Superfast Derandomization from Very Hard Functions

Lijie Chen and Roei Tell

Harvard CS Theory Seminar, March 2021

In this talk

› the setting

1. Background
2. Our results
3. A taste of techniques

1 Background

simple and fast derandomization

Randomness in computation

› context

› We need randomness

 › crypto, learning, sublinear-time algs…

› Conjecture: We don’t need randomness to efficiently

1. solve decision problems
2. solve “verifiable” search problems

BPP $\stackrel{?}{=}$ P

› historic recap

› BPP formally defined in [Gill'77]

› Immediately conjectured to "sort-of" equal P

> We believe that for the unrelativized classes of Turing machines, only speedups for infinitely many inputs can be achieved by probabilistic machines.

BPP $\stackrel{?}{=}$ P

› historic recap

› … in fact, paper even raises stronger conjecture:

<u>Conjecture:</u> If f is a recursive function computed in time T* by some probabilistic Turing machine with error probability bounded away from 1/2 , then there is a deterministic Turing machine which computes f in time O(T*(x)) for infinitely many x.

Hardness-to-randomness

› more recent history

- Hard functions ⇒ efficient pseudorandomness

  [Yao,'82, BM'84]

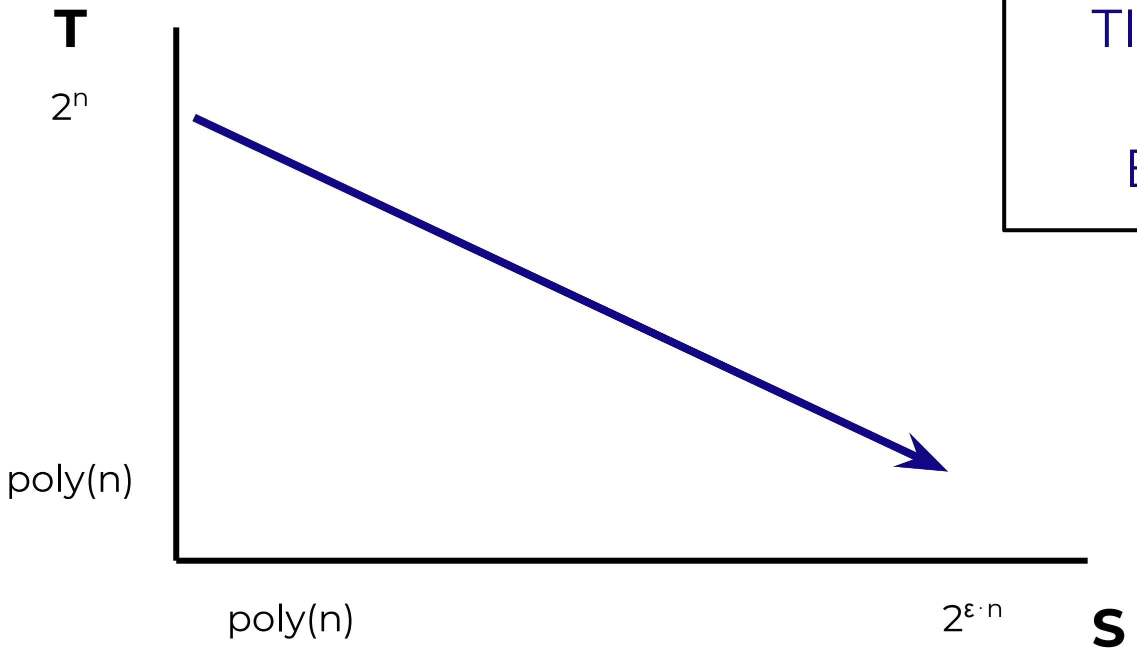
  ⇒ derandomization of BPP

  [NW'94, IW'99, STV'01, SU'01, Uma'03, and others]

- Conditioned on lower bounds, we have an answer

Hardness-to-randomness

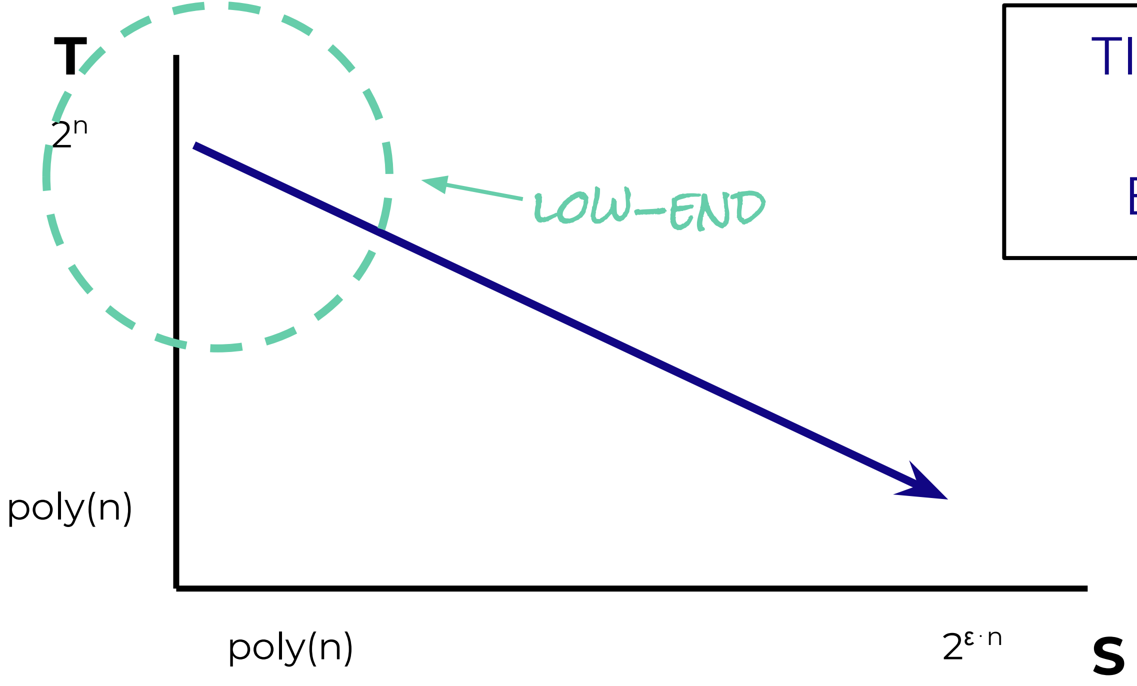
› more recent history



$$\mathrm{TIME}[2^n] \not\subset \mathrm{SIZE}[S]$$
$$\Rightarrow$$
$$\mathrm{BPP} \subseteq \mathrm{TIME}[T]$$

