Universal methods for derandomization

Lecture 2
Dispersers and Extractors

Yesterday

• Five examples of the usefulness of randomness in efficient computation.

• One universal task (finding hay in a haystack) which captures at least 2 of them with respect to derandomization.

A Universal Task:
Finding Hay in a Haystack
(Black Box version)

Given black box $C: \{0,1\}^n \to \{0,1\}$ with $\mu(C) \geq \frac{1}{2}$, find $x$ so that $C(x) = 1$.

Want: Algorithm polynomial in $n$.

A Universal Task:
Finding Hay in a Haystack
(Circuit version)

Given circuit $C: \{0,1\}^n \to \{0,1\}$ with $\mu(C) \geq \frac{1}{2}$, find $x$ so that $C(x) = 1$.

Want: Algorithm polynomial in $n$ and size of $C$.

Hypothesis H

There exists deterministic polytime procedure $\text{findHay}$ taking as input a circuit $C: \{0,1\}^n \to \{0,1\}$ so that

1. $\text{findHay}(C)$ is in $\{0,1\}^n$.
2. If $\mu(C) \geq \frac{1}{2}$ then $C(\text{findHay}(C)) = 1$. 
Fact

The constant $\frac{1}{2}$ can be replaced with
• any constant strictly between 0 and 1,
• $n^k$ (quite close to 0), or
• $1 - 2^{-n^{k'}}$ (very close to 1),
without changing truth value of Hypothesis H.
This is not the end of the story – we shall do even better later today!

Numerical integration revisited

Density Estimation: Given circuit $C: \{0,1\}^n \rightarrow \{0,1\}$, estimate $\mu(C)$ within additive error $\epsilon$.

Desired: Deterministic algorithm running in time polynomial in $C$ and $1/\epsilon$.

Finding Hay Derandomizes Monte Carlo Integration

Hypothesis H

An efficient deterministic algorithm for density estimation exists.

Lemma

To solve the density estimation problem it is sufficient to make polynomial procedure Estimate so that:
For $C: \{0,1\}^n \rightarrow \{0,1\}$,
• $\mu(C) \leq 2^{-n^{k'}} \Rightarrow$ Estimate($C$) returns small
• $\mu(C) \geq 1 - 2^{-n^{k'}} \Rightarrow$ Estimate($C$) returns big

Reduction

ApproximateDensity($C, \epsilon$)
for($\alpha$=0 to 1 step $\epsilon/2$)
Let $C_\alpha =$
[ $(x_1, x_2, \ldots, x_t) \rightarrow$
  if $\#(j | C(x_j) = 1) / t$ is $\epsilon/2$-close to $\alpha$
  then 1 else 0 ]
if Estimate($C_\alpha$) = big then return $\alpha$

Estimation algorithm, first attempt

Estimate($C: \{0,1\}^n \rightarrow \{0,1\}$),
if $(\text{findHay}(C)) = 1$ return big;
else return small;
If \( C \) is very small, the union of a small number of random translates of \( C \) is still a small set.

If \( C \) is very big, a small number of random translates of \( C \) covers everything with high probability.

**Lemma** (page 15, top)

- Let \( C: \{0,1\}^n \to \{0,1\}, \ C \subseteq \{0,1\}^n \)
- Pick \( y_1, y_2, ..., y_n \) at random in \( \{0,1\}^n \).
- Let \( C = \bigcup C \oplus y_i \)
- \( \mu(C) \leq 2^{-n/2} \Rightarrow \mu(C) \leq n2^{-n/2} \)
- \( \mu(C) \geq 1 - 2^{-n/2} \Rightarrow \Pr[\mu(C) = 1] > 1/2 \)

**Estimation algorithm**

\[
\text{Estimate}(C: \{0,1\}^n \to \{0,1\}), \\
\text{if } D(\text{findHay}(D))=1 \text{ return } \text{big}; \\
\text{else return } \text{small}; \\
\]

- **D**: Input: \( y_1, y_2, ..., y_n \) in \( \{0,1\}^n \).
  - Output: 0 if \( E(\text{findHay}(E))=1 \), 1 otherwise.

- **E**: Input: \( x \) in \( \{0,1\}^n \).
  - Output: 0 if \( \exists i: C(x \oplus y_i) = 1 \), 1 otherwise.

**Analysis**

- The characteristic set of \( E \) is the complement of \( \overline{C} \)
- If \( \mu(C) \) is very small, \( \mu(E) \) is very big, no matter what \( y_1, y_2, ..., y_n \) are. Hence \( D \) always outputs 0 and \( \text{Estimate}(C) \) returns small.
- If \( \mu(C) \) is very big, \( \mu(E) = 0 \) for more than half the possible values of \( y_1, y_2, ... \). Hence \( \mu(D) > 1/2 \) and \( \text{Estimate}(C) \) returns big.

