Today

1. finish AEL construction
   + GI algorithm

2. Codes & complexity

   AEL Codes

   3 ingredients:
   ① \( C_0 \): small code \( [d_1, R_0d, S_0d]_2 \)
   ② \( G \): bipartite graph with \( n \) left & right vertices
     & degree \( d \); \( G \)-uniform
   ③ \( C_{\text{big}} \): big code \( [n, k, D^2]_2 \)

   Combined code: \( C = \{ n, R_0k, 2^d \} \)

   **Defn:** \( G \) is \( E \)-uniform if \( \forall X \subseteq L \),
   \( \forall Y \subseteq R \),
   \[ |\# E(x, y) - d \cdot n \cdot \frac{1 \times 1}{n} n!| \leq \epsilon n. \]
Lemma: $S(c) \geq S_0 - \frac{\varepsilon n}{Dd}$

Proof: Let $w_{C_{big}}$ be word of distance $\geq D$.
- Let $X \subseteq L$ be set where $w_i \neq 0$.
- Let $Y \subseteq R$ be set where $w_j \neq 0$.

\[
\# \text{non-zero edges} \geq |X||. S_0 \cdot d
\]

But \# edges $\leq E(x, y)$

\[
\leq \frac{|X|}{n} \cdot \frac{|Y|}{n} \cdot d \cdot n + \varepsilon n
\]

\[
\Rightarrow \frac{|X|}{n} \geq \left( \frac{|X| \cdot S_0 \cdot d - \varepsilon n}{|X| \cdot d} \right)
\]

\[
= S_0 - \frac{\varepsilon \cdot n}{|X| \cdot d} \geq S_0 - \frac{\varepsilon \cdot n}{Dd}
\]

Will skip decoding but it leads to

$O(n)$ time decoding with $\frac{1}{4} - \varepsilon$ error (in binary code) \cite{Gurumani-Hdyak}
Codes & Computational Complexity

Obvious Direction:

- Codes need Encoding + Decoding.
- Needs efficient algorithms.
- When are they possible? Intractable?
- Will briefly discuss today.

Non-obvious direction

- Codes are combinatorial structures with some nice properties.
- Most extremal structures are connected to one another.
- Errors, model uncertainty / lack of knowledge. Often captures adversary.
Theme I - Pseudorandomness

(See excellent survey by [Salil Vadhan])

Background: Randomized algorithms

Computational Problem

Function $x \xrightarrow{f} y = f(x)$

or

Relation $x \xrightarrow{R} y$ s.t. $(x, y) \in R$

$P$ = class of functional problems solvable in polytime, where range of $f$ is $\{0, 1\}$ (Boolean).
Randomized algorithm

\[ x \xrightarrow{} A \xrightarrow{} A(x,y) \]

A probabilistically computes $f$ if

\[ \forall x \quad \Pr_y \left[ A(x,y) = f(x) \right] \leq \frac{1}{3} \]

A is polytime if

1. $|y| \leq |x|^c$ for constant $c$.
2. Running time of $A$ is poly.
**Example:** MAX 3SAT:

*Input:* \( \phi = C_1, C_2, \ldots, C_m \).

*Clause*

\[ C_j = x_{i_1(j)} \lor x_{i_2(j)} \lor x_{i_3(j)} \]

*Literal* \( = \) variable or its complement.

*(Desired) Output:* \( a_1, \ldots, a_n \in \{0,1\} \)

that maximizes

\[ \# \sum_j \{ C_j \text{ "satisfied" }, \text{i.e. one } \]

\( \phi \text{ literals in } C_j \text{ is } \geq \frac{7}{8} \}

Well known: \( \text{NP-hard to solve optimally.} \)

Can we do something "near optimally?"

\[ \exists \text{ prob. } \frac{7}{8} \text{-approximator:} \]

finds \( \overline{a} = a_1, \ldots, a_n \) s.t.

\[ \# \text{ satisfied clauses } \geq \left\lfloor \frac{7}{8} m \right\rfloor \]
Alg: Pick $a_1, \ldots, a_n$ at random
(uniformly in $\mathbb{F}_2^n$)

Analysis:

\[
\Pr \left[ C_j \text{ satisfied} \right] = \frac{7}{8} \\
\Rightarrow \mathbb{E} \left[ \# \{ j \mid C_j \text{ satisfied} \} \right] = \frac{7}{8} m
\]

(more formal analysis would pick many vectors $a \in \mathbb{F}_2^n$ and output best)

------------------

Question: Can we build "deterministic"
algorithm?

Or an algorithm that uses less randomness?

Or doesn't need "pure" (unbiased, independent) randomness?
The Randomness Processing Industry

\[ z \xrightarrow{\text{P}} y \]

\[ P: \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q^n \]

- Exact functionality of \( P \) is unimportant.

- Key issue: if \( z \sim D_1 \)
  then how is \( P(z) \) distributed?