Hardness-to-randomness

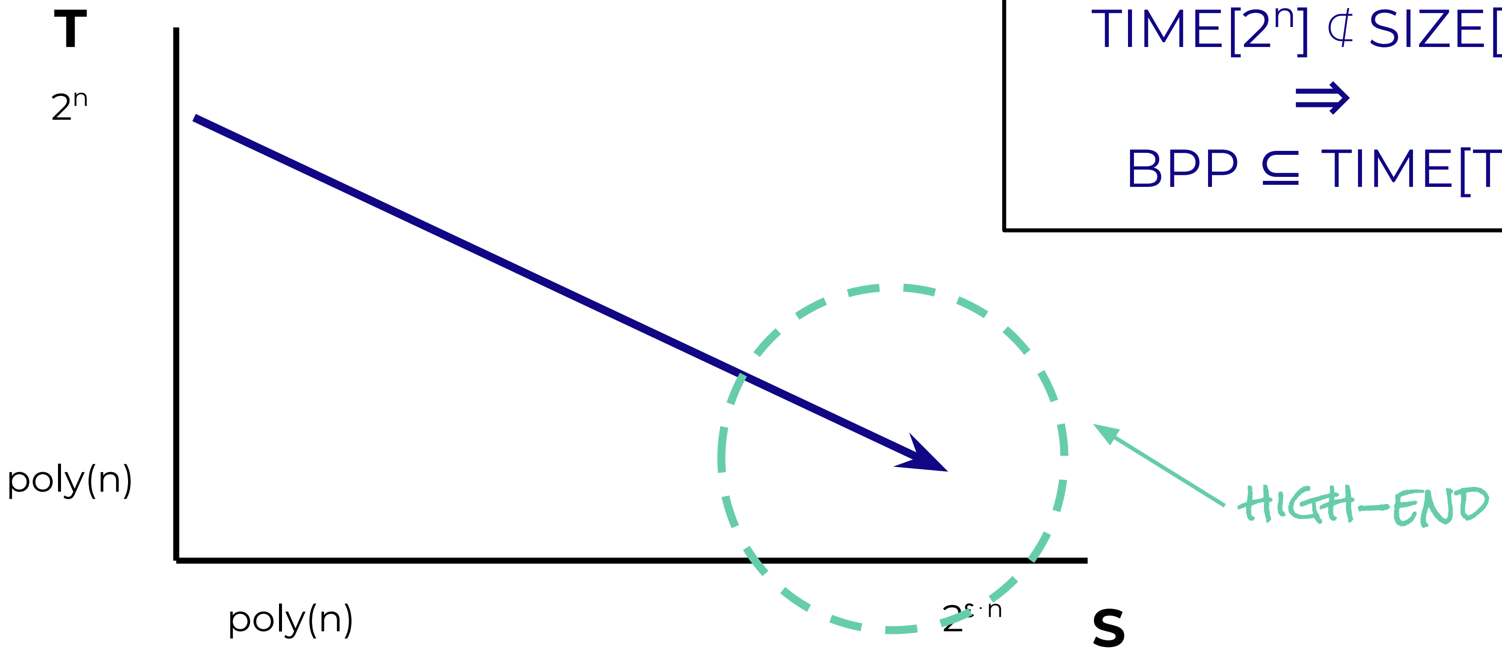
- more recent history



$$\mathrm{TIME}[2^n] \not\subset \mathrm{SIZE}[S]$$
$$\Rightarrow$$
$$\mathrm{BPP} \subseteq \mathrm{TIME}[T]$$

Hardness-to-randomness

› more recent history



$$\mathrm{TIME}[2^n] \not\subset \mathrm{SIZE}[S]$$
$$\Rightarrow$$
$$\mathrm{BPP} \subseteq \mathrm{TIME}[T]$$

Hardness-to-randomness

› more recent history

› Smooth trade-off from [Uma'03] gives $T \approx 2^{O(\,S\wedge\text{-}1(n)\,)}$

› Extremal point established in [IW'99]:

$$\mathbf{TIME[\ 2^n\ ] \not\subset {}_{io}SIZE[\ 2^{.01n}\ ]}$$

$$\Rightarrow \quad \mathbf{BPP = P}$$

$$\Rightarrow \quad \mathbf{BPTIME[\ T\ ] \subseteq TIME[\ T^{O(1)}\ ]}$$

Superfast derandomization

- … snap back to now

- Doron, Moshkovitz, Oh, and Zuckerman (FOCS 2020) recently asked: Can we do it faster?

  $\mathrm{BPTIME}[\,T\,] \subseteq \mathrm{TIME}[\,T^c\,]$ for a small c?

- Classical results can yield “reasonable” c when scaled-up
- Diff between (say) c = 10 and c = 3 is substantial !

Superfast derandomization

- … snap back to now

- What is the actual cost of simulating randomness?
  - new area to explore
  - theoretical basis not formed yet
- The obvious “end-goal” question:

Can we simulate randomness with no cost?

Superfast derandomization

- … snap back to now

  - Main result of [DMOZ'20]:

    $$\mathbf{BPTIME[\ T\ ] \subseteq TIME[\ T^{2.01}\ ]}$$

    conditioned on

    $$\mathbf{TIME[\ 2^{n}\ ] \not\subset {}_{io}MASIZE[\ 2^{.99n}\ ]}$$

Superfast derandomization

- ... snap back to now
  - Main result of [DMOZ'20]:

$$\mathbf{BPTIME[\ T\ ] \subseteq TIME[\ T^{2.01}\ ]}$$

QUADRATIC OVERHEAD MIGHT BE POSSIBLE

conditioned on

$$\mathbf{TIME[\ 2^{n}\ ] \not\subset ioMASIZE[\ 2^{.99n}\ ]}$$

Superfast derandomization

- … snap back to now

  - Main result of [DMOZ'20]:

$$\mathbf{BPTIME[\ T\ ] \subseteq TIME[\ T^{2.01}\ ]}$$

conditioned on

$$\mathbf{TIME[\ 2^{n}\ ] \not\subset {}_{io}MASIZE[\ 2^{.99n}\ ]}$$

HYPOTHESIS SEEMS
"TOO STRONG"

Superfast derandomization

› ... snap back to now

	hardness	derand. overhead	
[IW'99]	TIME[2^n] $\not\subset$ ioSIZE[$2^{.01n}$]	$T^{O(1)}$	
[DMOZ'20]	TIME[2^n] $\not\subset$ ioMASIZE[$2^{.99n}$]	$T^{2.01}$	

Superfast derandomization

- … snap back to now

- Takeaways:
  1. Superfast derand possible, under assumptions!
  2. Can we do better than quadratic overhead?
  3. We need stronger theoretical foundations
     - hypothesis seems “too strong” & a bit non-standard

2 Our results

simple and fast derandomization

Preliminary peek at our results

	hardness	derand. overhead	
[IW'99]	TIME[2^n] $\not\subset$ ioSIZE[$2^{.01n}$]	$T^{O(1)}$	
[DMOZ'20]	TIME[2^n] $\not\subset$ ioMASIZE[$2^{.99n}$]	$T^{2.01}$	
this work	(see next)		

Preliminary peek at our results

	hardness	derand. overhead	
[IW'99]	$\mathrm{TIME}[2^n] \not\subset$ $\mathrm{ioSIZE}[2^{.01n}]$	$T^{O(1)}$	
[DMOZ'20]	$\mathrm{TIME}[2^n] \not\subset$ $\mathrm{ioMASIZE}[2^{.99n}]$	$T^{2.01}$	
this work	(see next)	$\mathbf{n \cdot T^{1.01}}$	

Preliminary peek at our results

	hardness	derand. overhead	
[IW'99]	TIME[2^n] $\not\subset$ ioSIZE[$2^{.01n}$]	$T^{O(1)}$	
[DMOZ'20]	TIME[2^n] $\not\subset$ ioMASIZE[$2^{.99n}$]	$T^{2.01}$	
this work	(see next)	$\mathbf{n \cdot T^{1.01}}$	
		$\mathbf{n^{1.01} \cdot T}$	$T \leq 2^{n \wedge o(1)}$

