb{N}$ (polytime computable, monotone). $|G(x)| = S(\ell)$
  - The goal will be S (close to) exponential

Pseudorandom Generator

(Informal) Definition

A pseudorandom generator G is a *quickly computable* function which takes a (short) random string and outputs a *longer string* which is *indistinguishable* from random for every *polynomial* size circuit family.

- *quickly computable:* $G(x)$ runs in 2^ℓ steps. (where $\ell = |x|$)
- *longer string:* Let $S : \mathbb{N} \to \mathbb{N}$ (polytime computable, monotone). $|G(x)| = S(\ell)$
  - The goal will be S (close to) exponential
- *indistinguishable:* For every circuit family $C \lesssim S(\ell)^2$:

$$\Pr_{z \sim U_\ell}[C(G(z)) = 1] - \Pr_{x \sim U_{S(\ell)}}[C(x) = 1] < 0.1$$

PRG $\implies$ derandomization

Lemma 2.1

If there exists a quick pesudorandom generator $G : \ell \to S(\ell)$, then for any time constructible bound $t = t(n)$:

$$BPTIME(S(t)) \subseteq DTIME(2^{O(t)})$$

Ex: $S(\ell) = 2^{\varepsilon t}$ $\implies$ $BPTIME(2^{\varepsilon t}) \subseteq DTIME(2^{ct})$

$\varepsilon > 0$ union $t = a \log n$

$\implies BPP \subseteq P$

$S(\ell) = 2^{\ell^{\varepsilon}}$ $\implies$ by unioning over $\ell = a \log^{1/\varepsilon} t$

$\implies BPP \subseteq QuasiP$

PRG $\implies$ derandomization proof

Lemma 2.1

If there exists a quick pesudorandom generator $G : \ell \to S(\ell)$, then for any time constructible bound $t = t(n)$:

$$BPTIME(S(t)) \subseteq DTIME(2^{O(t)})$$

$M \in BPTIME(S(t))$ accepts L

– M can use at most $S(t)$ rand.

Define $N'(w, p \in \{0,1\}^t) := M(w, G(p))$

Claim: $w \in L$

$$\Pr_{p \sim U_t}[N'(w, p) = 1] \geq \frac{2}{3} - \frac{1}{10} > 1/2$$

proof

↳ Assume otherwise, Fix $w \in L$; then the machine which takes in $r \in \{0,1\}^{S(\ell)}$ and outputs $M(w, r)$ is a detector for G.

Lemma 2.1

If there exists a quick pesudorandom generator $G : \ell \to S(\ell)$, then for any time constructible bound $t = t(n)$:

$$BPTIME(S(t)) \subseteq DTIME(2^{O(t)})$$

Since N' uses only t bits of randomness, we can brute force it.

Define TM $N(w)$:

- for each $p \in \{0,1\}^t$, run $N'(w,p)$
- output the majority vote.

PRG $\implies$ derandomization proof ctn...

Lemma 2.1

If there exists a quick pesudorandom generator $G : \ell \to S(\ell)$, then for any time constructible bound $t = t(n)$:

$$BPTIME(S(t)) \subseteq DTIME(2^{O(t)})$$

Corollaries

Lemma 2.1

If there exists a quick pesudorandom generator $G : \ell \to S(\ell)$, then for any time constructible bound $t = t(n)$:

$$BPTIME(S(t)) \subseteq DTIME(2^{O(t)})$$

Corollaries

If there exists a __-pseudorandom generator then __:

1. $2^{\epsilon\ell}$ $\ldots BPP = P$
2. $2^{\ell^{\epsilon}}$ $\ldots BPP \subseteq QuasiP$
3. $\forall c,\ \ell^{c}$ $\ldots BPP \subseteq SUBEXP$

Pause for thought

The above lemma now gives us a goal for what we need from a pseudorandom generator.

Pause for thought

The above lemma now gives us a goal for what we need from a pseudorandom generator.

- "Long enough" stretch. Ideally exponential.

Pause for thought

The above lemma now gives us a goal for what we need from a pseudorandom generator.

- "Long enough" stretch. Ideally exponential.
- "Hard enough." In particular, hard for any circuit of size $S(\ell)^2$.

