Open Problems in Unconditional Derandomization

Luca Trevisan

University of California, Berkeley
Stanford University

Derandomization: efficient deterministic simulation of randomized algorithms

goals

Suppose $A(x, r)$ is an "efficient" randomized algorithm with input x and randomness r

Derandomization of decision problems Given the promise that

either

$$\mathbb{P}_r[A(x, r) = 1] \geq \frac{9}{10}$$

or

$$\mathbb{P}_r[A(x, r) = 1] \leq \frac{1}{10} \ ,$$

determine which is the case

goals

Suppose $A(x, r)$ is an "efficient" randomized algorithm with input x and randomness r

Derandomization of decision problems Given the promise that

either

$$\mathbb{P}_r[A(x, r) = 1] \geq \frac{9}{10}$$

or

$$\mathbb{P}_r[A(x, r) = 1] \leq \frac{1}{10} \ ,$$

determine which is the case

Note: If *efficient* = polynomial time, this implies $P = BPP$

goals

Suppose $A(x, r)$ is an "efficient" randomized algorithm with input x and randomness r

Derandomization of search problems Given the promise that

$$\mathbb{P}_r[A(x, r) = 1] \geq \frac{1}{10}$$

find r^* such that $A(x, r^*) = 1$

Note: If *efficient* = polynomial time, this derandomizes randomized search algorithms

goals

Suppose $A(x, r)$ is an "efficient" randomized algorithm with input x and randomness r

Approximate counting Find a number p such that

$$p - \frac{1}{10} \leq \mathbb{P}_r[A(x, r) = 1] \leq p + \frac{1}{10}$$

Note: If *efficient* = polynomial time, this derandomizes randomized approximate counting algorithms such as Permanent approximation

goals

Hitting Set Generation Find a set S such that, for every "efficient" randomized algorithm $A(x, r)$ and every input x, if