Preliminary peek at our results

	hardness	derand. overhead	
[IW'99]	TIME[2^n] $\not\subset$ ioSIZE[$2^{.01n}$]	$T^{O(1)}$	
[DMOZ'20]	TIME[2^n] $\not\subset$ ioMASIZE[$2^{.99n}$]	$T^{2.01}$	
this work	(see next)	$\mathbf{n \cdot T^{1.01}}$	
		$\mathbf{n^{1.01} \cdot T}$	$T \leq 2^{n^{o(1)}}$
		$\mathbf{n^{0.01} \cdot T}$	T = poly(n) on average

Preliminary peek at our results

	hardness	derand. overhead	
[IW'99]	$\mathrm{TIME}[\,2^n\,] \not\subset$ $\mathrm{ioSIZE}[\,2^{.01n}\,]$	$T^{O(1)}$	
[DMOZ'20]	$\mathrm{TIME}[\,2^n\,] \not\subset$ $\mathrm{ioMASIZE}[\,2^{.99n}\,]$	$T^{2.01}$	
this work	**one-way funcs & non-uniformly strong time hierarchy**	$\mathbf{n \cdot T^{1.01}}$	
		$\mathbf{n^{1.01} \cdot T}$	$T \leq 2^{n^{\wedge}o(1)}$
		$\mathbf{n^{0.01} \cdot T}$	$T = \mathrm{poly}(n)$ on average

Non-uniformly-strong time hierarchy

- our first main result

  - Classical hypotheses:

    TIME[2^n] hard for ioSIZE[$2^{.01 \cdot n}$] [IW'99]

Non-uniformly-strong time hierarchy

- our first main result

- Classical hypotheses:

  $\mathrm{TIME}[2^n]$ hard for $_{\mathrm{io}}\mathrm{SIZE}[2^{.01 \cdot n}]$ [IW'99]

NON-UNIFORMITY ISN'T ENOUGH
TO SPEED UP ALL ALGORITHMS

Non-uniformly-strong time hierarchy

- our first main result

- Classical hypotheses:

$\mathrm{TIME}[\,2^n\,]$ hard for $\mathrm{ioSIZE}[\,2^{.01 \cdot n}\,]$ [IW'99]

(NO "MAGIC SPEEDUP ADVICE" TAILORED TO INPUT LENGTH)

Non-uniformly-strong time hierarchy

- our first main result

- Classical hypotheses:

  $\mathrm{TIME}[2^{n}]$ hard for ${}_{\mathrm{io}}\mathrm{SIZE}[2^{.01 \cdot n}]$ [IW'99]

- Our hypotheses:

  $\mathrm{TIME}[2^{kn}]$ hard for ${}_{\mathrm{io}}\mathrm{TIME}[2^{.99kn}] / 2^{.99n}$

Non-uniformly-strong time hierarchy

› our first main result

› Classical hypotheses:

$\mathrm{TIME}[2^{n}]$ hard for $\mathrm{ioSIZE}[2^{.01 \cdot n}]$ [IW'99]

› Our hypotheses:

$\mathrm{TIME}[2^{kn}]$ hard for $\mathrm{ioTIME}[2^{.99kn}] / 2^{.99n}$

LARGE TIME

NEAR-MAXIMAL ADVICE

Non-uniformly-strong time hierarchy

› our first main result

› Classical hypotheses:

$\mathrm{TIME}[2^n]$ hard for ${}_{\mathrm{io}}\mathrm{SIZE}[2^{.01 \cdot n}]$ [IW'99]

› Our hypotheses:

$\mathrm{TIME}[2^{kn}]$ hard for ${}_{\mathrm{io}}\mathrm{TIME}[2^{.99kn}] / 2^{.99n}$

› Natural scale-up of classical hypotheses

Near-linear time derandomization

- our first main result

- Thm 1: Assume non-uniformly secure OWFs. Then,

  $\forall \varepsilon>0\ \exists \delta>0$ st $\forall T\ \exists k=k_T=O(1/\varepsilon)$ for which

  $$\mathbf{BPTIME[\,T\,] \subseteq TIME[\,n \cdot T^{1+\varepsilon}\,]}$$

  conditioned on

  $$\mathbf{TIME[\,2^{kn}\,] \not\subseteq {}_{io}TIME[\,2^{(k-\delta)\cdot n}\,] / 2^{(1-\delta)\cdot n}}$$

Additional properties of the result

- understanding superfast derandomizatoin

- The hardness assumption is necessary (when using PRGs)
  - … hypothesis is optimal for “black-box” techs up to OWFs
- Proof of Thm 1 is intuitive and technically non-involved
  - combining new insights with known technical tools

Complementing the first result

› zooming-in on the precise overhead

› Thm 2: Assume subexp-non-uniformly secure OWFs. Then,

$\forall \varepsilon>0 \quad \exists \delta, k=O(1/\varepsilon) \quad$ st $\quad \forall$ time $T \leq 2^{o(n)}$

$$\mathbf{BPTIME[\, T\,] \subseteq TIME[\, n^{1+\varepsilon} \cdot T\,]}$$

conditioned on

$$\mathbf{TIME[\, 2^{\delta \cdot n} \cdot T'\,] \not\subset {}_{io}TIME[\, T'\,]/2^{(1-\delta)\cdot n}}$$

where $T'(n) = T(2^{(1-\delta)\cdot n}) \cdot 2^{O(\delta \cdot n)}$

Complementing the first result

- zooming-in on the precise overhead

  - Thm 2 (reminder): Under assumptions…

$$\mathbf{BPTIME}[\,T\,] \subseteq \mathbf{TIME}[\,n^{1+\varepsilon} \cdot T\,]$$

Complementing the first result

› zooming-in on the precise overhead

› <u>Thm 2 (reminder):</u> Under assumptions...

$$\mathbf{BPTIME[\,T\,] \subseteq TIME[\,n^{1+\varepsilon} \cdot T\,]}$$

DO WE HAVE TO PAY " × N"?

(TEXTBOOK BPP ⊆ P/POLY
HAS THIS OVERHEAD)

Complementing the first result

- zooming-in on the precise overhead

- Thm 2 (reminder): Under assumptions…

  $$\mathbf{BPTIME[\,T\,] \subseteq TIME[\,n^{1+\varepsilon} \cdot T\,]}$$

- Prop 3: Conditioned on #NSETH, $\forall\,\varepsilon>0$

  $$\mathbf{BPTIME[\,T\,] \not\subseteq TIME[\,n^{1-\varepsilon} \cdot T\,]} \quad (\forall\ T = \text{poly})$$

- #NSETH: We can't count solutions of a given k-SAT formula in NTIME[$2^{(1-\varepsilon)\cdot n}$] (assuming suff. large $k=k_\varepsilon$)

Average-case derandomization

› bypassing this barrier

› Thm 4: Assume non-uniformly secure OWFs. Then,

$\forall \varepsilon>0 \quad \exists \delta, k=O(1/\varepsilon) \quad \text{st} \quad \forall$ time $T(n)=\mathrm{poly}(n)$

$$\mathbf{BPTIME[\,T\,] \subseteq TIME[\,n^{\varepsilon} \cdot T\,]\ on\ average}$$

conditioned on

$$\mathbf{TIME[\,2^{\delta \cdot n} \cdot T'\,] \not\subset {}_{io}TIME[\,T'\,] / 2^{(1-\delta)\cdot n}}$$

where $T'(n) \approx T(2^{(1-\delta)\cdot O(1/\varepsilon)\cdot n}) \cdot 2^{O(\delta \cdot n)}$

Average-case derandomization

- bypassing this barrier

- Thm 4: Under assumptions …

$$\mathbf{BPTIME[\ T\ ] \subseteq TIME[\ n^{\varepsilon} \cdot T\ ]\ on\ average}$$

- … with respect to all T-time samplable distributions
- … with success probability $1\text{-}n^{-\omega(1)}$
  - $L \in \text{BPTIME}[T] \Rightarrow$ one alg A_L “looks correct” to all T-time dist.

