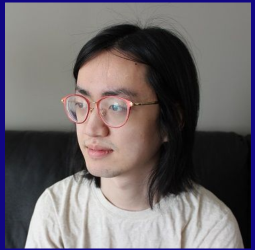
Unstructured hardness to average-case randomness

Lijie Chen and Ron Rothblum and Roei Tell

Simons, Feb 2023



Context

› what are we doing here?

- Recall that BPP = P if:
  - TIME[2^n] is hard for circuits of size $2^{\varepsilon \cdot n}$ [NW94, IW97]
  - TIMEDEPTH[n^{100}, n^2] is almost-all-inputs hard for probabilistic time n^{20} [CT21b, LP22a, LP22b, vMS23, …]

1 it's actually promiseBPP = promiseP, but I'll ignore it throughout the talk

Context

› what are we doing here?

› What if we only want BPP ⊆ P “on average”?

[IW98, GW02, CIS18]

1 “on average” = the derandomization succeeds on 1-1/n of inputs (over the uniform distribution)

Context

› what are we doing here?

› What if we only want BPP ⊆ P "on average"?

› standard hardness for algorithms?

[IW98, GW02, CIS18]

1 "on average" = the derandomization succeeds on 1-1/n of inputs (over the uniform distribution)

Context

› what are we doing here?

- What if we only want BPP $\subseteq$ P "on average"?
  - standard hardness for algorithms?
  - weak and intuitive assumptions?

[IW98, GW02, CIS18]

1 "on average" = the derandomization succeeds on 1-1/n of inputs (over the uniform distribution)

A sample theorem

- ... jumping way ahead

  - **Thm:** Assume that counting k-cliques requires probabilistic time $n^{c(k)}$, where c(k) grows with k.

    Then, RP = P on average.

1 "on average" = the derandomization succeeds on 1-1/n of inputs (over the uniform distribution)

Plan

- let's do it

1. A classical missing piece in hardness vs randomness
2. A natural and intuitive setting for hardness vs randomness
3. Some constructions & proof ideas
   - targeted PRGs, tolerant instance checkers, worst-case to avg-case

1 Classical missing piece

Hardness vs randomness

› historic recap

- Main focus is equivalence between explicit
  - Lower bounds for circuits
  - Pseudorandom generators for circuits

USED FOR
DERANDOMIZATION

Hardness vs randomness

› historic recap

- Main focus is equivalence between explicit
  - Lower bounds for circuits
  - Pseudorandom generators for circuits

$\boxed{\mathrm{TIME}[2^n] \not\subset \mathrm{ioSIZE}[2^{.01 \cdot n}]} \Longleftrightarrow \boxed{\text{log-seed PRG for ckts}}$

[NW94,IW97]

Hardness vs randomness

› historic recap

- Main focus is equivalence between explicit
  - Lower bounds for circuits
  - Pseudorandom generators for circuits

$\mathrm{TIME}[2^n] \not\subset \mathrm{ioSIZE}[2^{.01 \cdot n}]$ $\Leftrightarrow$ log-seed PRG for ckts