$$\mathbb{P}_r[A(x, r) = 1] \geq \frac{1}{10}$$

then

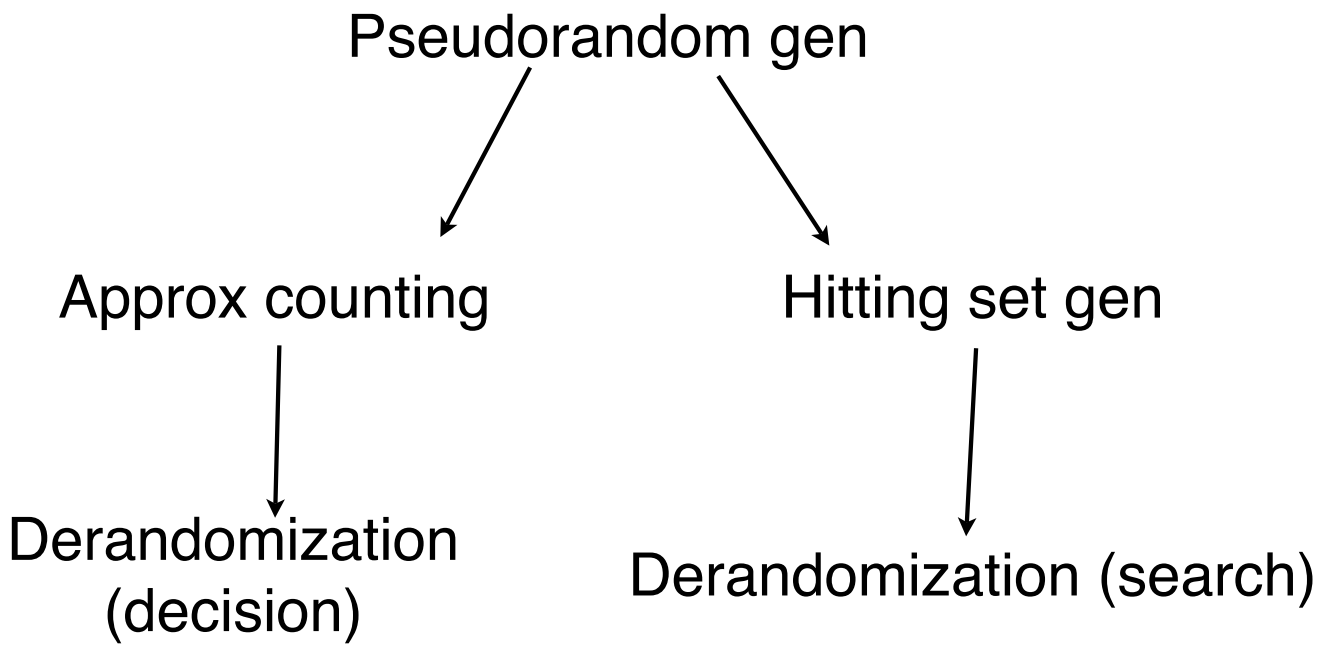
$$\exists r^* \in S.\ A(x, r^*) = 1$$

goals

Pseudorandom Generation Find a set S such that, for every "efficient" randomized algorithm $A(x, r)$ and every input x,

$$\mathbb{P}_{r \sim S}[A(x, r) = 1] - \frac{1}{10} \leq \mathbb{P}_{r \sim U}[A(x, r) = 1] \leq \mathbb{P}_{r \sim S}[A(x, r) = 1] + \frac{1}{10}$$

relationships



complexity classes

In general: for a class of algorithms, useful to focus on class of functions $\mathcal{C}$ of form $r \longrightarrow A(x, r)$ over choice of algorithm A and input x.
e.g.

- A polynomial time, $\mathcal{C}$ polynomial size circuits
- A logarithmic space, $\mathcal{C}$ polynomial width branching programs

More convenient to define derandomization, approximate counting, hitting set generation, pseudorandom generation in terms of $\mathcal{C}$.

conditional derandomization

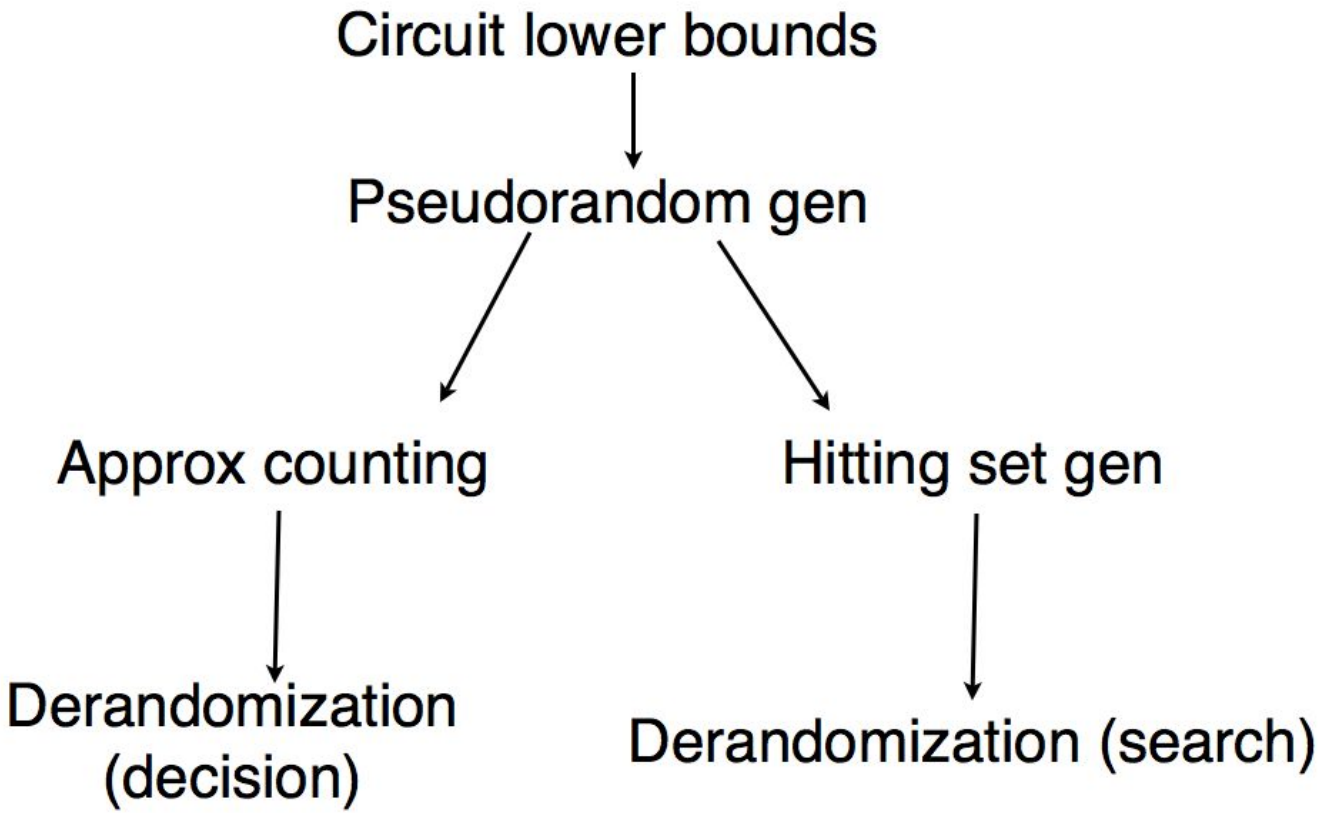
[Nisan-Wigderson, Babai-Fortnow-Nisan-Wigderson, Impagliazzo, Impagliazzo-Wigderson]

