

Universal methods for derandomization

Lecture 3 Hitting Set and Pseudorandom Generators

1

Convincing functional programmers about the truth of Hypothesis H

- Consider the algorithm that takes as input an **arbitrary** lambda or combinatory logic expression of length n , starts a clock, and does reductions (using a fixed strategy) until the clock shows 2^{100n}
- It outputs "true" if the computation manages to terminate, otherwise it outputs "false".
- Do you think this computation can be simulated using space $2^{n/100}$?
- If you think this is impossible, **then you also think that Hypothesis H is true and that universal derandomization is possible.**

2

Dispersers

- A **Disperser** is an efficient algorithm encoding a particular strategy for finding hay in a haystack:

$$D: \{0,1\}^R \times \{1,\dots,m\} \rightarrow \{0,1\}^n$$

- Using R random bits y , the disperser probes $T(D(y,1)), \dots, T(D(y,m))$.
- The error probability of the disperser is

$$\max_{T, \mu(T) \geq 1/2} \Pr_y [\forall i: T(D(y,i)) = 0]$$

3

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 $\{0,1\}^R$ has min-entropy at least k , U in $\{0,1\}^s$ uniform and independent from X , **then** $E(X,U)$ is ϵ -close to uniform on $\{0,1\}^n$.
- k is the **min-entropy threshold** of E and ϵ is the **error** of E .

4

Dispersers

	Random bits R	Probes m	Error Prob.
Chor-Goldreich '86	$2n$	$p(n)$	$1/p(n)$
Karp-Pippenger-Sipser '86	n	$p(n)^{O(1)}$	$1/p(n)$
Trevisan '99 Actually Extractor	$n^{O(1)}$	$n^{O(1)}$	2^{-R+R^ϵ}

5

Trevisan Extractor

$$E: \{0,1\}^R \times \{0,1\}^{O(\log n)} \rightarrow \{0,1\}^n$$

$R = n^k$ for any desired constant $k > 10$.

Min-entropy threshold n^3 .

Error $1/10$.

6

Importance of Trevisan Extractor

- When used for randomness efficient amplification of success probability of randomized algorithm, the error probability is 2^{-R+R^ϵ} .
- Amortized, *each* random bit used reduces the error probability by factor of $2^{-o(1)}$.
- Only 2^{R^ϵ} random choices out of 2^R are bad(!)

7

Hypothesis H

There exists polynomial procedure **findHay** taking as input a circuit $C: \{0,1\}^n \rightarrow \{0,1\}$ so that

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

8

Equivalent to H (Tuesday)

There exists polynomial procedure **findHay** taking as input a circuit $C: \{0,1\}^n \rightarrow \{0,1\}$ so that

1. **findHay**(C) is in $\{0,1\}^n$.
2. If $\mu(C) \geq 1 - 2^{-n^{1-\epsilon}}$ then $C(\mathbf{findHay}(C))=1$.

9

In other words,

It is sufficient to be able to find hay in a haystack of size 2^n where the number of needles is at most

$$2^{n-n^{1-\epsilon}}$$

10

Equivalent to H (now)

There exists polynomial procedure **findHay** taking as input a circuit $C: \{0,1\}^n \rightarrow \{0,1\}$ so that

1. **findHay**(C) is in $\{0,1\}^n$.
2. If $\mu(C) \geq 1 - 2^{-n+n^\epsilon}$ then $C(\mathbf{findHay}(C))=1$.

11

In other words,

It is sufficient to be able to find hay in a haystack of size 2^n where the number of needles is at most

$$2^{n^\epsilon}$$

12

Proof

- Assume we can find hay in haystacks with at most 2^{n^e} needles.
- Given a circuit C with $\mu(C) \geq 1/2$, construct $C': \{0,1\}^R \rightarrow \{0,1\}$ with

$$C'(x) = 1 \Leftrightarrow \exists i : C(\mathbf{E}(x, i)) = 1$$

where \mathbf{E} is the Trevisan Extractor

13

Main open problem

Construct for arbitrary parameters n, t ,

$\mathbf{E}: \{0,1\}^{n+t} \times \{0,1\}^{O(\log(n+t))} \rightarrow \{0,1\}^n$
with min-entropy threshold n .

In terms of amplification: Error probability 2^{-t} using $n+t$ random bits.

State of the art (Lu-Reingold-Vadhan-Wigderson 2003):
As above, but min-entropy threshold $1.01 n$.

Practicality would be nice..... Get constant in Big-O down...