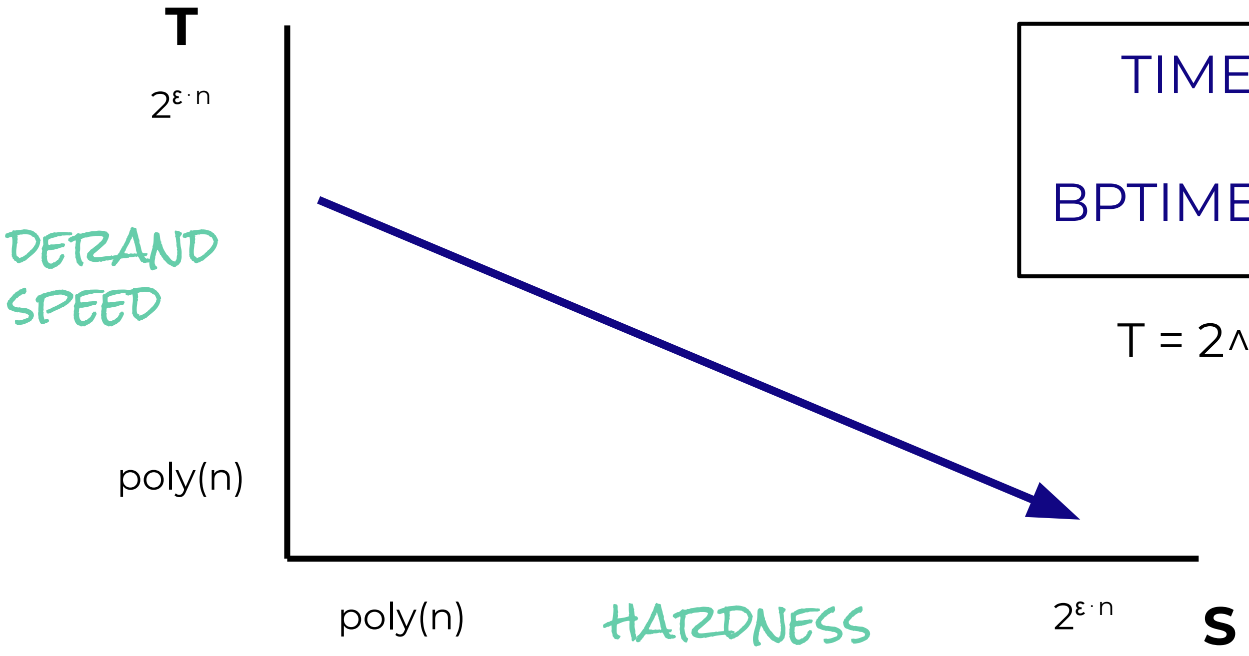
[NW94,IW97]

$\Downarrow$

BPP = P

General smooth tradeoff

- proved in [SU02, Uma03]



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

$T = 2 \wedge \{ O(S^{-1}(n^{O(1)})) \}$

Hardness vs randomness

› historic recap

- Second focus is equivalence between explicit
  - Lower bounds for uniform probabilistic algs
  - PRGs for uniform probabilistic distinguishers

EQUIVALENT TO
AVERAGE-CASE
DERANDOMIZATION

Hardness vs randomness

› historic recap

- Second focus is equivalence between explicit
  - Lower bounds for uniform probabilistic algs
  - PRGs for uniform probabilistic distinguishers

$\mathrm{TIME}[2^n] \not\subseteq \mathrm{BPP}$ $\Longleftrightarrow$ $\mathrm{BPP} \subseteq \mathrm{ioSUBEXP}$ on average

[IW98]