**PrP vs. PrRP**

- In the proof, Hypothesis H can be replaced with the hypothesis that we can efficiently distinguish between circuits \( C \) with \( \mu(C)=0 \) and circuits \( C \) with \( \mu(C) \geq 1/2 \).
- This is the \( \text{PrP} = \text{PrRP} \) assumption discussed in notes.
- It is not known how to get the conclusion assuming only \( \text{P} = \text{RP} \) or \( \text{P} = \text{BPP} \).
Exercise

If Hypothesis H is true, all randomized approximation algorithms and heuristics can be derandomized:

On input $x$, the expected quality of solution found by randomized algorithm is $q$

On input $x$, the quality of the solution found by deterministic algorithm is $(1+\epsilon)q$

It is not known how to get this assuming only $\Pr[\mathcal{RP}] = \Pr[\mathcal{RP}]$.

If Hypothesis H is true, can the construction of hard truth tables be derandomized?

Not so clear.....

But we shall see that the converse is the case! This is a cornerstone in the theory of derandomization: The Impagliazzo-Wigderson 1997 result.

Not all applications of the probabilistic method can be derandomized!

• Let $R$ be the set of Kolmogorov random strings: Strings whose shortest effective description (i.e. shortest generating program) is almost as long as the string itself.

• A randomly chosen string is in $R$ with high probability, but no recursive procedure can generate an element of $R$ of length $n$ on input $n$.

Other non-derandomizable settings

• Crypto
• Other Multiparty settings

Robustness of Hypothesis H

The constant $\frac{1}{2}$ can be replaced with

• any constant strictly between 0 and 1,
• $n^{-\epsilon}$ (quite close to 0), or
• $1-2^{-n^{1-\epsilon}}$ (very close to 1), without changing truth value of Hypothesis H.

To get even closer to 1, we should try to reduce error probability using few random bits. This question can be considered even in black-box model.

Independent Sampling

Let $T: \{0,1\}^n \rightarrow \{0,1\}$. We can find a point $x$ with $T(x)=1$ with

• Random bit usage: $m \cdot n$.
• $m$ probes.
• Error probability $2^{-m}$.

Can we achieve a better tradeoff?
Exhaustive Search
We can find a point \( x \) with \( T(x) = 1 \) with
- Random bit usage: 0.
- \( 2^n/2 + 1 \) probes.
- Error probability: 0.

We shall restrict our attention to protocols using \( n^{O(1)} \) probes and consider tradeoffs between the other two parameters.

Dispersers
- A **Disperser** is an efficient algorithm encoding a particular strategy for finding hay:
  \[
  D: \{0,1\}^R \times \{1,\ldots,m\} \rightarrow \{0,1\}^n
  \]
- Using \( R \) random bits \( y \), the disperser probes
  \( T(D(y,1)), \ldots, T(D(y,m)) \).
- The error probability of the disperser is
  \[
  \max_{T, \mu(T) \geq 1/2} \Pr \left[ \forall i : T(D(y,i)) = 0 \right]
  \]

“Amplification by Repetition” viewed as disperser
- \( R = mn \)
- \( D(x_1, x_2, \ldots, x_R, i) = x_{n(i-1)+1} x_{n(i-1)+2} \cdots x_{n(i-1)+n} \)
- Error probability \( 2^{-m} \).

Chor-Goldreich disperser
- Let \( \cdot, + \) be arithmetic operations over \( \mathbb{GF}[2^n] \)
  \[
  D((a,b), x) = a \cdot x + b
  \]
- For different \( x \) and \( y \) and random \( (a,b) \),
  \( a \cdot x + b \) and \( a \cdot y + b \) are **independent** random variables.

<table>
<thead>
<tr>
<th>Dispersers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Chor-Goldreich ’86</strong></td>
</tr>
<tr>
<td><strong>Karp-Pippenger-Sipser ’86</strong></td>
</tr>
<tr>
<td><strong>Trevisan ’99</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dispersers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Chor-Goldreich ’86</strong></td>
</tr>
<tr>
<td><strong>Karp-Pippenger-Sipser ’86</strong></td>
</tr>
<tr>
<td><strong>Trevisan ’99</strong></td>
</tr>
</tbody>
</table>
Disperser viewed as graph

Disperser viewed as graph

Dispersion = good expansion

D has error probability < \( \varepsilon \).

\[ \uparrow \]

Every set \( S \) of size \( \varepsilon 2^R \) has \( |\Gamma(S)| > \frac{1}{2} 2^n \).

