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 \( G^{\delta} \)

NW: PRG \( G \)

<table>
<thead>
<tr>
<th>Seed length</th>
<th>Running time</th>
<th>Output length</th>
<th>Error</th>
<th>Fooling size</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t = m^\delta )</td>
<td>( t = O(\log m) )</td>
<td>( m )</td>
<td>( \epsilon &lt; 1/m )</td>
<td>( s = m )</td>
</tr>
</tbody>
</table>

NW PRG

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

  • Using this PRG we obtain \( \text{BPP} = \text{P} \)
    – to fool size \( n^k \) use \( G_{n^k} \)
    – running time \( O(n^k + n^{k\delta}2^k) = \text{poly}(n) \)

NW PRG

• First attempt: build PRG assuming \( E \) contains unapproximable functions

Definition: The function family

\[ f = \{ f_n \}, f_n : \{0,1\}^n \rightarrow \{0,1\} \]

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

\[ \Pr_x[C_n(x) = f_n(x)] \leq 1/2 + 1/s(n). \]

One bit

• Suppose \( f = \{ f_n \} \) is \( s(n) \)-unapproximable, for \( s(n) = 2^{O(n^k)} \), 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 \( 1/2 + 1/s(\log n) \) agreement; contradiction.
One bit

- Suppose \( f = \{ f_n \} \) is \( s(n) \)-unapproximable, for \( s(n) = 2^{\delta n} \), and in \( E \)
- a "1-bit" generator family \( G = \{ G_n \} \):
  \[ G_n(y) = y \oplus f_{\log n}(y) \]
  - seed length \( t = \log n \)
  - output length \( m = \log n + 1 \) (want \( n^0 \))
  - fooling size \( s \approx s(\log n) = n^{\delta} \)
  - running time \( n^{\delta} \)
  - error \( \epsilon \approx 1/s(\log n) = 1/n^{\delta} \)

Many bits

- Try outputting many evaluations of \( f \):
  \[ G(y) = f(b_1(y)) \oplus f(b_2(y)) \oplus \ldots \oplus f(b_m(y)) \]
- 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 \( f \):
  \[ G(y) = f(b_1(y)) \oplus f(b_2(y)) \oplus \ldots \oplus f(b_m(y)) \]
- 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**: \( S_1, S_2, \ldots, S_m \subseteq \{1 \ldots t\} \) is an \((h, a)\) design if
- for all \( i \), \( |S_i| = h \)
- for all \( i \neq j \), \( |S_i \cap S_j| \leq a \)

**Lemma**: for every \( \epsilon > 0 \) and \( m < n \) can in \( \text{poly}(n) \) time construct an
\( (h = \log n, a = \epsilon \log n) \) design
\( S_1, S_2, \ldots, S_m \subseteq \{1 \ldots t\} \) with \( t = O(\log n) \).

**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 \( \epsilon \log n \) 
- probability overlap with \( S_i \) is \( > \epsilon \log n \) 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

\( f \in E \) unapproximable, for \( s(n) = 2^{\delta n} \)

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

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

\[ f_{\log n} = \begin{bmatrix} 0101001011101010111001010 \end{bmatrix} \]

\( S_1, \ldots, S_m \subseteq \{1, \ldots, t\} \) \( \log n, a = \delta \log n / 3 \)

- \( f \) \in \( E \)

\[ s(n) = 2^{\delta n}, \ldots, f_{\log n}(y_{S_m}) \]

- \( S_1, \ldots, S_m \subseteq \{1, \ldots, t\} \)

- \( \log n, a = \delta \log n / 3 \)

\( \delta \log n / 3 \)

- \( \text{output length} m = n^{\delta / 3} \)

- \( \text{running time} n^c \)

- fooling size \( s = m \)

- error \( \epsilon = 1 / m \)

- contradiction

- size \( m = O(m) \times (m-1)2^s \)

- \( s(\log n) = n^s \)

- advantage \( \epsilon / m = 1 / m^s \)

- contradiction

- hardwired tables
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"

Error-correcting codes

**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

Distance and error correction

- $C$ is an ECC with distance $d$
- can uniquely decode from up to $\lfloor d/2 \rfloor$ errors

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

Example: Reed-Solomon

- alphabet $\Sigma = F_q$: field with $q$ elements
- message $m \in \Sigma^k$
- polynomial of degree at most $k-1$
  $$p_m(x) = \sum_{i=0}^{k-1} m_ix^i$$
- codeword $C(m) = (p_m(x))_{x \in F_q}$
- 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 $F_q$
  - 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 $\leq h$
- $\text{codeword } C(m) = (p_m(x))_{x \in (F_q)^t}$
- rate $= k/q$

Example: Reed-Muller

- $M_i$ is monomial of total degree $h$
  - e.g. $x_1^2 x_2 x_4^3$
  - need $\#$ monomials $(h + t \text{ choose } t) > k$
- codeword $C(m) = (p_m(x))_{x \in (F_q)^t}$
- rate $= k/q$
- 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)
  - $m = 011000010$
    - truth table of $f' : \{0,1\}^{2\log n} \rightarrow \{0,1\}$
      - (average-case hard)
    - $\text{Enc}(m) = 0110^001000010$

Codes and Hardness

- if $n = \text{poly}(k)$ then
  - $f \in E$ implies $f' \in E$

- 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 $R$
- Decoding procedure $D$ computes $z$ exactly
- Requires special notion of efficient decoding

Encoding

- encoding procedure (continued):
  - Hadamard code $\text{Had}: (0,1)^{\log q} \rightarrow (0,1)^t$
    - $t$ (dimension), $q$ (field size)
    - distance $\frac{1}{2}$ by Schwartz-Zippel
  - final codeword: $(\text{Had}(p_m(x)))_i \in F^t_q$
    - evaluate $p_m$ at all points, and encode each evaluation with the Hadamard code

Decoding

- small circuit $C$ computing $R$, agreement $\frac{1}{2} + \delta$
- Decoding step 1
  - produce circuit $C'$ from $C$
  - given $x \in F^t_q$, outputs “guess” for $p_m(x)$
  - $C'$ computes $z : \text{Hadamard has agreement } \frac{1}{2} + \delta/2$
  - with $x$-th block, outputs random $z$ in this set

Decoding

- **Decoding step 1** (continued):
  - for at least $\delta/2$ of blocks, agreement in block is at least $1/2 + \delta/2$
  - Johnson Bound: when this happens, list size is $S = O(1/\delta^2)$, so probability $C'$ correct is $1/S$
  - altogether:
    - $\Pr[C'(x) = p_m(x)] \geq \Omega(\delta^3)$
    - $C'$ makes $q$ queries to $C$
    - $C'$ runs in time $\text{poly}(q)$

- **Decoding step 2**
  - produce circuit $C''$ from $C'$
  - given $x \in \text{emb}(1,2,\ldots,k)$ outputs $p_m(x)$
  - idea: restrict $p_m$ to a random curve; apply efficient R-S list-decoding; fix “good” random choices

Restricting to a curve

- points $x=\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_r \in F_q$ specify a degree $r$ curve $L: F_q \rightarrow F_q$
  - $w_1, w_2, \ldots, w_r$ are distinct elements of $F_q$
  - for each $L_i: F_q \rightarrow F_q$
    - is the degree $r$ poly for which $L_i(w_j) = (\alpha_j)^i$ for all $i$
  - Write $p_m(L(z))$ to mean $p_m(L_1(z), L_2(z), \ldots, L_t(z))$
  - $p_m(L(w_i)) = p_m(x)$