Hardness vs randomness

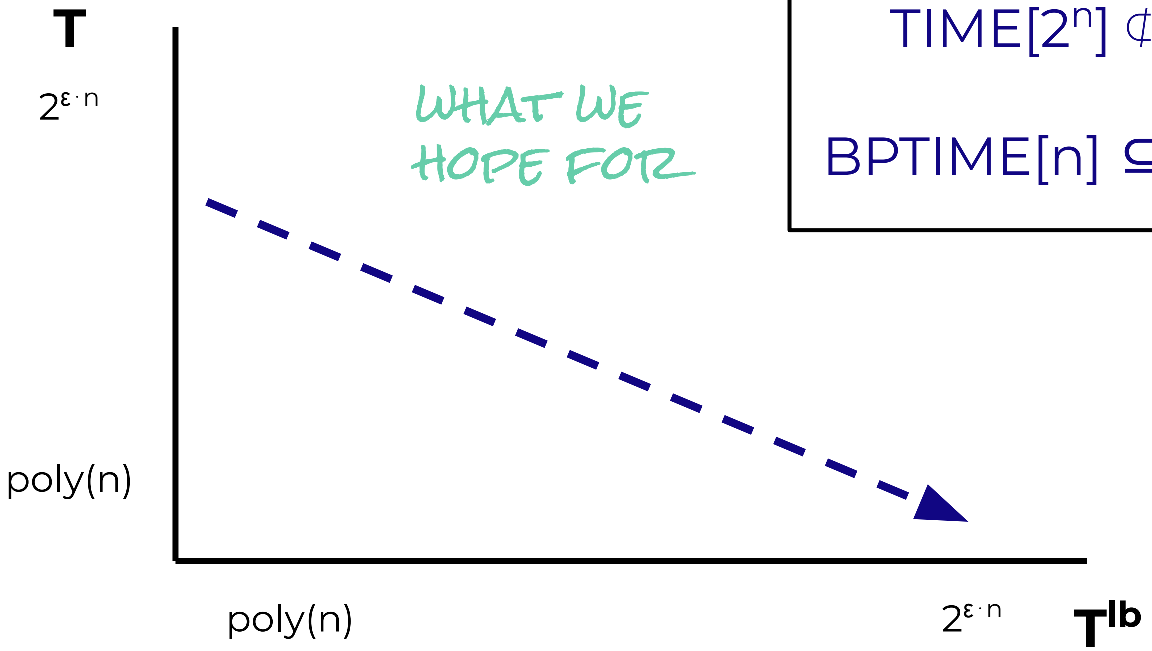
› historic recap

- infinitely-often
  vs
  almost-always
- average-case over
  which distribution

  (⇒ always uniform)

General smooth tradeoff

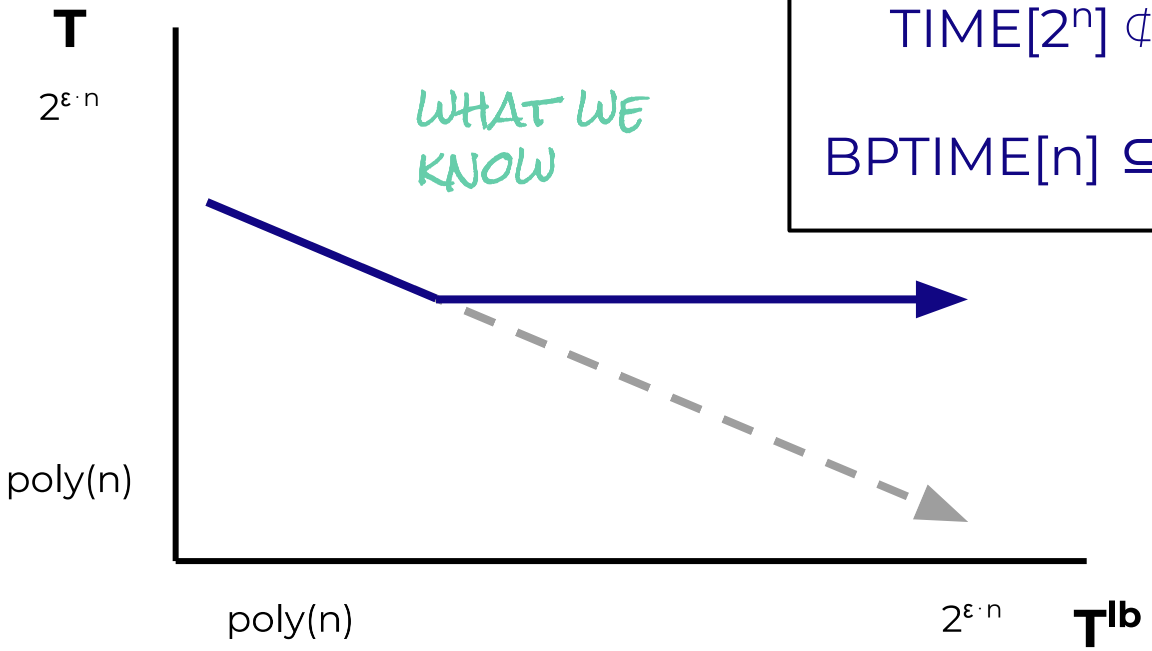
- an analogous "ideal" result



$$\mathrm{TIME}[2^n] \not\subset \mathrm{BPTIME}[T^{lb}]$$
$$\Rightarrow$$
$$\mathrm{BPTIME}[n] \subseteq \text{avg-}\mathrm{DTIME}[T]$$

A non-smooth tradeoff

- best known result [IW98]



$$\mathrm{TIME}[2^n] \not\subset \mathrm{BPTIME}[T^{lb}]$$
$$\Rightarrow$$
$$\mathrm{BPTIME}[n] \subseteq \text{avg-}\mathrm{DTIME}[T]$$