Threshold of disperser: \( \varepsilon 2^R \)

Proof

\( \mathbf{D} \) has error probability \( \geq \varepsilon \)

\[ \uparrow \]

\[ \exists T \subseteq \{0,1\}^n, \mu(T) \geq \frac{1}{2}, \Pr_{y}[\forall i: D(y,i) \notin T] \geq \varepsilon \]

\[ \downarrow \]

\[ \exists T \subseteq \{0,1\}^n, \mu(T) \geq \frac{1}{2}, \]

\[ S \subseteq \{0,1\}^R, \mu(S) \geq \varepsilon, \]

\[ T \cap \Gamma(S) = \emptyset \]

\[ \downarrow \]

\[ \exists S \subseteq \{0,1\}^R, \mu(S) \geq \varepsilon, |\Gamma(S)| \leq \frac{1}{2} \]

Dispersers

<table>
<thead>
<tr>
<th></th>
<th>Random bits ( R )</th>
<th>Probes ( m )</th>
<th>Error Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chor-Goldreich ‘86</td>
<td>( 2n )</td>
<td>( p(n) )</td>
<td>( \Omega(p(n)) )</td>
</tr>
<tr>
<td>Karp-Pippenger-Sipser ‘86</td>
<td>( n )</td>
<td>( p(n)^{O(1)} )</td>
<td>( \Omega(p(n)) )</td>
</tr>
<tr>
<td>Trevisan ‘99</td>
<td>( n^{O(1)} )</td>
<td>( n^{O(1)} )</td>
<td>( 2^{-R-R'} )</td>
</tr>
</tbody>
</table>

Karp-Pippenger-Sipser disperser

- Let \( G \) be a constant degree explicit expander graph on \( \{0,1\}^n \).

- \( D(x, s) = y \) for some \( s \) if and only if the distance between \( x \) and \( y \) in \( G \) is at most \( c \log n \).
### Dispersers

<table>
<thead>
<tr>
<th></th>
<th>Random bits ( R )</th>
<th>Probes ( m )</th>
<th>Error Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chor-Goldreich '86</td>
<td>( 2n )</td>
<td>( p(n) )</td>
<td>( 1/p(n) )</td>
</tr>
<tr>
<td>Karp-Pippenger-Sipser '86</td>
<td>( n )</td>
<td>( p(n)^{O(1)} )</td>
<td>( 1/p(n) )</td>
</tr>
<tr>
<td>Trevisan '99</td>
<td>( n^{O(1)} )</td>
<td>( n^{O(1)} )</td>
<td>( 2^{-n^{o(1)}} )</td>
</tr>
</tbody>
</table>

### A physical random bit generator

- How can we convert bits from a “dirty” random bit source to “pure” random bits?

### Randomness Purification

**Entropy:**

\[
H(X) = \sum_x -\Pr(X = x) \log \Pr(X = x)
\]

### Measure of randomness

- Whenever \( X \in \{0,1\}^n \) “contains much randomness”, \( E(X) \in \{0,1\}^n \) should be uniformly distributed.

### Example: Bit-fixing source

1. An adversary chooses at most \( n-k \) “bad” bit positions out of \( n \).
2. The \( k \) “good” bit positions are filled in randomly.
3. The “bad” bit positions are filled in by the adversary.
4. The resulting string is given as output without indicating which bits are good and which are bad.

\[
0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1
\]
Example: Bit-fixing source

1. An adversary chooses at most \( n : k \) “bad” bit positions out of \( n \).
2. The \( k \) “good” bit positions are filled in randomly.
3. The “bad” bit positions are filled in by the adversary.
4. The resulting string is given as output without indicating which bits are good and which are bad.

\[
\begin{array}{cccccccc}
0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
\end{array}
\]

This is a source of entropy at least \( k \).

Randomness Purifiers don’t exist!

Min-entropy

- \textbf{Reason 1:} Sources of high entropy may output 000000000...000 with high probability.
  \( X = 000000000...000 \) with probability \( \frac{1}{2} \).
  \( X = \text{uniform on} \ n \text{ bits with probability} \ \frac{1}{2} \).
  \( H(X) > n/2 \).
- \textbf{Solution:} Use a more restrictive notion of randomness-content: \textbf{Min-entropy}.

Randomness Purifiers don’t exist!

- \textbf{Reason 2:} Output of \( E \) cannot be uniform on \( \{0,1\}^n \) unless probabilities of outcomes of source can be divided into bags of combined measure \textbf{exactly} \( 2^n \).
- \textbf{Solution:} We shall only demand that the output is \textbf{close} to uniform.

Variation Distance

Let \( X_1 \) and \( X_2 \) be two random variable on the same domain \( V \). The \textbf{variation distance} or statistical distance between \( X_1 \) and \( X_2 \) is

\[
\text{dist}(X_1, X_2) = \max_{T \subseteq V} |\Pr[X_1 \in T] - \Pr[X_2 \in T]|.
\]

The \( T \)s are called \textbf{statistical tests}. 
Why Variation Distance?

\[ \Pr(\text{Uniform bits}) < \delta \]

Why Variation Distance?

\[ \Pr(\text{Purifier } E) < \delta + \varepsilon \]

Fact

Let \( f_1 : V \rightarrow [0,1] \), \( f_1(x) = \Pr[X_1 = x] \). Let \( f_2 : V \rightarrow [0,1] \), \( f_2(x) = \Pr[X_2 = x] \).

\[ \text{dist}(X_1, X_2) = \frac{1}{2} \| f_1 - f_2 \|_1 \]

Randomness Purifiers don’t exist!