Extra goodies in the paper

› technical insights & results intertwined in our proofs

1. Easy way to bypass a formidable-looking barrier
2. Simplify & extend [DMOZ'20]: Derandomization with overhead $c \in \{1,2,3,4\} + \varepsilon$ from corresponding assump.
3. General simplification of a well-known PRG paradigm
   - "extract-from-pseudoentropic string" as a special case of an easy-to-analyze strategy
   - new light on avoiding the hybrid argument
4. Batch-computable PRGs vs amortized time-complexity

Meaning of our results

› zooming out

› Takeaways:

1. Derandomization with near-linear overhead is possible, under natural assumptions
2. Hypotheses are different than in [DMOZ'20] and support trade-offs with conclusion
3. Broadening the emerging theoretical basis for superfast derandomization

Near-linear time derandomization

- in a world of BPTIME[T] ≈ TIME[T]

- Randomness might be nearly useless
  - time overhead is minor
  - derandomization is simple & solves search problems
- Derandomize “better-than-brute-force” algorithms
- Lower bds for DTIME ⇒ lower bds for BPTIME
  - SETH ⇒ rSETH (assuming Thm 1 for arbitrarily small savings)

3 A taste of techniques

observations & proof sketches

Technical roadmap

› what we'll talk about

- Bypassing the seed-length barrier
- Proof sketch for Thm 1
- Simplifying a well-known PRG paradigm

Bypassing the seed-length barrier

one technical observation to remember

Hardness-to-randomness

- derandomization from PRGs

distinguisher

PRG

G: $\ell(n)$ ⟹ $T(n)$

Hardness-to-randomness

› derandomization from PRGs

- replace $T(n)$ coins with $\ell(n)$ coins, enumerate in time $2^{\ell(n)}$
- textbook results [NW'94,IW'99,STV'01,SU'01,Uma'03]:

$\mathrm{DTIME}[2^n] \not\subset$ SIZE[s] $\Rightarrow$ **PRG with stretch ≈ s** $\Rightarrow$ **BPP in DTIME ≈ $2^{O(s^{-1}(n))}$**

A formidable-looking barrier

- why experts might think that $c<2$ requires "new techniques"

  - Textbook approach: To derandomize time-T algs, construct a PRG that fools non-uniform size-T circuits
  - Such a PRG requires a seed of length $\log(T)$
  - The derandomization time is $2^{\log(T)} \cdot T(n) \geq T(n)^2$

Tracking the non-uniformity

› modeling distinguishers, carefully

who is this distinguisher?

T(n)

Tracking the non-uniformity

› modeling distinguishers, carefully

› For any L ∈ BPTIME[T], our focus is:

Does the probabilistic machine M_L behave the same on $\mathbf{G^f(u_{\ell(n)})}$ & $\mathbf{u_{T(n)}}$ for all inputs x?

› Distinguisher is M_L with an arbitrary fixed input x

T(n)

Tracking the non-uniformity

› modeling distinguishers, carefully

- Textbook distinguisher:

  Non-uniform circuit of size T

- Our pivotal observation:

  Distinguisher is a time-T machine with $|x| = n \ll T$ bits of non-uniformity

T(n)

Why is this helpful?

- fooling small non-uniformity with small seed length

- non-uniformity is $n \ll T(n)$
  - we want to fool TIME[T]/n rather than SIZE[T]
- $\exists$ non-explicit PRG with seed length $\log(n) \ll \log(T)$!
- opens the door to derandomization in time $n \cdot T(n)$
  - we'll make this PRG explicit, under assumptions

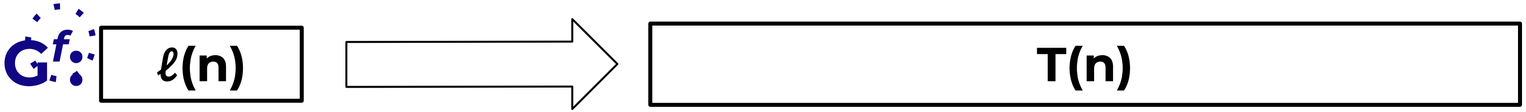
Proof sketch for Thm 1

main ideas & some parameters

Hardness-to-randomness

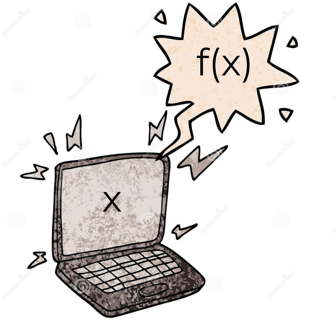
› reconstructive PRGs

PRG based on hard function f



Hardness-to-randomness

› reconstructive PRGs



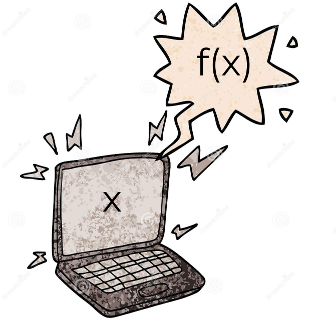
distinguisher
yields efficient
procedure for f

G^f: $\boxed{\ell(n)}$ ⟹ $\boxed{T(n)}$

Hardness-to-randomness

- reconstructive PRGs

"RECONSTRUCTION" OF F

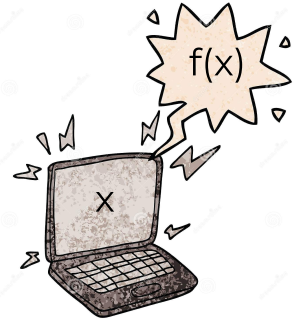


distinguisher
yields efficient
procedure for f

G^f: $\boxed{\ell(n)}$ ⟹ $\boxed{T(n)}$

Hardness-to-randomness

- reconstructive PRGs



distinguisher
yields efficient
procedure for f

G^f: $\boxed{\ell(n)} \Rightarrow \boxed{T(n)}$

F HARD ⇒ NO EFFICIENT DISTINGUISHER

Reconstruction overhead

› and its discontents

› Reconstruction overhead is the main bottleneck

› Inefficient reconstruction

$\Rightarrow$ inefficient procedure for f

$\Rightarrow$ stronger hardness hypothesis

Reconstruction overhead

› and its discontents

- Best known overhead [Uma'03]:

  distinguisher in time $T \Rightarrow$ procedure for f in time $T^{O(1)}$

  - … so we need to assume f is hard for time $T^{O(1)}$
  - … since the PRG computes $f \Rightarrow$ PRG takes time $\geq T^{O(1)}$
- Derandomization with large polynomial overhead