Under plausible circuit complexity assumptions,

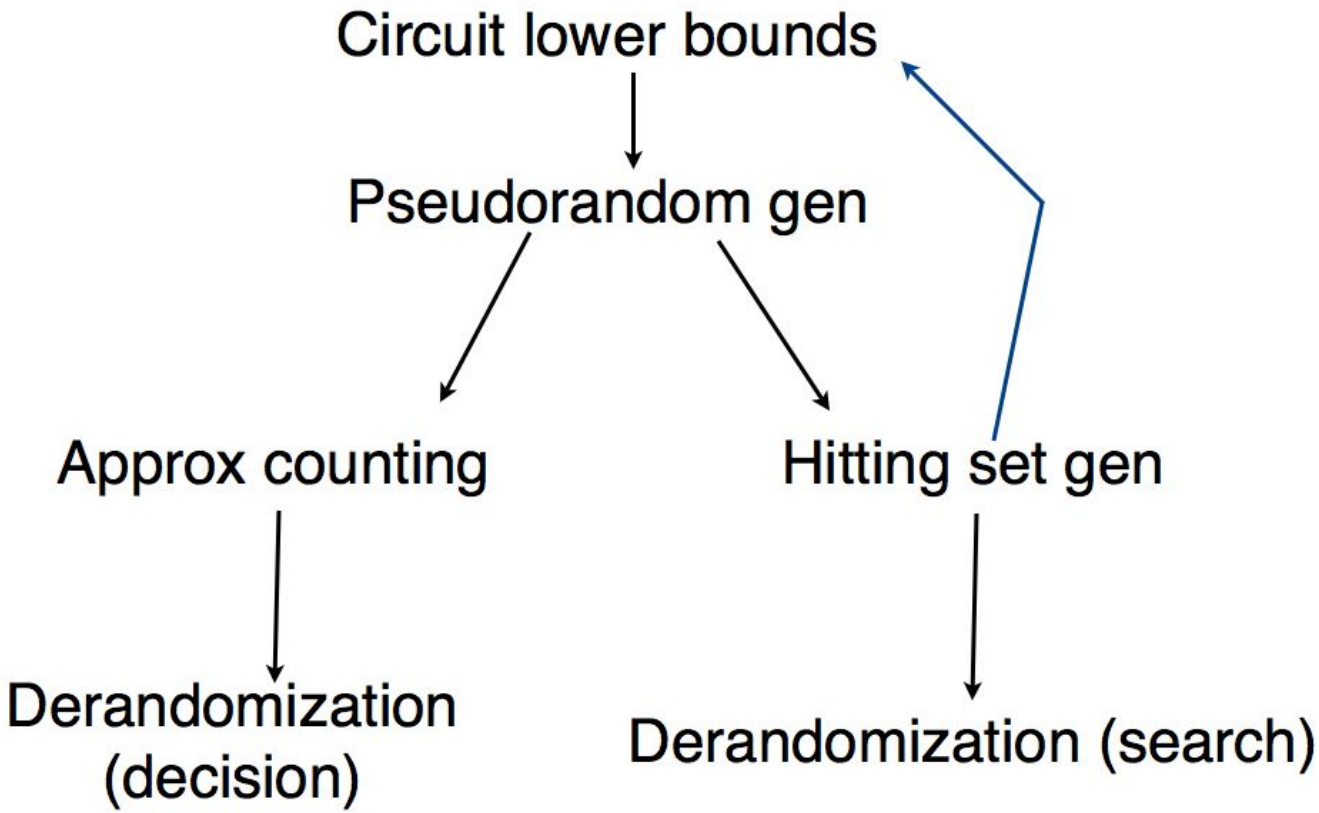
there are polynomial time computable pseudorandom generators for polynomial size circuits (and $P = BPP$, etc.)

and log-space computable pseudorandom generators for polynomial size branching programs (and $L = BPL$, etc.)