Pause for thought

The above lemma now gives us a goal for what we need from a pseudorandom generator.

- "Long enough" stretch. Ideally exponential.
- "Hard enough." In particular, hard for any circuit of size $S(\ell)^2$.

The next step is to show how to construct such a generator from circuit lowerbound assumptions.

Table of Contents

Hardness

Definition

Let $f_m : \{0,1\}^m \rightarrow \{0,1\}$ be a boolean valued function family. The *hardness* of f, denoted $H_f(m)$, is the maximum integer h_m s.t. for all circuits of size at most h_m:

$$|\Pr_{x \in U_m}[C(x) = f(x)] - \frac{1}{2}| < \frac{1}{h_m}$$

"No small circuit can approximate f"

Hardness

Definition

Let $f_m : \{0,1\}^m \to \{0,1\}$ be a boolean valued function family. The *hardness* of f, denoted $H_f(m)$, is the maximum integer h_m s.t. for all circuits of size at most h_m:

$$|\Pr_{x \in U_m}[C(x) = f(x)] - \frac{1}{2}| < \frac{1}{h_m}$$

"No small circuit can approximate f"

Lemma (Yao) Informal

Hardness on average can be amplified to worst case hardness.

Baby steps

How can we generate *one* pseudorandom bit?

Baby steps

How can we generate *one* pseudorandom bit?

Lemma (Yao)

The following are equivalent (fixing length n, size s, indistinguishability parameter ϵ):

- G_n *is pseudorandom. For any circuit C smaller than s:*

$$\Pr_{z \sim G_n}[C(z) = 1] - \Pr_{x \sim U_n}[C(x) = 1] < \epsilon$$

- G_n *is unpredictable. For any circuit C smaller than s:*

$$\Pr_{\substack{z \sim G_n \\ i \in [n]}}[C(z_1, z_2, \ldots, z_{i-1}) = z_i] < \frac{1}{2} + \epsilon/n$$

Baby steps

How can we generate *one* pseudorandom bit?

Lemma (Yao)

The following are equivalent (fixing length n, size s, indistinguishability parameter ϵ):

- G_n *is pseudorandom. For any circuit C smaller than s:*

$$\Pr_{z\sim G_n}[C(z)=1]-\Pr_{x\sim U_n}[C(x)=1]<\epsilon$$

- G_n *is unpredictable. For any circuit C smaller than s:*

$$\Pr_{\substack{z\sim G_n \\ i\in[n]}}[C(z_1,z_2,\ldots,z_{i-1})=z_i]<\frac{1}{2}+\epsilon/n$$

Goal becomes: hard $\implies$ unpredictable

Pause for ideation

Goal: $G : \ell \to \ell + 1$

$x \mapsto x \cdot f(x)$ → any "predictor" must be able to compute f w/ non-negligible probability.

But how to generalize ...

↳ we can split x into smaller pieces

↳ but this still gives at most constant additive stretch

Pause for ideation

Definitions

Nisan Wigderson Pseudorandom Generator

Let $\mathcal{Q}^\ell = \{Q_1, \ldots, Q_n\}$ where $Q_i \subseteq [\ell]$.
Let f be a boolean function.
Then we define $NW^f_{\mathcal{Q}} : \{0,1\}^\ell \to \{0,1\}^n$ as

$$NW^f_{\mathcal{Q}}(x) = f(x|_{Q_1}).f(x|_{Q_2}) \ldots f(x|_{Q_n})$$

Definitions

Nisan Wigderson Pseudorandom Generator

Let $\mathcal{Q}^\ell = \{Q_1, \ldots, Q_n\}$ where $Q_i \subseteq [\ell]$.
Let f be a boolean function.
Then we define $NW^f_{\mathcal{Q}} : \{0,1\}^\ell \to \{0,1\}^n$ as

$$NW^f_{\mathcal{Q}}(x) = f(x|_{Q_1}).f(x|_{Q_2}) \ldots f(x|_{Q_n})$$

Goal: Define $\mathcal{Q}$ so that (if f is hard) NW is pseudorandom.