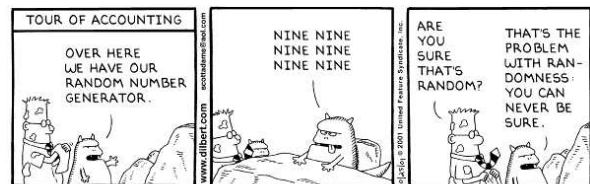
14

Complete derandomization

- To derandomize a randomized algorithm, replace the random bits used by the algorithm by the output of a **pseudorandom generator**.
- In reality, almost all implemented randomized algorithms have already been derandomized in this way in practice!! So are we wasting our time?

15

The Paradox of Pseudorandomness



Copyright © 2001 United Feature Syndicate, Inc.

16

A Universal Task:

Finding Hay in a Haystack
(Circuit version)

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

We have seen obvious – and not so obvious – randomized algorithms for this task. What would a good pseudorandom generator for these look like?

17

Hitting Sets

A **Hitting Set** is a subset S of $\{0,1\}^n$ so that for all circuits C of size n with $\mu(C) \geq 1/2$,

$$\exists x \in S: C(x)=1$$

18

Existence of Hitting Sets

- Consider potential hitting (multi)sets S of size m .
- #Sets = $(2^n)^m$.
- #Sets that fail to hit a *particular* circuit = $(2^{n-1})^m$.
- #Circuits of size $n \leq (3(n+n)^2)^n$.
- #Actual hitting sets $\geq (2^n)^m - (3(n+n)^2)^n (2^{n-1})^m$.

Hitting sets of size $m = 10 n \log n$ exists!

19

Hitting set generators

- A **Hitting Set Generator** is an algorithm that on input n runs in time polynomial in n and outputs a subset of $\{0,1\}^n$ that is a hitting set.
- If a Hitting Set Generator exists, then hypothesis H is true.

20

Pseudorandom Sets

A **Pseudorandom Set** with error ϵ is a subset S of $\{0,1\}^n$ so that for all circuits C of size n ,

$$\left| \frac{|\{x \in S : C(x) = 1\}|}{|S|} - \mu(C) \right| \leq \epsilon$$

21

Variation Distance

Let X_1 and X_2 be two random variables on the same domain V . The **variation distance** or statistical distance between X_1 and X_2 is

$$\max_{T \subseteq V} |\Pr[X_1 \in T] - \Pr[X_2 \in T]|$$

22

Computational Distance

Let X_1 and X_2 be two random variables on $\{0,1\}^n$. The **computational distance** between X_1 and X_2 is

$$\max_{C \text{ of size } n} |\Pr[C(X_1) = 1] - \Pr[C(X_2) = 1]|$$

Computational distance is a metric and always smaller than or equal to variation distance.

23

Pseudorandom Sets, Alternative Definition

A **Pseudorandom Set** with error ϵ is a subset S of $\{0,1\}^n$ so that the uniform distribution on S is ϵ -close to uniform in computational distance.

24

Pseudorandom Generators

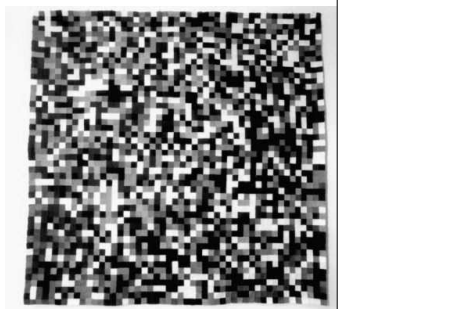
- A Pseudorandom Generator is an algorithm that on input n runs in time polynomial in n and outputs a subset of $\{0,1\}^n$ that is a pseudorandom set.
- Pseudorandom sets are also hitting sets, so pseudorandom generators are also hitting set generators.
- A Pseudorandom Generator can be used to **directly** derandomize Monte Carlo density estimation.

25

PRGs vs. Crypto-PRGs

- Our generators don't use seeds!
- We know the exact computational power of our adversary in advance and can use more time than he.
- What is a good candidate for our kind of generator?

26



50 × 50 decimal digits of

27

Google Search: Patterns in Digits of pi - Mozilla Firefox

Web Images Groups News Groups Local more

Patterns in Digits of pi

The following words are very common and were not included in your search: In of: [\[details\]](#)

Web Results 1 - 10 of about 69,100 for Patterns in Digits of pi @ 0.25 seconds

112 pieces of Pi
 ... Pi with Gillian, Garth, and Zaamen, my math group from Tracy's class at Metropolitan Learning Center, I started to generate images with the digits of Pi ...
[www.ameripress.com/MatlabPI/PIk_Cached - Similar pages](#)

Frequency of Each Digit of Pi
 ... It has no repeating pattern, yet not all digits are equally likely. It has been conjectured that pi is "normal" which, mathematically speaking, means that all ...
[www.ersanderson.com/piprecalculated-frequencies-12k - Cached - Similar pages](#)