Proof approach

› reconstruction argument

› Proof by a reconstruction argument:

  › break the PRG ⇒ "efficiently" compute the hard func

› Reconstruction hard-wires non-uniform advice

  ⇒ yields a circuit computing the hard func

  ⇒ we assume hardness for circuits

Proof approach

› uniform reconstruction arguments

- Proof by a uniform reconstruction argument:
  - break the PRG ⇒ compute the hard func by a uniform alg
- Reconstruction has to be a uniform algorithm
- Problem: No efficient uniform reconstruction for arbitrary functions in time 2^n

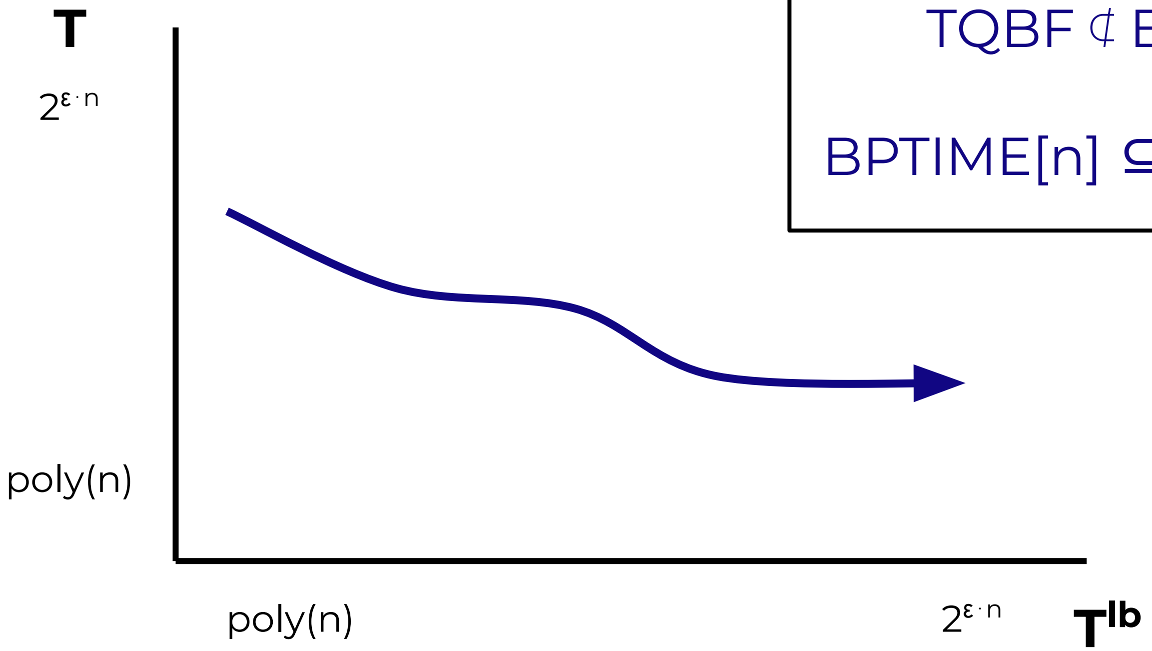
Classical barrier

› uniform reconstruction arguments

- Idea [IW98, CNS99, TV02]: Use specific hard functions that have "nice" structural properties
  - downward self-reducible + random self-reducible
- Uniform reconstruction, relying on nice properties

A tradeoff for specific functions

› best known result for PSPACE [TV02, CRTY20]