Our PRG construction

› high-level overview

› Our goal is to avoid this overhead

› Two ideas in the proof:

1. Compose “low-cost” PRGs
2. Use a tiny & super-exponentially-hard truth-table

Our PRG construction

› high-level overview

› Our goal is to avoid this overhead

› Two ideas in the proof:

1. Compose "low-cost" PRGs

2. Use a tiny & super-exponentially-hard truth-table

COMPUTABLE IN TIME $T^{1.01}$, LOW-OVERHEAD RECONSTRUCTION

Composing two “low-cost” PRGs

- each computable in time $\approx T^{1.01}$

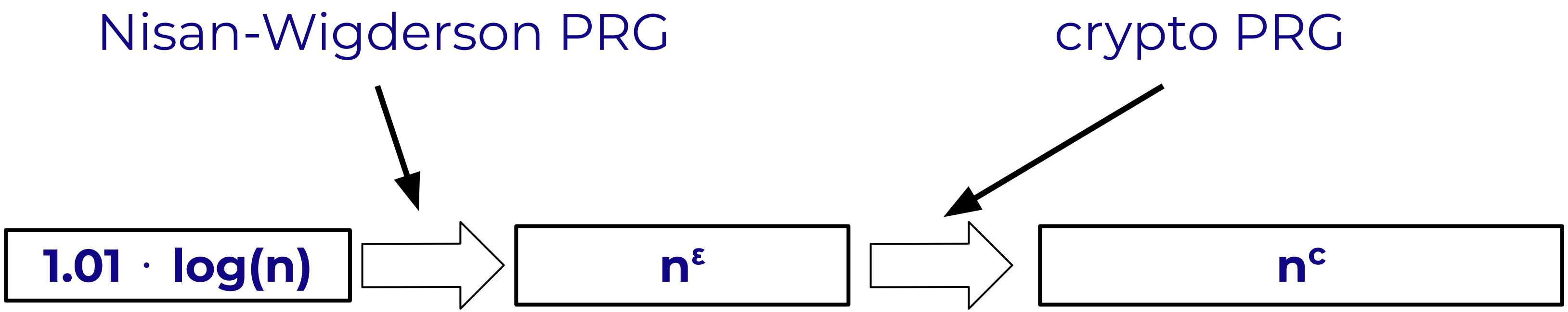
- Focus on $T(n) = n^c$ for simplicity

$$\boxed{1.01 \cdot \log(n)} \Rightarrow \boxed{n^{\varepsilon}} \Rightarrow \boxed{n^{c}}$$

Composing two "low-cost" PRGs

- each computable in time $\approx T^{1.01}$

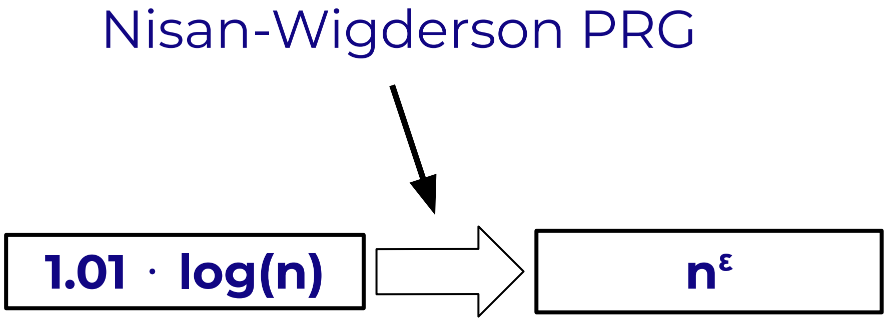
  - Focus on $T(n) = n^c$ for simplicity



Composing two “low-cost” PRGs

› the “inner” PRG

- Small seed, but small output length
- <u>Obs:</u> Small output length ⇒ small reconst. overhead



Composing two "low-cost" PRGs

› the "inner" PRG

› Small seed, but small output length

› <u>Obs:</u> Small output length ⇒ small reconst. overhead

Nisan-Wigderson PRG

(INCLUDING A CODE FOR HARDNESS AMPLIFICATION)

$\boxed{\mathbf{1.01 \cdot \log(n)}} \Rightarrow \boxed{\mathbf{n^{\varepsilon}}}$

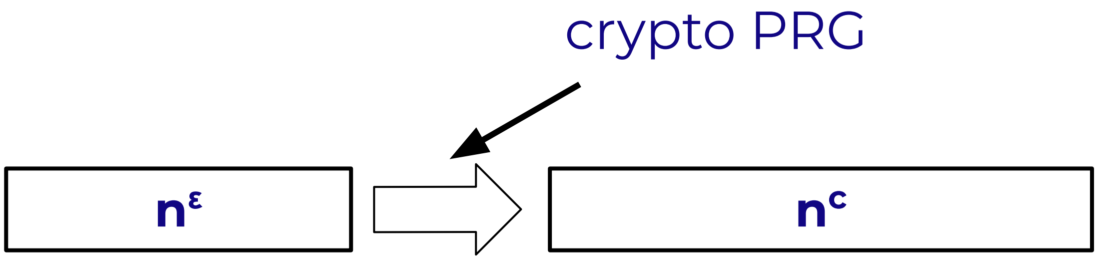
Composing two "low-cost" PRGs

› the "outer" PRG

› Large output length, but large seed

› Obs: OWF ⇒ crypto PRG ⇒ near-linear time PRG[1]

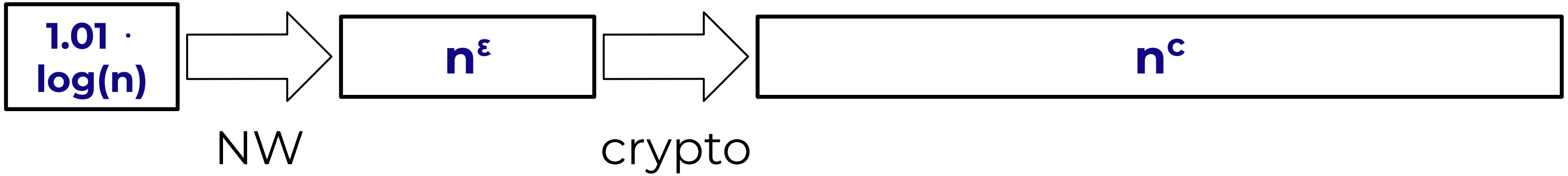
  › we'll use it as a non-crypto PRG, i.e. distinguisher is weaker



1 take PRG $n^\varepsilon \mapsto 2n^\varepsilon$ computable in time $n^{O(\varepsilon)}$, and "compose" it $\approx n^c$ times to extend output to n^c