Definitions

Nisan Wigderson Pseudorandom Generator

Let $\mathcal{Q}^\ell = \{Q_1, \ldots, Q_n\}$ where $Q_i \subseteq [\ell]$.
Let f be a boolean function.
Then we define $NW^f_{\mathcal{Q}} : \{0,1\}^\ell \to \{0,1\}^n$ as

$$NW^f_{\mathcal{Q}}(x) = f(x|_{Q_1}).f(x|_{Q_2}) \ldots f(x|_{Q_n})$$

Goal: Define $\mathcal{Q}$ so that (if f is hard) NW is pseudorandom. The idea will be to maximize size of each Q_i while minimizing dependence/overlap.

Definitions

(k, m)-design

Define $\mathcal{Q}^\ell$ as above. $\mathcal{Q}^\ell$ is a (k, m)-design if:

1. $\forall i, |Q_i| = m$
2. $\forall i \neq j, |Q_i \cap Q_j| \leq k$

The goal being to bound the dependence between sets.

Hardness $\implies$ PRG. Main Lemma

Lemma 2.4

Let f have hardness $H_f(m) > n^2$. Let $\mathcal{Q}^\ell$ be a $(\log n, m)$-design. Then $NW_A^f : \ell \to n$ is a pseudorandom generator.

Hardness $\implies$ PRG. Main Lemma

Lemma 2.4

Let f have hardness $H_f(m) > n^2$. Let $\mathcal{Q}^\ell$ be a $(\log n, m)$-design. Then $NW_A^f : \ell \to n$ is a pseudorandom generator.

Some notes:

- $\log n$ overlap is going to be important
- We'll come back to figuring out how to actually construct $\mathcal{Q}^\ell$

Hardness $\implies$ PRG. Proof...

f

Lemma 2.4

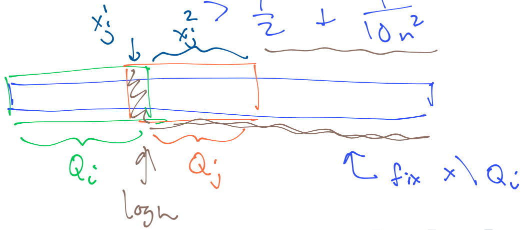
Let have hardness $H_f(m) > n^2$. Let $\mathcal{Q}^\ell$ be a $(\log n, m)$-design. Then $NW_A^f : \ell \to n$ is a pseudorandom generator.

Assume FSOC NW is NOT pseudorandom

$\Rightarrow \exists C, |C| < n^2$

$$\Pr_{\substack{x \sim U_\ell \\ i \sim [n]}}\left[C(f(x|_{Q_1}), f(x|_{Q_2}), \ldots, f(x|_{Q_{i-1}})) = f(x|_{Q_i})\right] > \frac{1}{2} + \frac{1}{10n}$$

$$\exists i \in [n], \quad \Pr\left[C(f(x_1), \ldots, f(x_{i-1})) = f(x_i)\right] > \frac{1}{2} + \frac{1}{10n^2}$$



Hardness $\implies$ PRG. Proof...

Lemma 2.4

Let have hardness $H_f(m) > n^2$. Let $\mathcal{Q}^\ell$ be a $(\log n, m)$-design. Then $NW_A^f : \ell \to n$ is a pseudorandom generator.

$$f(x_i) = f(x_i^1, \underset{\sim}{x_i^2}) \leftarrow$$
$$= f_i'(x_i')$$

$$\exists i \in [n], x \setminus Q_i \quad \Pr_{x|Q_i}[C(f_1'(x_1'), f_2'(x_2'), \ldots, f_{i-1}'(x_{i-1}')) = f(x_i)]$$

$$> \frac{1}{2} + \frac{1}{10n^2}$$

Now, we can brute force

$$f_i' : \{0,1\}^{\log n} \to \{0,1\}$$
$$\hookrightarrow \exists \text{ CNF } \phi_i, |\phi_i| < n$$

Lemma 2.4

Let have hardness $H_f(m) > n^2$. Let $\mathcal{Q}^\ell$ be a $(\log n, m)$-design. Then $NW_A^f : \ell \to n$ is a pseudorandom generator.