Links to Pi Pages
 ... Pi Patterns #1. Mark Conahan creates patterns based on the digits of pi. Pi Patterns #2. Mike Keith shows how to decode pi to see a picture ...
[www.joyofpi.com/pilinks.html - 22k - Cached - Similar pages](#)

The problem with looking for patterns in pi@Everything2.com
 ... Anything. Hence, there are no "patterns" in pi ... Note: The assertion that the digits of pi are normal is widely conjectured but unproven ...
[www.everything2.com/index.pl?mode=view&id=2095 - 24k - Cached - Similar pages](#)

We are in the Digits of Pi and Live Forever
 ... of digits that could be said to encode me, but do any of them exist in PIP? A number can fall to repeat and still not contain all possible finite patterns ...
[sprout.physics.wisc.edu/pickover/pimains.html - 45k - Cached - Similar pages](#)

The Randomness of Pi: The Frequency of Digits and Patterns ...
 ...and Randomness of π seen. Is Pi Normal ... variations in the frequency of its digits page

Lady Pi - Mozilla Firefox

http://users.acl.com/~st57/g/ladypi.htm

The result is shown below.

Figure 2: Lady pi walks in the Szaalit Garden.

After drawing 50 revolutions of the binary-pi spiral, we have colored in three areas using a flood fill. Vivaldi - it's Lady Pi. She has a face with two eyes (and, well, yes, kind of an odd-shaped mouth), hair streaming off to her left, a long flowing robe, and - most dramatic - her right hand is

Find: SETI Find Next Find Previous Highlight Match case

Is there a pattern in the digits of π or are they "random"?

- Question pondered by science fiction writers, amateur scientists, numerologists... and other mysticists.
- Example: **Carl Sagan: Contact.**
- There is at least *one* way in which the digits of π are not "random":
Whenever you compute π , those are the digits you get.... Is this the "only" pattern?

30

The π generator

- On input n , output the first 100 n^3 binary digits of π as 100 n^2 strings, each of length n .
- Hypothesis H : The π generator is a hitting set generator.
- Hypothesis H implies Hypothesis H .

31

A Win-Win Situation

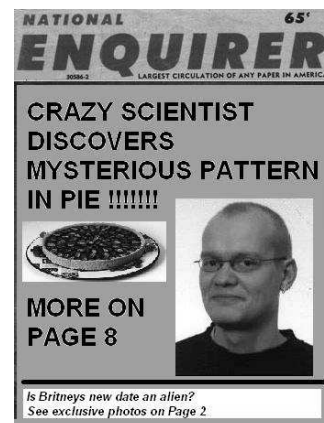
- From now on, I'm going to assume Hypothesis π and always use the π generator to find hay.
- **Case 1:** I'll never run into any problems. **GREAT!**
- **Case 2:** One day we fail to find hay in a haystack, so the π generator is not a hitting set generator after all. **STILL GREAT(!)**

32

Why is this still great?

- Suppose the generator fails, say, for $n=100$.
- I have found a **simple** property (described by a circuit of size 100) that is satisfied by half of all 100-bit strings but which **none** of the first 1000000 blocks of 100 bits of π satisfies.
- **AMAZING!** This is what all the amateur scientists were looking for and **much** more convincing than the "patterns" they found.

33



34

Philosophical interpretation

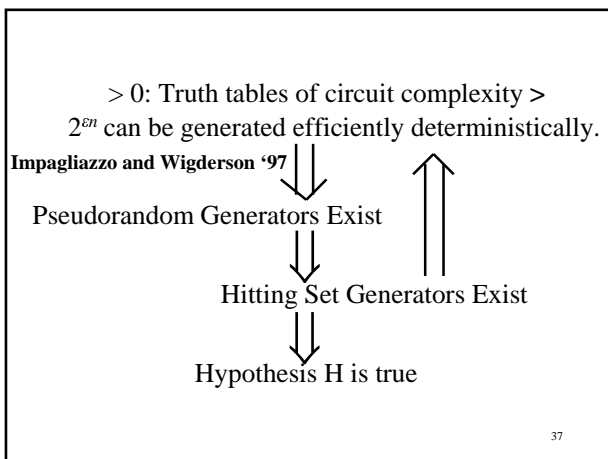
- Hypothesis H is just one way in which we may establish Hypothesis H .
- If Hypothesis H is false, there are amazing patterns to be found **everywhere** – not just in π . How great!
- Thus I tend to think H is true.....