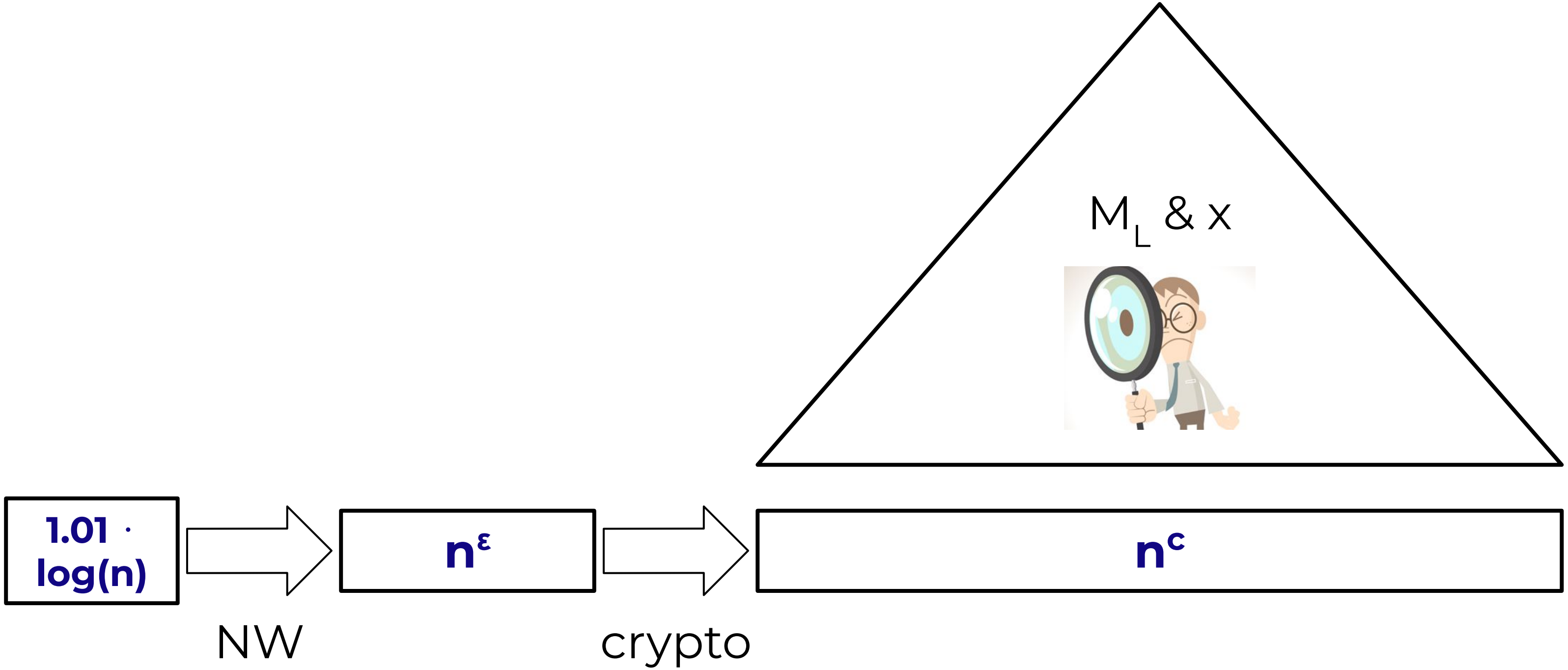
Tiny, superexp-hard truth-table

› a parametric overview



Tiny, superexp-hard truth-table

› a parametric overview

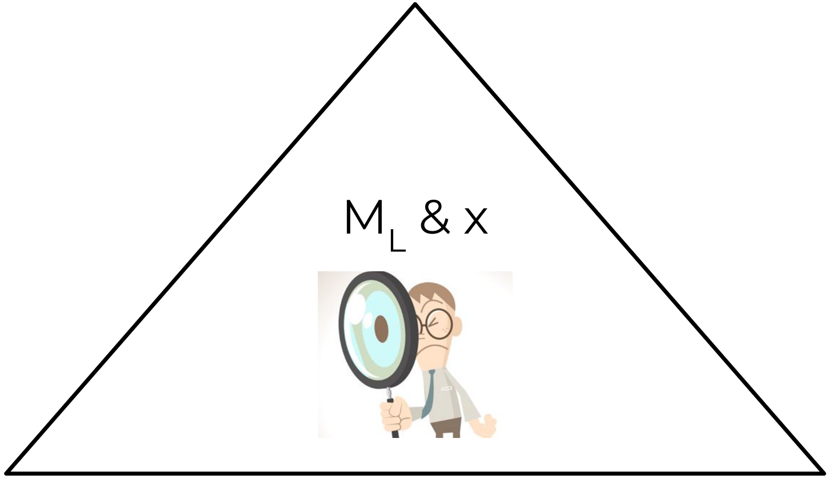


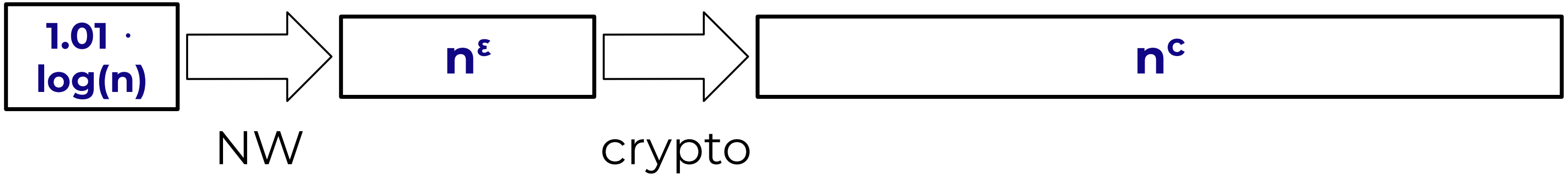
Tiny, superexp-hard truth-table

› a parametric overview

› Our distinguisher uses

1. time n^c
2. advice n



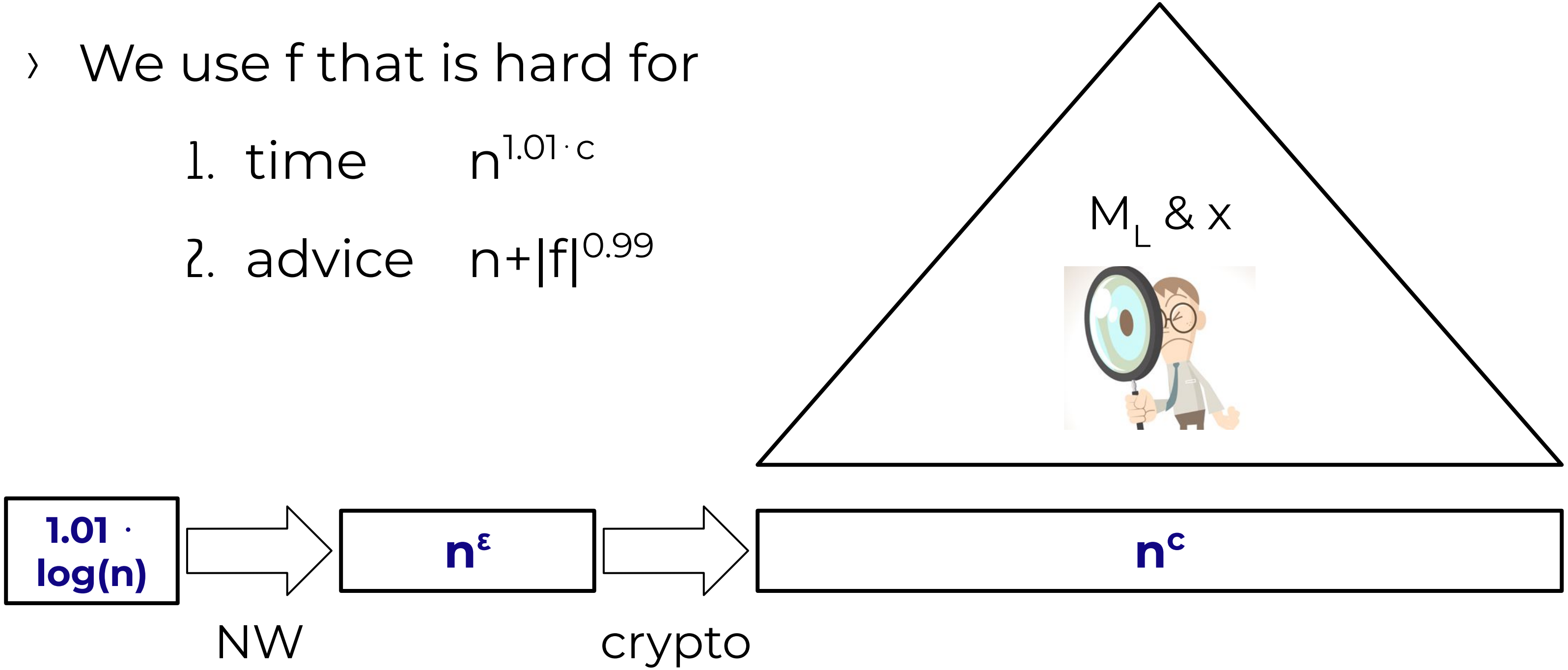


Tiny, superexp-hard truth-table

› a parametric overview

› We use f that is hard for

1. time $n^{1.01 \cdot c}$
2. advice $n+|f|^{0.99}$



Tiny, superexp-hard truth-table

› a parametric overview

› We use f that is hard for

1. time $n^{1.01 \cdot c}$
2. advice $n+|f|^{0.99}$

M_L & x

f $n^{1+O(\varepsilon)}$

1.01 · log(n) $\xrightarrow{\text{NW}}$ $\mathbf{n^{\varepsilon}}$ $\xrightarrow{\text{crypto}}$ $\mathbf{n^{c}}$

Tiny, superexp-hard truth-table

› a parametric overview

› We use f that is hard for

1. time $n^{1.01 \cdot c}$ $\approx \mathbf{2^{c \cdot \ell}}$
2. advice $n+|f|^{0.99} \approx \mathbf{2^{0.99 \cdot \ell}}$

M_L & x

f $n^{1+O(\varepsilon)} = \mathbf{2^{\ell}}$

1.01 · log(n) → NW → $\mathbf{n^{\varepsilon}}$ → crypto → $\mathbf{n^{c}}$

Hardness hypothesis

› generalizing classical hardness hypotheses

› Our derandomization uses a tiny truth-table with super-exponential time complexity

› Our hardness hypothesis (for $k \approx c$)

$$f \in \mathrm{TIME}[\, 2^{k \cdot \ell}\,] \text{ and hard for } \mathrm{TIME}[\, 2^{.99k \cdot \ell}\,]/2^{.99m}$$

A last small gap

- final running-time of derandomization?

- we'll have $n^{1.01}$ seeds (for the inner PRG NW)
- naive approach:
  - $\Rightarrow$ PRG computable on each seed in time $\approx T$
  - $\Rightarrow$ derandomization in time $O(n^{1.01} \cdot T)$
- unfortunately this doesn't work...

A last small gap