Define $D(\omega)$

- replace each input j of C with f_j
- set $x|_{Q_i} = x$

Then clearly $|D| \approx |C| \leq n^2$

And $\Pr[D(\omega) = f(\omega)]$

$= \Pr[C(f_1'(x_1'), \ldots, f_{i-1}'(x_{i-1}')) = f(x_i)]$

$> \frac{1}{2} - \frac{1}{10n^2}$

Contradicting Hardness. $\square$

Lemma 2.4

Let $f \in EXPTIME$ have hardness $H_f(m) > n^2$. Let $\mathcal{Q}^\ell$ be a $(\log n, m)$-design. Then $NW_A^f : \ell \to n$ is a pseudorandom generator.

Table of Contents

Constructing combinatorial designs

Lemma 2.5

Let $\log n \leq m \leq n$, and $\ell = O(m^2)$. Then there exists a $(\log n, m)$-design of size n on input length ℓ. This design is $DSPACE(\log n)$ computable (and therefore polytime computable).

Constructing combinatorial designs

Lemma 2.5

Let $\log n \leq m \leq n$, and $\ell = O(m^2)$. Then there exists a $(\log n, m)$-design of size n on input length ℓ. This design is $DSPACE(\log n)$ computable (and therefore polytime computable).

Really nice proof for this, but first to motivate the parameters...

Connecting the bounds

Theorem

If there exists a function $f \in EXPTIME$ with hardness $H_f(\ell) \geq S(\ell^c)$ for some $c > 0$, then there exists an $S(\ell^d)$-pseudorandom generator for some $d > 0$.

Connecting the bounds

$\ell \le S(\ell) \le 2^{\ell}$

Lemma 2.4

Let $f \in EXPTIME$ have hardness $H_f(m) > n^2$. Let $\mathcal{Q}^{\ell}$ be a $(\log n, m)$-design. Then $NW_A^f : \ell \to n$ is a pseudorandom generator.

Theorem

If there exists a function $f \in EXPTIME$ with hardness $H_f(\ell) \geq S(\ell^c)$ for some $c > 0$, then there exists an $S(\ell^d)$-pseudorandom generator for some $d > 0$.

$d = \frac{c}{4}$ $\quad\quad$ $\ell = m^2$

$$H_f(m) \geq S(m^c) = S(m^{4d})$$
$$= S(\ell^{2d})$$
$$\geq S(\ell^d)^2$$

Lemma 2.5

Let $\log n \leq m \leq n$, and $\ell = O(m^2)$. Then there exists a $(\log n, m)$-design of size n on input length ℓ. This design is polytime computable (in n).

The big payoff

Main Theorem

If there exists a function $f \in EXPTIME$ with hardness $H_f(\ell) \geq S(\ell^c)$ for some $c > 0$, then there exists some $d > 0$ s.t. for all time constructable bounds $t : \mathbb{N} \to \mathbb{N}$, $BPTIME(S(t^d)) \subseteq DTIME(2^{O(t^d)})$

Main corollaries

$\in EXPTIME$

If there exists a function f s.t. $H_f \geq$ __ then __:

1. $2^{\epsilon\ell}$ ... $BPP = P$
2. 2^{ℓ^ϵ} ... $BPP \subseteq QuasiP$
3. $\forall c$, ℓ^c ... $BPP \subseteq SUBEXP$

just replace —- pseudorandom

w/ $H_f \geq$ —

Back to combinatorial designs

Lemma 2.5

Let $\log n \leq m \leq n$, and $\ell = O(m^2)$. Then there exists a $(\log n, m)$-design of size n on input length ℓ. This design is $DSPACE(\log n)$ computable (and therefore polytime computable).

