Hardness vs. randomness

- We have shown:

\[
\text{If one-way permutations exist then } \quad \text{BPP} \subset \bigcap_{\delta > 0} \text{TIME}(2^n^\delta) \subsetneq \text{EXP}
\]

- simulation is better than brute force, but just barely

- stronger assumptions on difficulty of inverting OWF lead to better simulations…
Hardness vs. randomness

• We will show:

If $E$ requires exponential size circuits then $BPP = P$

by building a different generator from different assumptions.

$$E = \bigcup_k \text{DTIME}(2^{kn})$$
Hardness vs. randomness

• BMY: for every $\delta > 0$, $G^\delta$ is a PRG with
  seed length $t = m^\delta$
  output length $m$
  error $\varepsilon < 1/m^d$ (all $d$)
  fooling size $s = m^e$ (all $e$)
  running time $m^c$

• running time of simulation dominated by $2^t$
Hardness vs. randomness

• To get $\text{BPP} = \text{P}$, would need $t = O(\log m)$
• BMY building block is one-way-permutation:
  
  \[ f : \{0,1\}^t \rightarrow \{0,1\}^t \]

• required to fool circuits of size $m^e \ (\text{all } e)$
• with these settings a circuit has time to invert $f$ by brute force!
  
  can’t get $\text{BPP} = \text{P}$ with this type of PRG
Hardness vs. randomness

• BMY pseudo-random generator:
  – one generator fooling all poly-size bounds
  – one-way-permutation is hard function
  – implies hard function in $\text{NP} \cap \text{coNP}$

• New idea (Nisan-Wigderson):
  – for each poly-size bound, one generator
  – hard function allowed to be in
    $$E = \bigcup_k \text{DTIME}(2^{kn})$$
Comparison

**BMY:** \( \forall \delta > 0 \) PRG \( G^\delta \)

- seed length: \( t = m^\delta \)
- running time: \( t^c m \)
- output length: \( m \ll m^c \)
- error: \( \varepsilon < \frac{1}{m^d} \) (all \( d \))
- fooling size: \( s = m^e \) (all \( e \))

**NW:** PRG \( G \)

- running time: \( t = O(\log m) \)
- output length: \( m^c \)
- error: \( \varepsilon < \frac{1}{m} \)
- fooling size: \( s = m \)
NW PRG

- NW: for fixed constant $\delta$, $G = \{G_n\}$ with
  - seed length $t = O(\log n)$
  - running time $n^c$
  - output length $m = n^\delta$
  - error $\varepsilon < 1/m$
  - fooling size $s = m$

- Using this PRG we obtain $\text{BPP} = \text{P}$
  - to fool size $n^k$ use $G_{n^{k/\delta}}$
  - running time $O(n^k + n^{ck/\delta})2^t = \text{poly}(n)$
NW PRG

• First attempt: build PRG assuming $E$ contains **unapproximable** functions

**Definition**: The function family 

$$f = \{f_n\}, \ f_n: \{0,1\}^n \to \{0,1\}$$

is $s(n)$-unapproximable if for every family of size $s(n)$ circuits $\{C_n\}$:

$$\Pr_x[C_n(x) = f_n(x)] \leq \frac{1}{2} + \frac{1}{s(n)}.$$
One bit

• Suppose \( f = \{f_n\} \) is \( s(n) \)-unapproximable, for \( s(n) = 2^{\Omega(n)} \), and in \( E \)

• a “1-bit” generator family \( G = \{G_n\} \):

\[
G_n(y) = y \circ f_{\log n}(y)
\]

• Idea: if not a PRG then exists a predictor that computes \( f_{\log n} \) with better than \( \frac{1}{2} + \frac{1}{s(\log n)} \) agreement; contradiction.
One bit

• Suppose \( f = \{f_n\} \) is \( s(n) \)-unapproximable, for \( s(n) = 2^{\delta n} \), and in \( \mathbb{E} \)

• a “1-bit” generator family \( G = \{G_n\} : \)
  \[
  G_n(y) = y \circ f_{\log n}(y)
  \]

  – seed length \( t = \log n \)
  – output length \( m = \log n + 1 \)
  – fooling size \( s \approx s(\log n) = n^\delta \)
  – running time \( n^c \)
  – error \( \varepsilon \approx 1/s(\log n) = 1/n^\delta \)
Many bits

• Try outputting many evaluations of $f$:

$$G(y) = f(b_1(y)) \circ f(b_2(y)) \circ \ldots \circ f(b_m(y))$$

• Seems that a predictor must evaluate $f(b_i(y))$ to predict $i$-th bit

• Does this work?
Many bits

• Try outputting many evaluations of \( f \):
  \[
  G(y) = f(b_1(y)) \circ f(b_2(y)) \circ \ldots \circ f(b_m(y))
  \]

• predictor might notice correlations without having to compute \( f \)

• but, more subtle argument works for a specific choice of \( b_1 \ldots b_m \)
Nearly-Disjoint Subsets

**Definition:** $S_1, S_2, \ldots, S_m \subset \{1 \ldots t\}$ is an $(h, a)$ design if

- for all $i$, $|S_i| = h$
- for all $i \neq j$, $|S_i \cap S_j| \leq a$
Nearly-Disjoint Subsets

**Lemma**: for every $\varepsilon > 0$ and $m < n$ can in $\text{poly}(n)$ time construct an

$$(h = \log n, a = \varepsilon \log n)$$

design

$S_1, S_2, \ldots, S_m \subseteq \{1 \ldots t\}$ with $t = O(\log n)$. 