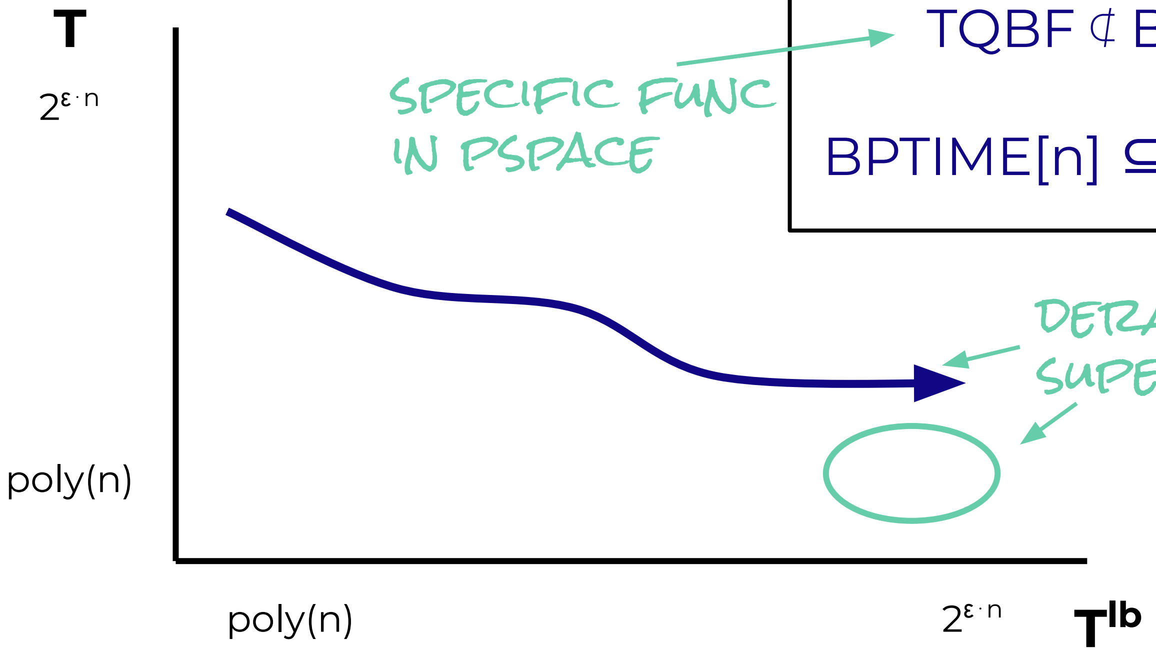
$$\mathrm{TQBF} \not\subseteq \mathrm{BPTIME}[T^{lb}]$$
$$\Rightarrow$$
$$\mathrm{BPTIME}[n] \subseteq \text{avg-}\mathrm{DTIME}[T]$$

A tradeoff for specific functions

› best known result for PSPACE [TV02, CRTY20]

SPECIFIC FUNC IN PSPACE

$$\mathrm{TQBF} \not\subset \mathrm{BPTIME}[T^{lb}]$$
$$\Rightarrow$$
$$\mathrm{BPTIME}[n] \subseteq \text{avg-}\mathrm{DTIME}[T]$$