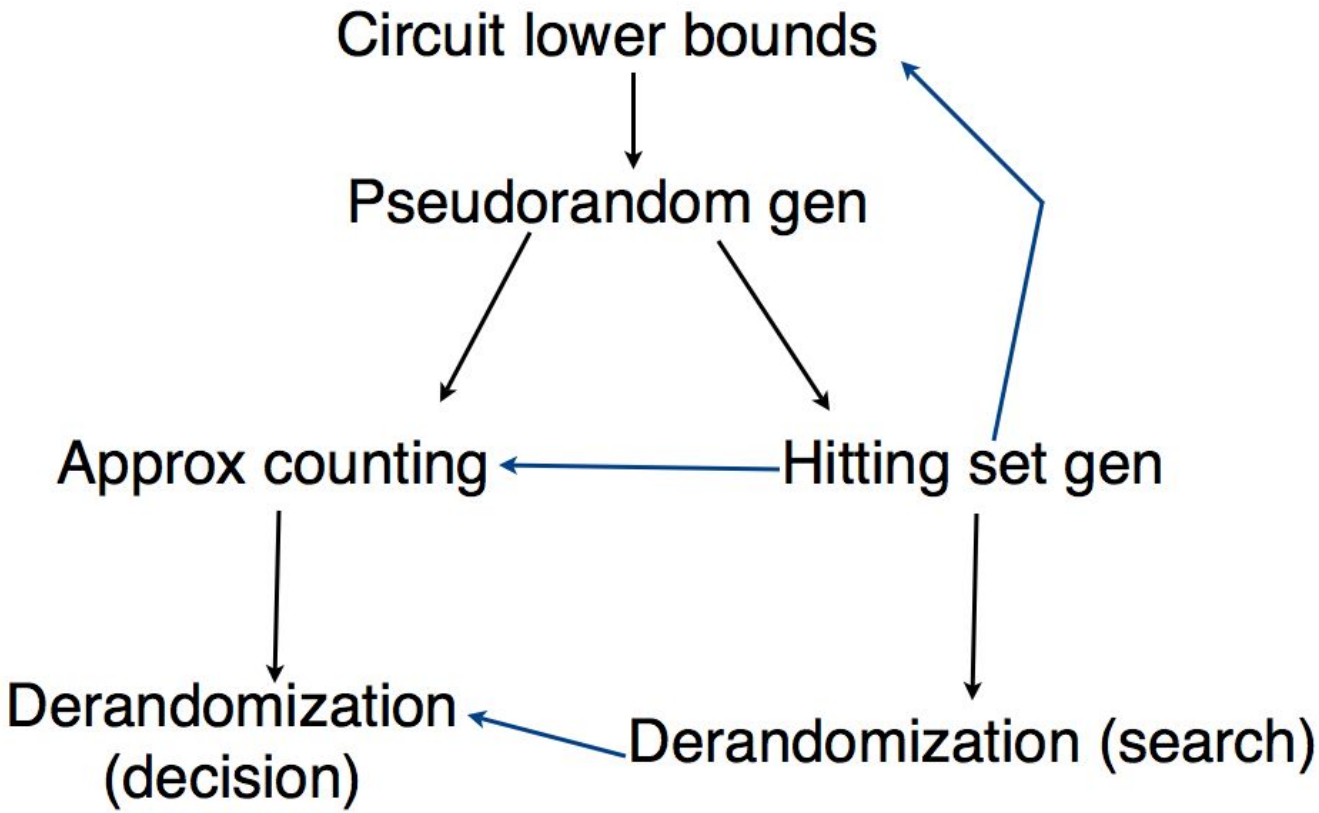
polynomial size circuits



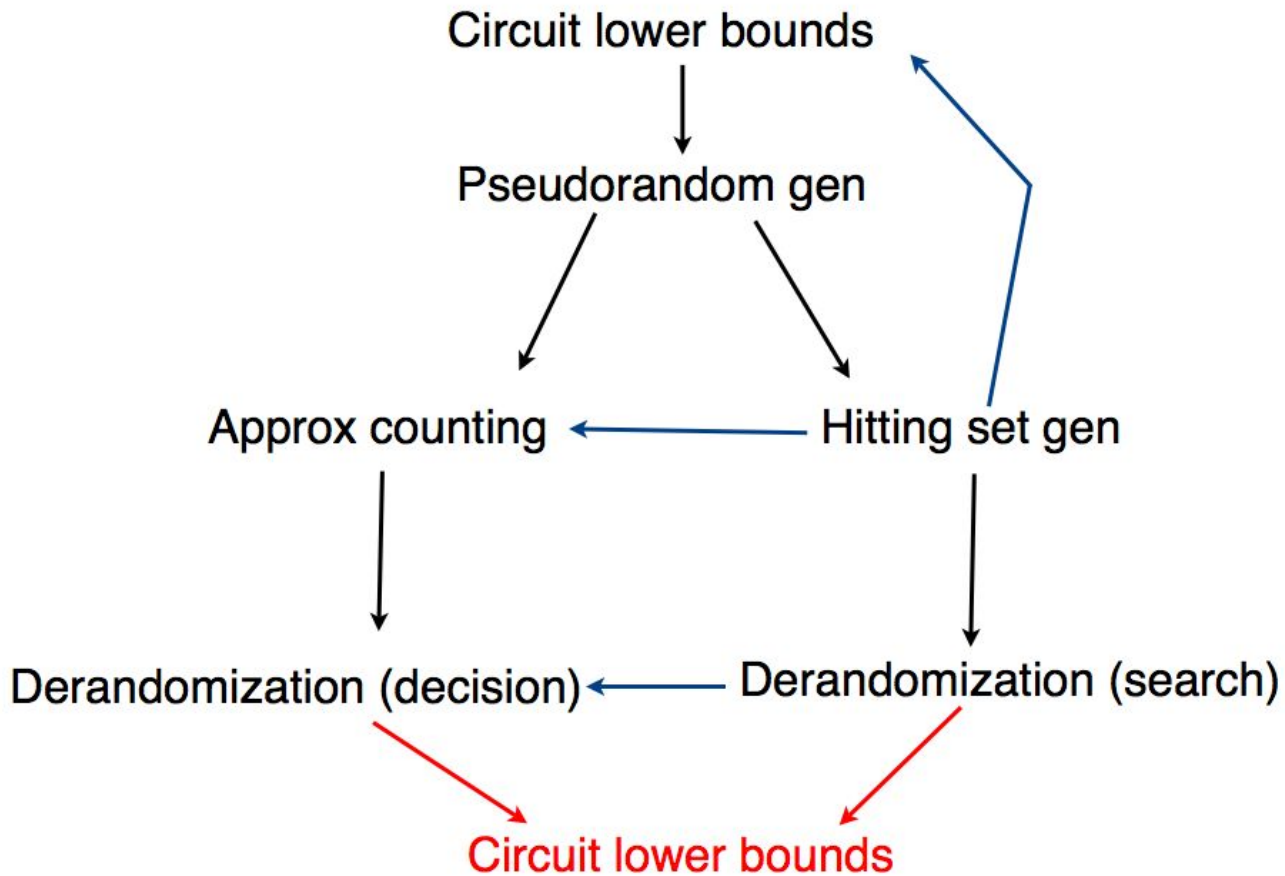
polynomial size circuits



polynomial size circuits



polynomial size circuits



conditional derandomization

[..., Kabanets-Impagliazzo, ...]
Circuit lower bound assumptions are necessary to construct pseudorandom generators, and even for search or decision derandomization.

unconditional derandomization

- A pseudorandom generator with $n^{poly \log n}$ set size for bounded-depth circuits
  [Nisan, Nisan-Wigderson, 1989]

- A pseudorandom generator with $n^{\log n}$ set size for polynomial width branching programs
  [Nisan 1990]

unconditional derandomization

- A pseudorandom generator with $n^{poly \log n}$ set size for bounded-depth circuits
  [Nisan, Nisan-Wigderson, 1989]