May 1, 2017
Nearly-Disjoint Subsets

• Proof sketch:
  – pick random \((\log n)\)-subset of \(\{1 \ldots t\}\)
  – set \(t = O(\log n)\) so that expected overlap with a fixed \(S_i\) is \(\varepsilon \log n/2\)
  – probability overlap with \(S_i\) is \(> \varepsilon \log n\) is at most \(1/n\)
  – union bound: some subset has required small overlap with all \(S_i\) picked so far…
  – find it by exhaustive search; repeat \(n\) times.
The NW generator

- \( f \in E \) \( s(n) \)-unapproximable, for \( s(n) = 2^{\delta n} \)
- \( S_1, \ldots, S_m \subseteq \{1 \ldots t\} \) (log \( n \), \( a = \delta \log n/3 \))

Design with \( t = O(\log n) \)

\[
G_n(y) = f_{\log n}(y_{|S_1}) \circ f_{\log n}(y_{|S_2}) \circ \ldots \circ f_{\log n}(y_{|S_m})
\]

\( f_{\log n} : 010100101111101010111001010 \)

seed \( y \)
The NW generator

**Theorem** (Nisan-Wigderson): $G=\{G_n\}$ is a pseudo-random generator with:

- seed length $t = O(\log n)$
- output length $m = n^{\delta/3}$
- running time $n^c$
- fooling size $s = m$
- error $\varepsilon = 1/m$

May 1, 2017
The NW generator

• Proof:
  – assume does not $\varepsilon$-pass statistical test $C = \{C_m\}$ of size $s$:
    \[
    |\Pr_x[C(x) = 1] - \Pr_y[C(G_n(y)) = 1]| > \varepsilon
    \]
  – can transform this **distinguisher** into a **predictor** $P$ of size $s' = s + O(m)$:
    \[
    \Pr_y[P(G_n(y)_1 \ldots i-1) = G_n(y)_i] > \frac{1}{2} + \frac{\varepsilon}{m}
    \]
The NW generator

\[ G_n(y) = f_{\log n}(y_{|S_1}) \circ f_{\log n}(y_{|S_2}) \circ \ldots \circ f_{\log n}(y_{|S_m}) \]

\[ f_{\log n}: \quad 010100101111101010111001010 \]

- Proof (continued):

\[ \Pr_y[\mathbb{P}(G_n(y)_1 \ldots i-1) = G_n(y)_i] > \frac{1}{2} + \frac{\varepsilon}{m} \]

- fix bits outside of \( S_i \) to preserve advantage:

\[ \Pr_{y'}[\mathbb{P}(G_n(\alpha y' \beta)_1 \ldots i-1) = G_n(\alpha y' \beta)_i] > \frac{1}{2} + \frac{\varepsilon}{m} \]
The NW generator

\[ G_n(y) = f_{\log n}(y_{|S_1}) \circ f_{\log n}(y_{|S_2}) \circ \ldots \circ f_{\log n}(y_{|S_m}) \]

- Proof (continued):
  - \( G_n(\alpha y' \beta)_i \) is exactly \( f_{\log n}(y') \)
  - for \( j \neq i \), as vary \( y' \), \( G_n(\alpha y' \beta)_j \) varies over \( 2^a \) values!
  - hard-wire up to \((m-1)\) tables of \( 2^a \) values to provide \( G_n(\alpha y' \beta)_1 \ldots i-1 \)

May 1, 2017
The NW generator

\[ G_n(y) = f_{\log n}(y_{|S_1}) \circ f_{\log n}(y_{|S_2}) \circ \ldots \circ f_{\log n}(y_{|S_m}) \]

\[ f_{\log n}: \quad \text{010100101111101010111001010} \]

- size \( m + O(m) + (m-1)2^\alpha \) < \( s(\log n) = n^\delta \)
- advantage \( \epsilon/m = 1/m^2 > 1/y' \) → \( s(\log n) = n^{-\delta} \)
- contradiction

May 1, 2017
Worst-case vs. Average-case

**Theorem (NW):** if $E$ contains $2^{\Omega(n)}$-unapproximable functions then $\text{BPP} = \text{P}$.

- How reasonable is unapproximability assumption?
- Hope: obtain $\text{BPP} = \text{P}$ from worst-case complexity assumption
  - try to fit into existing framework without new notion of “unapproximability”
Worst-case vs. Average-case

**Theorem** (Impagliazzo-Wigderson, Sudan-Trevisan-Vadhan)

If $E$ contains functions that require size $2^{\Omega(n)}$ circuits, then $E$ contains $2^{\Omega(n)}$–unapproximable functions.