- we didn't *really* see that the PRG is linear-time computable yet

- our PRG is only computable per-seed in time $\approx n^{1.01} \cdot T$
  - need to compute the entire truth-table, even for one seed
- … but it's computable on all seeds in amortized time $\approx T$
  - suffices for derandomization
- … this allows relaxing the hypothesis, only requiring that f will be computable on all inputs in amortized time $\approx T$

A last small gap

- we didn’t *really* see that the PRG is linear-time computable yet

- Assuming OWFs, tight equivalence of
  1. hard functions with small amortized time-complexity
  2. batch-computable PRGs

- The “right” objects to study in hardness-to-randomness
  - the tightness is significant for superfast derandomization

Reminder of more results

› whose proof we won't see today

› Thm 2: Reduce overhead to $n^{1.01} \cdot T$ for $T(n) \leq 2^{o(n)}$

› Prop 3: Assuming #NSETH, overhead of $n^{.99} \cdot T$ is optimal

› Thm 4: Average-case derandomization with effectively no overhead at all (only n^{ε}, below lower bound)

Simplifying a well-known PRG paradigm

via quantified derandomization

Well-known PRG paradigm

- underlies [HILL’99, BSW’03, …, DMOZ’20]

- Well-studied paradigm for constructing PRGs
- Based on composition of two algorithms
  - pseudoentropy generator & extractor
- We will show: Any such composition can be viewed & analyzed in a very simple way

Well-known PRG paradigm

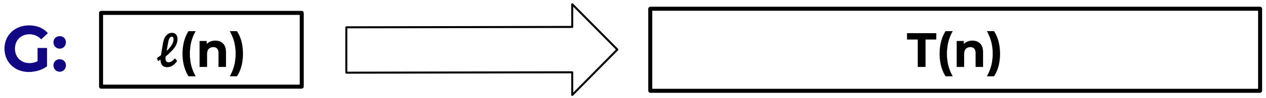
› underlies [HILL’99, BSW’03, …, DMOZ’20]

1. a pseudoentropy generator (PEG)

Well-known PRG paradigm

› underlies [HILL'99, BSW'03, ..., DMOZ'20]

1. a pseudoentropy generator (PEG)



Well-known PRG paradigm

› underlies [HILL'99, BSW'03, ..., DMOZ'20]

1. a pseudoentropy generator (PEG)

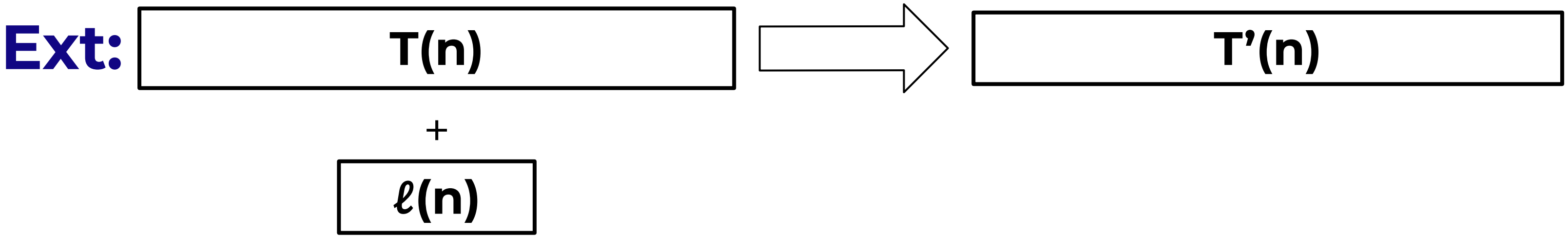
wow, this has
high entropy!