- A pseudorandom generator with $n^{\log n}$ set size for polynomial width branching programs
  [Nisan 1990]

In past twenty years: some exciting developments, but main questions still open

Bounded Depth Circuits

Nisan's generator

[Nisan, Nisan-Wigderson '88]

Hardness vs. Randonness: Parity is hard for bounded-depth circuits [Furst-Saxe-Sipser, Yao, Håstad '81-'86]; construct pseudorandom generator that is as hard to break as it is hard to compute parity; generator cannot be broken by poly size circuits

Result: Pseudorandom set of size $n^{O(\log^{2d+5} n)}$ for depth-d circuits, $n^{\log^9 n}$ for depth-2; optimized to $n^{\log^3 n}$ [Luby-Velickovic-Wigderson '93]

question 1

Hardness of parity against depth-2 circuits: every dept-2 circuit of size $2^{o(\sqrt{n})}$ has agreement at most $\frac{1}{2} + \frac{1}{2^{\Omega(\sqrt{n})}}$ with Parity.

question 1

Hardness of parity against depth-2 circuits: every dept-2 circuit of size $2^{o(\sqrt{n})}$ has agreement at most $\frac{1}{2} + \frac{1}{2^{\Omega(\sqrt{n})}}$ with Parity.

Best possible result for symmetric function

question 1

Hardness of parity against depth-2 circuits: every dept-2 circuit of size $2^{o(\sqrt{n})}$ has agreement at most $\frac{1}{2} + \frac{1}{2^{\Omega(\sqrt{n})}}$ with Parity.

Best possible result for symmetric function

Open: is there an explicit function $f : \{0,1\}^n \to \{0,1\}$ (e.g. computable in EXP) such that every depth-2 circuit of size $2^{o(n)}$ has agreement at most $\frac{1}{2} + \frac{1}{2^{\Omega(n)}}$ with f?

Note: not sufficient to construct optimal Pseudorandom Generators via Nisan-Wigderson, but important first step

Linial-Nisan

Conjecture: every $(\log^{O(d)} n)$-wise independent distribution is pseudorandom for depth-d circuits

Linial-Nisan

Conjecture: every $(\log^{O(d)} n)$-wise independent distribution is pseudorandom for depth-d circuits

Bazzi: every $O(\log^2 n)$-wise independent distribution is pseudorandom for depth-2 circuits
Gives $n^{\log^2 n}$ size pseudorandom set (better than via Nisan-Wigderson)

Linial-Nisan

Conjecture: every $(\log^{O(d)} n)$-wise independent distribution is pseudorandom for depth-d circuits

Bazzi: every $O(\log^2 n)$-wise independent distribution is pseudorandom for depth-2 circuits
Gives $n^{\log^2 n}$ size pseudorandom set (better than via Nisan-Wigderson)

Braverman: every $(\log^{O(d^2)} n)$-wise independent distribution is pseudorandom for depth-d circuits

Linial-Nisan

Conjecture: every $(\log^{O(d)} n)$-wise independent distribution is pseudorandom for depth-d circuits

Bazzi: every $O(\log^2 n)$-wise independent distribution is pseudorandom for depth-2 circuits
Gives $n^{\log^2 n}$ size pseudorandom set (better than via Nisan-Wigderson)

Braverman: every $(\log^{O(d^2)} n)$-wise independent distribution is pseudorandom for depth-d circuits

De-Etesami-T-Tulsiani: every $n^{-\tilde{O}(\log n)}$-biased distribution is pseudorandom for depth-2 circuits
Gives $n^{\tilde{O}(\log n)}$ size pseudorandom set

question 2

Is it true that every $\frac{1}{n^{O(1)}}$-biased distribution is $\frac{1}{10}$-pseudorandom for depth-2 circuits?