Goal: $\mathcal{Q} = \{ Q_1, \ldots, Q_n \} \subseteq [\ell]$

$\ell = m^2$

$[\ell] \approx \{ (a,b) : a, b \in \mathbb{F}_m \}$

$\hookrightarrow$ WLOG $m = p^d$

Define:

$S_\gamma := \{ (a, \gamma(a)) : a \in \mathbb{F}_m \}$

$\gamma \in \mathbb{F}_m[x]$

$\hookrightarrow |S_\gamma| = m$

Goal: $\{ \gamma_1, \ldots, \gamma_n \}$ s.t. $\forall i \neq j$:

$| \{ a \in \mathbb{F}_m : \gamma_i(a) = \gamma_j(a) \} | \leq \log n$

Back to combinatorial designs

$\gamma_i(a) = \gamma_j(a) \Leftrightarrow a$ is a root of $(\gamma_i - \gamma_j)$

So we restrict:

$\forall i \quad \deg(\gamma_i) \le \log n$

Lemma 2.5

Let $\log n \le m \le n$, and $\ell = O(m^2)$. Then there exists a $(\log n, m)$-design of size n on input length ℓ. This design is $DSPACE(\log n)$ computable (and therefore polytime computable).

$\{\gamma_1, \dots, \gamma_n\} \subseteq$ poly. of degree $\le \log n$

$Q = \{S_{\gamma_1}, \dots, S_{\gamma_n}\}$

① Are there enough such γ's?

$\hookrightarrow m^{\log n} \ge (\log n)^{\log n} = 2^{\log((\log n)^{\log n})}$

$= 2^{\log n (\log(\log n))}$

$= n^{\log(\log n)}$

$\ge n$

Table of Contents

Summary

- **Goal:** Show that circuit lower bounds imply derandomization

Summary

- **Goal:** Show that circuit lower bounds imply derandomization
- Step 1: PRG implies derandomization, via brute force simulation

Summary

- **Goal:** Show that circuit lower bounds imply derandomization
- Step 1: PRG implies derandomization, via brute force simulation
  - showed that if an $S(\ell)$-pseudorandom generator exists, then $BPTIME(S(t)) \subseteq DTIME(2^{O(t)})$

Summary

- **Goal:** Show that circuit lower bounds imply derandomization
- Step 1: PRG implies derandomization, via brute force simulation
  - showed that if an $S(\ell)$-pseudorandom generator exists, then $BPTIME(S(t)) \subseteq DTIME(2^{O(t)})$
- Step 2: Hardness implies PRG, via combinatorial designs

Summary

- **Goal:** Show that circuit lower bounds imply derandomization
- Step 1: PRG implies derandomization, via brute force simulation
  - showed that if an $S(\ell)$-pseudorandom generator exists, then $BPTIME(S(t)) \subseteq DTIME(2^{O(t)})$
- Step 2: Hardness implies PRG, via combinatorial designs
  - constructed a pseudorandom generator from a hard function and a combinatorial design

Summary

- **Goal:** Show that circuit lower bounds imply derandomization
- Step 1: PRG implies derandomization, via brute force simulation
  - showed that if an $S(\ell)$-pseudorandom generator exists, then $BPTIME(S(t)) \subseteq DTIME(2^{O(t)})$
- Step 2: Hardness implies PRG, via combinatorial designs
  - constructed a pseudorandom generator from a hard function and a combinatorial design
  - constructed a combinatorial design via polynomials on a finite field

A sampling of corollaries

The main corollary was a conditional result. But some nice unconditional results fall out as well.

A sampling of corollaries

The main corollary was a conditional result. But some nice unconditional results fall out as well.

- $BPAC^0 \subseteq \bigcup_c DSPACE(\log^c n)$ (using parity)

A sampling of corollaries

The main corollary was a conditional result. But some nice unconditional results fall out as well.

- $BPAC^0 \subseteq \bigcup_c DSPACE(\log^c n)$ (using parity)
- almost-PH $=$ PH

A sampling of corollaries

The main corollary was a conditional result. But some nice unconditional results fall out as well.

- $BPAC^0 \subseteq \bigcup_c DSPACE(\log^c n)$ (using parity)
- almost-PH = PH
- $BPP \subseteq \Sigma_2 \cap \Pi_2$ (not new, just a new proof)

Questions

- Can we relax this to *uniform* circuits?
- Do we get any nice followup results by applying Yao's minimax principle?
- Can this generator be used to convert public randomness to private randomness in two party-communication?

References

N. Nisan. A. Wigderson.
Hardness vs Randomness
29th Annual Symposium on Foundations of Computer Science (1988)

S. Arora. B. Barak.
Computational Complexity: A Modern Approach, 403-413