• Proof:
  – main tool: *error correcting code*
Error-correcting codes

• Error Correcting Code (ECC):
  \[ C: \Sigma^k \to \Sigma^n \]

• message \( m \in \Sigma^k \)

• received word \( R \)
  \( \sim C(m) \) with some positions corrupted

• if not too many errors, can decode: \( D(R) = m \)

• parameters of interest:
  – rate: \( k/n \)
  – distance:
    \[ d = \min_{m \neq m'} \Delta(C(m), C(m')) \]
Distance and error correction

- C is an ECC with distance d
- can *uniquely* decode from up to \[
\left\lfloor \frac{d}{2} \right\rfloor
\]
  errors
Distance and error correction

- can find **short list** of messages (one correct) after closer to $d$ errors!

**Theorem** (Johnson): a binary code with distance $(\frac{1}{2} - \delta^2)n$ has at most $O(1/\delta^2)$ codewords in any ball of radius $(\frac{1}{2} - \delta)n$. 
Example: Reed-Solomon

- alphabet $\Sigma = \mathbb{F}_q$: field with $q$ elements
- message $m \in \Sigma^k$
- polynomial of degree at most $k-1$
  \[ p_m(x) = \sum_{i=0}^{k-1} m_i x^i \]
- codeword $C(m) = (p_m(x))_x \in \mathbb{F}_q$
- rate $= k/q$
Example: Reed-Solomon

• Claim: distance \( d = q - k + 1 \)
  – suppose \( \Delta(C(m), C(m')) < q - k + 1 \)
  – then there exist polynomials \( p_m(x) \) and \( p_{m'}(x) \)
    that agree on more than \( k-1 \) points in \( \mathbb{F}_q \)
  – polynomial \( p(x) = p_m(x) - p_{m'}(x) \) has more than \( k-1 \) zeros
  – but degree at most \( k-1 \)…
  – contradiction.
Example: Reed-Muller

• Parameters: $t$ (dimension), $h$ (degree)
• alphabet $\Sigma = F_q$ : field with $q$ elements
• message $m \in \Sigma^k$
• multivariate polynomial of total degree at most $h$

$$p_m(x) = \sum_{i=0}^{k-1} m_i M_i$$

$\{M_i\}$ are all monomials of degree $\leq h$
Example: Reed-Muller

- $M_i$ is monomial of total degree $h$
  - e.g. $x_1^2x_2x_4^3$
  - need # monomials $(h+t \text{ choose } t) > k$

- codeword $C(m) = (p_m(x))_{x \in (F_q)^t}$

- rate $= k/q^t$

- Claim: distance $d = (1 - h/q)q^t$
  - proof: Schwartz-Zippel: polynomial of degree $h$ can have at most $h/q$ fraction of zeros
Codes and hardness

• Reed-Solomon (RS) and Reed-Muller (RM) codes are efficiently encodable

• efficient **unique decoding?**
  – yes (classic result)

• efficient **list-decoding?**
  – yes (RS on problem set)
Codes and Hardness

• Use for worst-case to average case:
  
  truth table of $f: \{0, 1\}^{\log k} \rightarrow \{0, 1\}$
  
  (worst-case hard)

  $m: \begin{array}{cccccccc}
  0 & 1 & 1 & 0 & 0 & 0 & 1 & 0
  \end{array}$

  truth table of $f': \{0, 1\}^{\log n} \rightarrow \{0, 1\}$
  
  (average-case hard)

  $Enc(m): \begin{array}{cccccccccccc}
  0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0
  \end{array}$
Codes and Hardness

• if \( n = \text{poly}(k) \) then
  \[ f \in \mathcal{E} \implies f' \in \mathcal{E} \]

• Want to be able to prove:
  if \( f' \) is \( s' \)-approximable,
  then \( f \) is computable by a
  size \( s = \text{poly}(s') \) circuit
Codes and Hardness

• Key: circuit $C$ that approximates $f'$ implicitly gives received word $R$

$$\begin{array}{cccccccccc}
R: & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
\end{array}$$

$$\begin{array}{cccccccccc}
Enc(m): & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
\end{array}$$

• Decoding procedure $D$ “computes” $f$ exactly

- Requires special notion of efficient decoding
Codes and Hardness

\[ f : \{0,1\}^{\log k} \rightarrow \{0,1\} \]

\[ f' : \{0,1\}^{\log n} \rightarrow \{0,1\} \]

\[ m : \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \]

\[ \text{Enc}(m) : \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \]

\[ R : \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \]

Small circuit \( C \) approximating \( f' \)

Small circuit that computes \( f \) exactly

Decoding procedure

\[ i \in \{0,1\}^{\log k} \]

\[ f(i) \]