True for read-once [DETT '10] and read-k [Klivans-Lee-Wan '10]

False if one wants $\frac{1}{n}$-pseudorandomness

approximate counting

Given a depth-2 circuit C, an algorithm of Luby and Velickovic (1993) runs in time

$$n^{2^{O(\sqrt{\log \log n})}}$$

approximate counting

Given a depth-2 circuit C, an algorithm of Luby and Velickovic (1993) runs in time

$$n^{2^{O(\sqrt{\log \log n})}}$$

(that's $n^{o(\log n)}$)

approximate counting

Given a depth-2 circuit C, an algorithm of Luby and Velickovic (1993) runs in time

$$n^{2^{O(\sqrt{\log \log n})}}$$

(that's $n^{o(\log n)}$)

and computes a number that is $\mathbb{P}[C(x) = 1] \pm \frac{1}{10}$

approximate counting

Given a depth-2 circuit C, an algorithm of Luby and Velickovic (1993) runs in time

$$n^{2^{O(\sqrt{\log \log n})}}$$

(that's $n^{o(\log n)}$)

and computes a number that is $\mathbb{P}[C(x) = 1] \pm \frac{1}{10}$

As far as I can see, the algorithm does not give a way to find a satisfying assignment under the promise that $\mathbb{P}[C(x) = 1] \geq \frac{1}{10}$

question 3

Develop a search algorithm with the Luby-Velickovic $n^{2^{O(\sqrt{\log \log n})}}$ running time

Why should it be possible?

question 3

Develop a search algorithm with the Luby-Velickovic $n^{2^{O(\sqrt{\log \log n})}}$ running time

Why should it be possible?

Suppose ϕ is $\frac{1}{10}$-satisfiable CNF

question 3

Develop a search algorithm with the Luby-Velickovic $n^{2^{O(\sqrt{\log \log n})}}$ running time

Why should it be possible?

Suppose ϕ is $\frac{1}{10}$-satisfiable CNF

Luby-Velickovic algorithm (with error set at 1%) outputs an approximation of $\mathbb{P}[\phi(x) = 1]$ which is $\geq .09$

question 3

Develop a search algorithm with the Luby-Velickovic $n^{2^{O(\sqrt{\log \log n})}}$ running time

Why should it be possible?

Suppose ϕ is $\frac{1}{10}$-satisfiable CNF

Luby-Velickovic algorithm (with error set at 1%) outputs an approximation of $\mathbb{P}[\phi(x) = 1]$ which is $\geq .09$

This certifies that $\mathbb{P}[\phi(x) = 1] \geq .08 > 0$

question 3

Develop a search algorithm with the Luby-Velickovic $n^{2^{O(\sqrt{\log \log n})}}$ running time

Why should it be possible?

Suppose ϕ is $\frac{1}{10}$-satisfiable CNF

Luby-Velickovic algorithm (with error set at 1%) outputs an approximation of $\mathbb{P}[\phi(x) = 1]$ which is $\geq .09$

This certifies that $\mathbb{P}[\phi(x) = 1] \geq .08 > 0$

How come this certificate does not yield a satisfying assignment?

depth-2 summary

Pseudorandomness: set of size $n^{\tilde{O}(\log n)}$ via small-bias distributions [DETT '10]

Hitting Sets: see Pseudorandomness

Approximate Counting: running time $n^{2^{O(\sqrt{\log \log n})}} \leq n^{o(\log n)}$ [Luby-Velickovic '93]

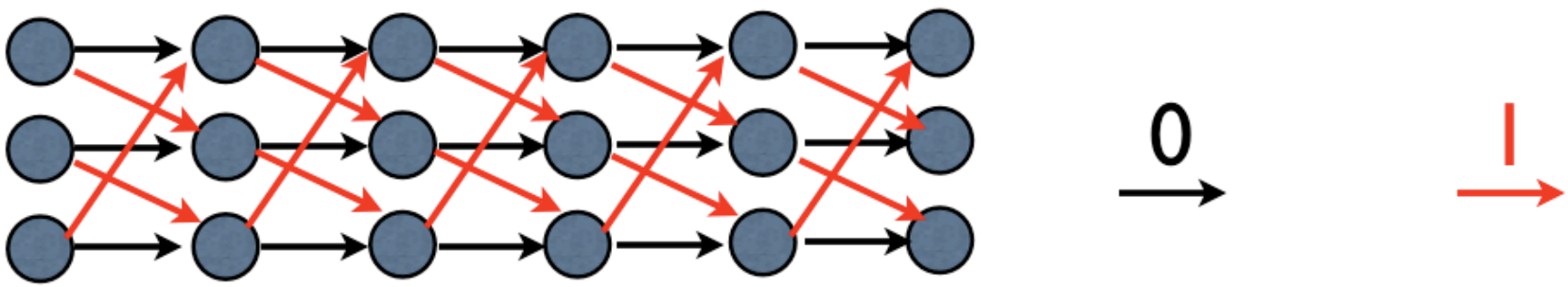
Derandomization (search): ??