**Reason 3:** Suppose \( E : \{0,1\}^k \rightarrow \{0,1\}^n \) is a purifier. Let \( y \in \{0,1\}^n \) maximize \( |E^i(y)| \). The uniform distribution on \( E^i(y) \) has min-entropy \( \log |E^i(y)| \geq \log 2^{R-n} = R-n \). But \( E \) is constant on this set!

**Solution:** \( E \) needs a catalyst of \( s \ll n \) pure random bits

Extractors

- An Extractor extracts randomness from a weak random source:
  \( E : \{0,1\}^k \times \{0,1\}^k \rightarrow \{0,1\}^n \)
- If \( X \) in \( \{0,1\}^k \) has min-entropy at least \( k \), \( U \) in \( \{0,1\}^k \) uniform and independent from \( X \), then \( E(X,U) \) is \( k \)-close to uniform on \( \{0,1\}^n \).
- \( k \) is the min-entropy threshold of \( E \) and \( n-k \) is the error of \( E \).

Fact

- For restricted classes of sources, the pure random bits are not necessary and we may have deterministic extractors.
- Example (Von Neuman): independent, identically distributed random bits.
Extractors

- An Extractor extracts randomness from a weak random source:
  \[ E : \{0,1\}^r \times \{0,1\}^s \rightarrow \{0,1\}^n \]
- If \( X \) in \( \{0,1\}^s \) has min-entropy at least \( k \),
  \( U \) in \( \{0,1\}^r \) uniform and independent from \( X \), then
  \( E(X,U) \) is \( k \)-close to uniform on \( \{0,1\}^n \).
- \( k \) is the min-entropy threshold of \( E \) and is the error of \( E \).

Dispersers

- A Disperser is an efficient algorithm encoding a particular strategy for finding hay:
  \[ D : \{0,1\}^r \times \{1,\ldots,m\} \rightarrow \{0,1\}^n \]
- Using \( R \) random bits \( y \), the disperser probes \( T(D(y,1)),\ldots,T(D(y,m)) \).
- The error probability of the disperser is
  \[ \max_{T,\mu(T)\geq 1/2} \Pr[\forall i : T(D(y,i)) = 0] \]

Extractors are Dispersers

- An extractor with error \( < \frac{1}{2} \) and min-entropy threshold \( k \) is a disperser with error probability \( < 2^{-k} \).

Proof:
- Suppose not.
- \( S \subseteq \{0,1\}^s \) of size \( 2^k \times 2^s = 2^s \) has \( |S| = \frac{1}{2} 2^s \).
- The uniform distribution on \( S \) has min-entropy \( k \).
- But \( E(S,U) \) only takes values in \( (S) \) and is hence \( 1/2 \)-far from uniform.

Monte Carlo Integration

\[
\text{volume}(A,m) \{ \\
\quad c := 0; \\
\quad \text{for (j=1; j<=m; j++)} \\
\quad \quad \text{if (random(U)\in A) c++;} \\
\quad \text{return c/m;}
\}
\]

Randomness-efficient Density Estimation

- Let \( m = 10 (1/ \log(1/)) \). With probability at least \( 1- \), \text{volume}(A,m) correctly
  estimates the volume of \( A \) within additive error.
- Can we achieve error probability \( \log(|U|) \log(1/ \) random
  bits? \textbf{Not answered by dispersers}....
Fact

- An extractor encodes a strategy for randomness-efficient density estimation (fairly easy).
- Conversely, any strategy for randomness efficient density estimation is an extractor (somewhat harder).
- As a consequence, the Chor-Goldreich disperser (pairwise independent random variables) is actually a (weak) extractor.

Dispersers

<table>
<thead>
<tr>
<th></th>
<th>Random bits $R$</th>
<th>Probes $m$</th>
<th>Error Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chor-Goldreich ’86</td>
<td>$2n$</td>
<td>$p(n)$</td>
<td>$1/p(n)$</td>
</tr>
<tr>
<td>Karp-Pippenger-Sipser ’86</td>
<td>$n$</td>
<td>$p(n)^{O(1)}$</td>
<td>$1/p(n)$</td>
</tr>
<tr>
<td>Trevisan ’99</td>
<td>$n^{O(1)}$</td>
<td>$n^{O(1)}$</td>
<td>$2^{-R+\epsilon'}$</td>
</tr>
</tbody>
</table>