G: $\boxed{\ell(n)}$ ⇒ $\boxed{T(n)}$

Well-known PRG paradigm

› underlies [HILL’99, BSW’03, ..., DMOZ’20]

1. a pseudoentropy generator (PEG)
2. a randomness extractor

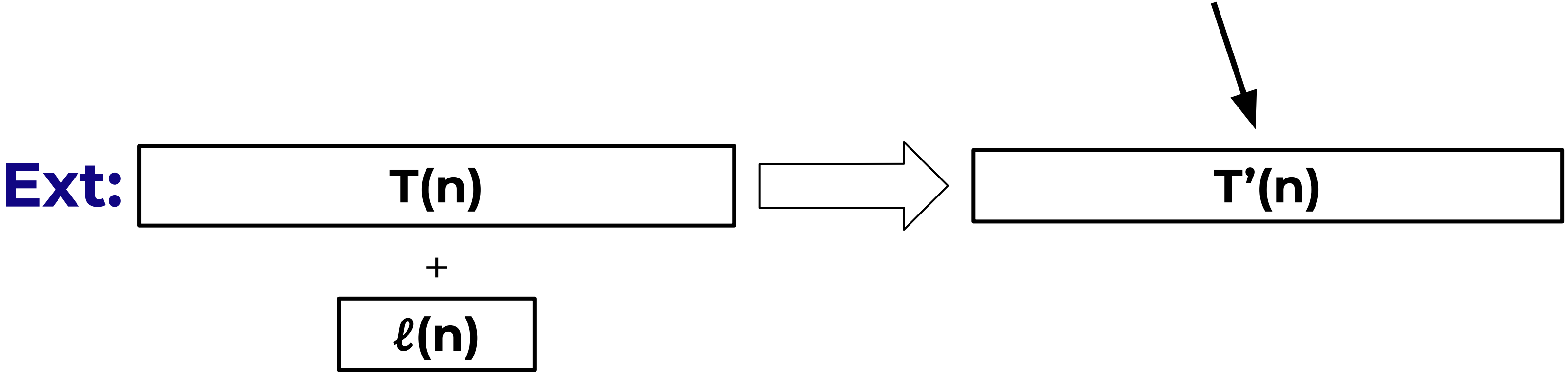


Well-known PRG paradigm

› underlies [HILL'99, BSW'03, …, DMOZ'20]

1. a pseudoentropy generator (PEG)
2. a randomness extractor

all the entropy "extracted"
to almost-uniform string



Well-known PRG paradigm

› underlies [HILL'99, BSW'03, …, DMOZ'20]

› PRG: **$G(s_1, s_2) = \mathrm{Ext}(\, \mathrm{PEG}(s_1), s_2 \,)$**

› Intuition: If $\mathrm{PEG}(s_1)$ looks entropic, then $\mathrm{Ext}(\, \mathrm{PEG}(s_1), s_2 \,)$ should look random

› Good extractors are known, so we "just" need a PEG, and to make the composition idea work

Well-known PRG paradigm

- underlies [HILL'99, BSW'03, …, DMOZ'20]

- Key problem: Idea hard to materialize
  - Extractors known, focus on PEG & composition
- Approach 1: Construct good PEGs
  (in which case composition works)
- Approach 2: Construct weak PEGs [DMOZ'20]
  and try to salvage composition

Well-known PRG paradigm

- underlies [HILL'99, BSW'03, …, DMOZ'20]

- Key problem: Idea hard to materialize
  - Extractors known, focus on PEG & composition
- Approach 1: Construct good PEGs (in which case composition works)
- Approach 2: Construct weak PEGs and try to salvage composition [DMOZ'20]

HARD TO DO

Easier & more general paradigm

- error-reduction then quantified derandomization

  - PRG: $\mathbf{G(s_1, s_2) = Ext(\ PEG(s_1), s_2\ )}$

Easier & more general paradigm

- error-reduction then quantified derandomization

- PRG: $\mathbf{G(s_1, s_2) = Ext(\,PEG(s_1), s_2\,)}$
- We show a simple general analysis such that
  - … composition is easy to prove
  - … generator can be weaker than in [DMOZ'20]

Easier & more general paradigm

- error-reduction then quantified derandomization

- PRG: $\mathbf{G(s_1, s_2) = Ext(\ PEG(s_1), s_2\ )}$
- We show a simple general analysis such that
  - … composition is easy to prove
  - … generator can be weaker than in [DMOZ'20]
- Meaning: New approach is easier & more general

Easier & more general paradigm

- error-reduction then quantified derandomization

- PRG: $\mathbf{G(s_1, s_2) = Ext(\ PEG(s_1), s_2\ )}$

- New analysis has two steps:

  1. (non-standard) error reduction, using Ext

  2. quantified derandomization, using the inner generator

Easier & more general paradigm

- error-reduction then quantified derandomization

- PRG: $\mathbf{G(s_1, s_2) = Ext(\ QD(s_1), s_2\ )}$

- New analysis has two steps:
  1. (non-standard) error reduction, using Ext
  2. quantified derandomization, using the inner generator

$\boxed{\textbf{QD}} \equiv \boxed{\textbf{metric (weak) PEG}}$

Easier & more general paradigm

› error-reduction then quantified derandomization

› PRG: $\mathbf{G(s_1, s_2) = Ext(\ QD(s_1), s_2\ )}$

› New analysis has two steps:

1. (non-standard) error reduction, using Ext
2. quantified derandomization, using the inner generator

$\boxed{\textbf{QD}}$ $\equiv$ $\boxed{\textbf{metric (weak) PEG}}$

IN [DMOZ] WE NEED A METRIC PEG FOR A NON-STANDARD CLASS OF DISTINGUISHERS

Easier & more general paradigm

- high-level recap

- Prop 5:

  Any construction that can be analyzed as

  “extract from a pseudoentropic string”

  can be analyzed (easily) as

  “non-standard error-reduction and QD”

- (converse not known)

Derandomization with overhead $c \in \{2,3,4\}$

- easy & versatile proof for superfast derandomization

- Cor 1: New simple proof for main result of [DMOZ'20]
  - use hypothesis to get a QD generator
  - combine QD & Ext in the simple way

- Cor 2: Proof extends to cubic/quartic derandomization from hardness only for NSIZE
  - (details in the paper)

4 Key takeaways

results to remember

Take-home message

1. Derandomization with overhead $\approx n \cdot T(n)$ possible under natural assumptions
2. Simple & intuitive proofs yield conditional derandomization with overhead $c \in \{1,2,3,4\} + \varepsilon$
3. Broadening the theoretical basis for superfast derandomization

Results from an upcoming work

- under preparation, again joint with Lijie Chen

- Superfast derandomization in time $n^{0.01} \cdot T$:

  $\Rightarrow$ from fully uniform assumptions

  $\Rightarrow$ wrt all polynomial-time-samplable distributions

- Under uniform assumptions, randomness is "indistinguishable from useless" for decision problems and natural search problems

A sample of open questions

› new area to explore

1. Is the overhead of $n \cdot T$ optimal?
   › evidence without #NSETH
2. Superfast derandomization from classical hypotheses?
   › no crypto, no hardness for MASIZE/NSIZE
   › boils down to the hybrid argument barrier
3. Search-to-decision with minimal overhead?
   › true given OWFs, show unconditional reduction

Thank you!

$\Rightarrow$ derandomization in near-linear time
$\Rightarrow$ simple & intuitive proofs, high-level insights
$\Rightarrow$ broadening theoretical basis for superfast derand