Derandomization (decision): see Approximate Counting

Bounded-width Branching Programs

the model of width-w branching programs

A layered graph with w nodes in each layer.
Each node has two outgoing edges to next layer, labeled 0 and 1.
Models computations with $\log_2 w$ bits of memory



poly(n) width

Models log-space computations

Pseudorandomness: $n^{\log n}$ size set [Nisan '90]

Hitting Sets: see pseudorandomness

Approximate counting: $n^{\sqrt{\log n}}$ [Saks-Zhou '95]

Derandomization (search): ??

Derandomization (decison): see approximate counting

poly(n) width

Models log-space computations

Pseudorandomness: $n^{\log n}$ size set [Nisan '90]

Hitting Sets: see pseudorandomness

Approximate counting: $n^{\sqrt{\log n}}$ [Saks-Zhou '95]

Derandomization (search): ??

Derandomization (decison): see approximate counting

Reingold's breakthrough applies to "undirected" branching programs

$O(1)$ width

Most promising set-up to improve Nisan's generator

$O(1)$ width

Most promising set-up to improve Nisan's generator

Note:

- Approximate counting and derandomization trivial in $O(1)$-width case

$O(1)$ width

Most promising set-up to improve Nisan's generator

Note:

- Approximate counting and derandomization trivial in $O(1)$-width case
- Program:
  1. improve Nisan's generator in $O(1)$-width case;

$O(1)$ width

Most promising set-up to improve Nisan's generator

Note:

- Approximate counting and derandomization trivial in $O(1)$-width case
- Program:
  1. improve Nisan's generator in $O(1)$-width case;
  2. transfer improvement to poly(n) width;

$O(1)$ width

Most promising set-up to improve Nisan's generator

Note:

- Approximate counting and derandomization trivial in $O(1)$-width case
- Program:
  1. improve Nisan's generator in $O(1)$-width case;
  2. transfer improvement to poly(n) width;
  3. used improved generator in Saks-Zhou to improve derandomization.

open question 4

Pseudorandom generators:

1. Width-1: 0 bits of memory, trivial setting
2. Width-2: 1 bit of memory
   $\frac{1}{n^{O(1)}}$-biased distribution give poly(n)-size pseudorandom set [Saks-Zuckerman '99]
3. Width-3: better than Nisan's?

width-3 and width-4 branching programs

Some bottlenecks:

- Small bias distributions are not pseudorandom for width-3 branching programs

  Counterexample: uniform distribution over bit strings whose number of ones is a multiple of 3

- Width-4 branching programs can compute read-once polynomials mod 2 and mod 3

  Open to construct optimal pseudorandom sets for read-once polynomials

hitting sets

Recently [Sima-Zak '10] give a $n^{O(1)}$-size hitting set construction for width-3 branching programs, for sufficiently large constant probability parameter.

Construction: start from the support X of a $\frac{1}{n^{O(1)}}$-biased distribution, and consider set of strings of hamming distance at most 2 from X

a sample of questions

- Given a CNF formula, and promise that a 1% fraction of assignments satisfy it, find such an assignment in polynomial time

  Sufficient to look at support of a $\frac{1}{n^{O(1)}}$-biased distribution?

a sample of questions

- Given a CNF formula, and promise that a 1% fraction of assignments satisfy it, find such an assignment in polynomial time

  Sufficient to look at support of a $\frac{1}{n^{O(1)}}$-biased distribution?

- Construct an ϵ-Hitting Set Generator of polynomial size of every ϵ of polynomial size

  Sima-Zak construction works for all constant $\epsilon > 0$?

a sample of questions

- Given a CNF formula, and promise that a 1% fraction of assignments satisfy it, find such an assignment in polynomial time

  Sufficient to look at support of a $\frac{1}{n^{O(1)}}$-biased distribution?

- Construct an ϵ-Hitting Set Generator of polynomial size of every ϵ of polynomial size

  Sima-Zak construction works for all constant $\epsilon > 0$?

- Improve Nisan's pseudorandom generator for width-3 branching programs

  XOR of two small-bias distributions?