35

Hitting set generators derandomize the construction of hard truth tables

- Let $S \subseteq \{0,1\}^n$ be a hitting set of size n^k .
- Let $m = k \log n + 1$ and $f: \{0,1\}^m \rightarrow \{0,1\}$, $f(x)=1$ iff x is *not* a prefix of an element of S .
- $\mu(f) \geq 1/2$.
- Let $g: \{0,1\}^n \rightarrow \{0,1\}$, $g(x)=1$ iff $f(x_{1..m})=1$.
- $\mu(g) = \mu(f) \geq 1/2$.
- S does not hit g , so g does not have circuits of size $n \Rightarrow f$ does not have circuits of size $n = 2^{m/2k}$.

36



Generation vs. Computation

- $\exists \epsilon > 0$: Truth tables of circuit complexity $> 2^{\epsilon n}$ can be **generated** efficiently deterministically

↕

- There is a language in **E** so that any circuit family **computing** it must have size $2^{\Omega(n)}$

38

Hardness-based Hitting Set generators

- A **Hardness-based Hitting Set generator** is an efficient algorithm converting a hard truth table into a hitting set.

$\mathbf{G}: \{0,1\}^R \times \{1,\dots,m\} \rightarrow \{0,1\}^n$

- When y is a truth table of circuit complexity at least k , the set $\mathbf{G}(y,1), \dots, \mathbf{G}(y,m)$ is a hitting set.
- k is called the **hardness-threshold** of the generator.

39

Dispersers

- A **Disperser** is an efficient algorithm encoding a particular strategy for finding hay:

$\mathbf{D}: \{0,1\}^R \times \{1,\dots,m\} \rightarrow \{0,1\}^n$

- Using R random bits y , the disperser probes $T(\mathbf{D}(y,1)), \dots, T(\mathbf{D}(y,m))$.
- The error probability of the disperser is $\max_{T, \mu(T) \geq 1/2} \Pr_y[\forall i: T(\mathbf{D}(y,i))=0]$

40

- Hardness-based hitting set generators are dispersers.
- Hardness-based pseudorandom generators are extractors.

41

Weak Hitting Set Generators

- An (q, ϵ) -**Weak Hitting Set Generator** is an efficient algorithm that on input n runs in time polynomial in n and outputs a sequence S so that for all circuits C of size n^q that output 0 on at most $2^{\epsilon n}$ input vectors, some x in S has $C(x)=1$.

42

Weak Hitting Set Generators

- If weak hitting set generators exist, Hypothesis H holds.
- Actually, if weak hitting set generators exist, hitting set generators exist:
- If S is a weak hitting set and E is Trevisan's extractor, $E(S,*)$ is a hitting set.

43

- Trevisan's disperser makes it much easier to build hitting set generators as we only need to build a weak one.
- (Strong) Hardness-based hitting set generators *are* dispersers, so we would have to build a disperser anyway.
- Similarly for extractors and pseudorandom generators.

44

Incompressibility of computation

$\exists k > 0$ and a deterministic algorithm A running in time $t = 2^{kn}$ so that any algorithm using space $\ell^{0.999}$ fails to simulate A on all sufficiently large input lengths.



Hypothesis H is true

45

A weak hitting set generator

- Split the memory of A into blocks of length n bits.
- Simulate A on all inputs of length $c \log n$.
- The hitting set is the set of all bit patterns occurring as a block at some time during one of these computations.

46

Incompressibility of computation

$\exists k > 0$ and a deterministic algorithm A running in time $t = 2^{kn}$ so that any algorithm using space $\ell^{0.999}$ fails to simulate A on all sufficiently large input lengths.



Hypothesis H is true

47

Analysis

- If what I generate is *not* a weak hitting set, there is a small circuit C mapping n bits to 1 bit, with at most 2^{n^c} out of 2^n inputs mapping to 0, but *all* elements of my set map to 0.
- But then the computation of A is compressible after all! Reason: I have found a mysterious pattern that all n -bit blocks that will occur during my computation have.
- I store each n bit block in my memory as a binary number i of n^c bits, so that the i th input (in lexicographic order) that makes C evaluate to 0 is my block!

48

Incompressibility of computation

$\exists k > 0$ and a deterministic algorithm A running in time $t = 2^{kn}$ so that any algorithm using space $\ell^{0.999}$ fails to simulate A on all sufficiently large input lengths.



Hypothesis H is true

49

Stronger version

$\exists k > 0, \epsilon > 0$ and a deterministic algorithm A running in time $t = 2^{kn}$ so that any algorithm using space ℓ^ϵ fails to simulate A on all sufficiently large input lengths.



Hypothesis H is true

50

Disclaimer

- These lectures have focused mainly on definitions and have barely scratched the surface of the field of derandomization.

51