- Pseudorandom generator: (for \( (A,f) \))
  - \( d \ll n \)
  - if \( z \) uniform over \( \mathbb{Z}_q^n \)

\[ \forall x \]

\[ Pr[ A(x, P(z)) = f(x) ] \approx \sum_{y} Pr[ A(x, y) = f(x) ] \]
Other concepts: Dispersers, Extractors, Condensers, Mergers ...
(can be seeded/unseeded, lossless/lossy, zero-error/\texttt{re}-error)

Today: Pseudo-Randomness.

if p.r.g.s exist for every polytime $A$, with $l = O(Lgn)$,
then $\text{BPP} = \text{P}$

```
\uparrow
\text{prob. polytime}
```

"for every $A$" — open
but simple $A$ — like MAX 3SAT
above can be derandomized.
I. Limited Independence

- Note that $a_1, \ldots, a_n$ don't have to be completely independent; only limitedly so.

- Sufficient that for $i, j, k$

  $$(a_i, a_j, a_k) \text{ are uniform.}$$

- **Defn:** $\ell$-wise independence

  $Y = (y_1, \ldots, y_n) = P(z_1, \ldots, z_\ell)$ is $\ell$-wise independent if

  $\forall S \subseteq [n], \ |S| = \ell$

  $\forall b \in \{0, 1\}^\ell$

  $$\Pr_{Y} \left[ P(z) \big|_{S} = b \right] = \frac{1}{2^\ell}$$

  ("$P(z)$ restricted to $S$ is uniform")
Claim: Mx 3SAT algorithm works as well with 3-wise independent sources.

**Lemma:** Let \( C \) be an \([n, k, \geq t]\) code with \( C^\perp = [n, n-k, t+1]\) code.

Let \( \mathcal{E}: \{0,1\}^k \rightarrow \{0,1\}^n \) be encoder of \( C \). Then \( \{\mathcal{E}(z)\}_{z \in \{0,1\}^k}^{2^{-t}} \) is \( t \)-wise independence.

**Proof:** Follows from definitions.

No codewords in \( C^\perp \) of wt. \( \leq t \)

\( \Rightarrow \) No linear dependence in \( \leq t \) coordinates of \( C \)

\( \Rightarrow \) No dependence on \( \leq t \) coordinates of \( C \)
To make generator good, use smallest \( k \) possible \( \Rightarrow \) use best possible (highest rate) \( C^+ \).

Using best known codes

1. **Pairwise Independence**:
   
   \( C = \text{Hadamard Code} = \text{linear function} \)
   
   \( C^+ = \text{Hamming Code} \)
   
   \( k = \log n \quad [\text{no} \quad "O(\cdot)"\] \)

2. **\( t \)-wise independence**:
   
   \( C = \text{dual-BCH code} \)
   
   \( C^+ = \text{BCH code} \)
   
   \( k \approx \frac{t}{2} \log n \)

3. **3-wise independence**
   
   \( C = \text{affine functions.} \)
   
   \( |C| = 2n \)
II. Small-Biased Spaces

**Definition:**

\[
\bar{y} \in \epsilon \text{-biased if } \forall \, S \subseteq \{n\}, S \neq \emptyset \quad \left| \Pr_\bar{y} \left[ \bigoplus_{i \in S} y_i = 1 \right] - \Pr_\bar{y} \left[ \bigoplus_{i \in S} y_i = 0 \right] \right| \leq \epsilon
\]

**Motivation:**

- **Definitionally:** Output of \( P \) "fools" every linear algorithm.

- **Real reason:**
  1. Almost limited independence
  2. Ingredients in fooling many other algorithms.
almost $\epsilon$-wise independence

if $\forall S \subseteq [n], |S| = t$

$$\sum_{b \in \mathbb{F}_{2^t}} |\Pr[y|_S = b] - 2^{-t}| \leq \epsilon$$

Lemma: $G$-biased space

is $\epsilon \cdot 2^t$-almost $\epsilon$-wise independent

Lemma: Let $P_1 : \mathbb{F}_{2^t} \rightarrow \mathbb{F}_{2^t}$ generate $G$-biased bits

- Let $P_2 : \mathbb{F}_{2^t} \rightarrow \mathbb{F}_{2^t}^n$ be linear and $\epsilon$-wise independent

- Then $P_2 \circ P_1 : \mathbb{F}_{2^t} \rightarrow \mathbb{F}_{2^t}^n$

\((\epsilon \cdot 2^t)$-almost $\epsilon$-wise independent\)
**Lemma:** Let $G \in \mathbb{F}_{2}^{K \times N}$ be the generator of code of distance $(\frac{1}{2} - \epsilon)N$. Let $\mathcal{T}^c$ be code $(G)$.

Then the map

$P: \mathbb{F}_{2}^{\log N} \rightarrow \mathbb{F}_{2}^{K}$

that maps $i \mapsto i^{th}$ column of $G$

is an $\epsilon$-biased generator. $\blacksquare$

Putting all together with MAX 1OSAT

<table>
<thead>
<tr>
<th>Random assignment</th>
<th>$1 - 2^{-10}$</th>
<th>$2^n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-wise indep.</td>
<td>$1 - 2^{-10}$</td>
<td>$n^5$</td>
</tr>
<tr>
<td>$\epsilon$-bias with $\epsilon = 2^{-22}$</td>
<td>$1 - 2^{-10} - 2^{-n}$</td>
<td>$O\left(\frac{n}{\epsilon^3}\right)$</td>
</tr>
<tr>
<td>$\epsilon 2^{10}$-almost 10-wise indep.</td>
<td>$\left(\frac{\log n}{\epsilon^3}\right)^5$</td>
<td></td>
</tr>
</tbody>
</table>

- Method
- Fraction of clauses sat.
- Sample space