DERAND STILL IN SUPER-POLY TIME

Classical barrier

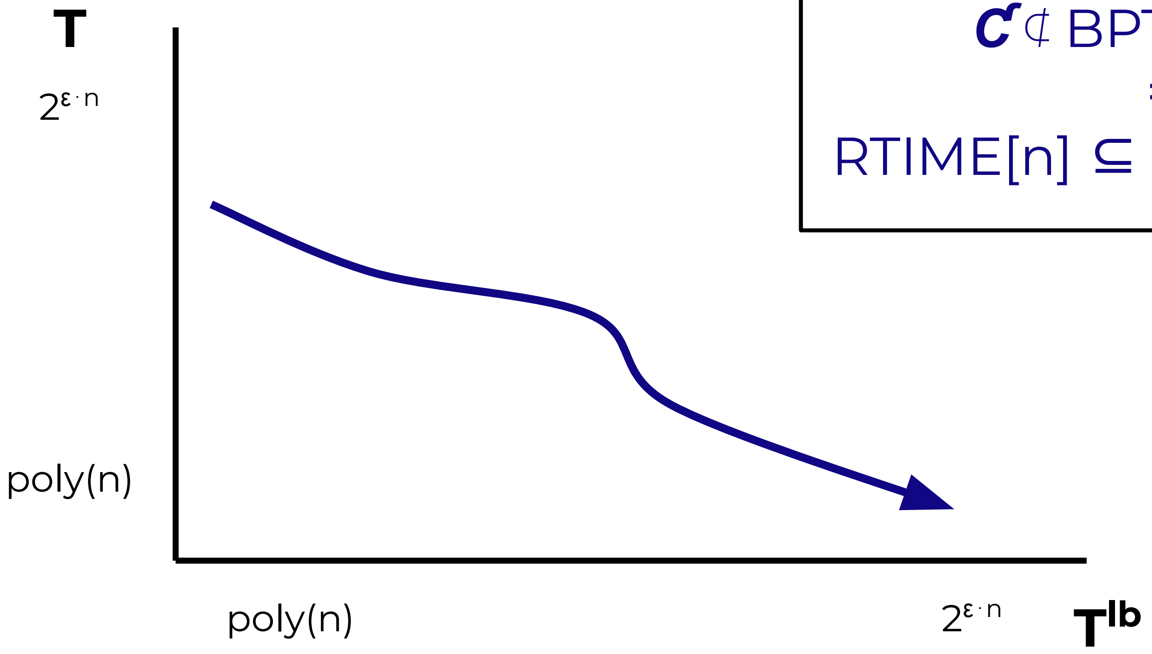
› uniform reconstruction arguments

› Limitations of the idea:

1. such funcs (provably) exist only in PSPACE
2. these are very specific funcs
3. known suitable funcs aren't hard enough
4. arguments incur runtime overheads

First main result

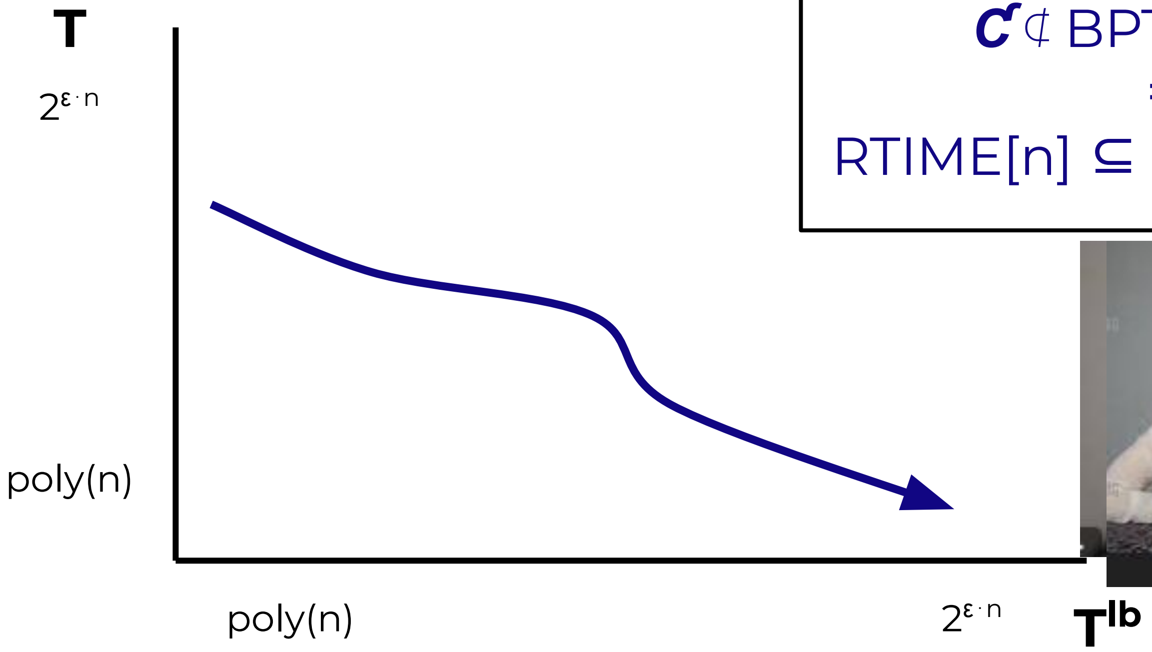
- informal diagram



$$\mathcal{C} \not\subset \mathrm{BPTIME}[T^{lb}]$$
$$\Rightarrow$$
$$\mathrm{RTIME}[n] \subseteq \text{avg-}\mathrm{DTIME}[T]$$

First main result

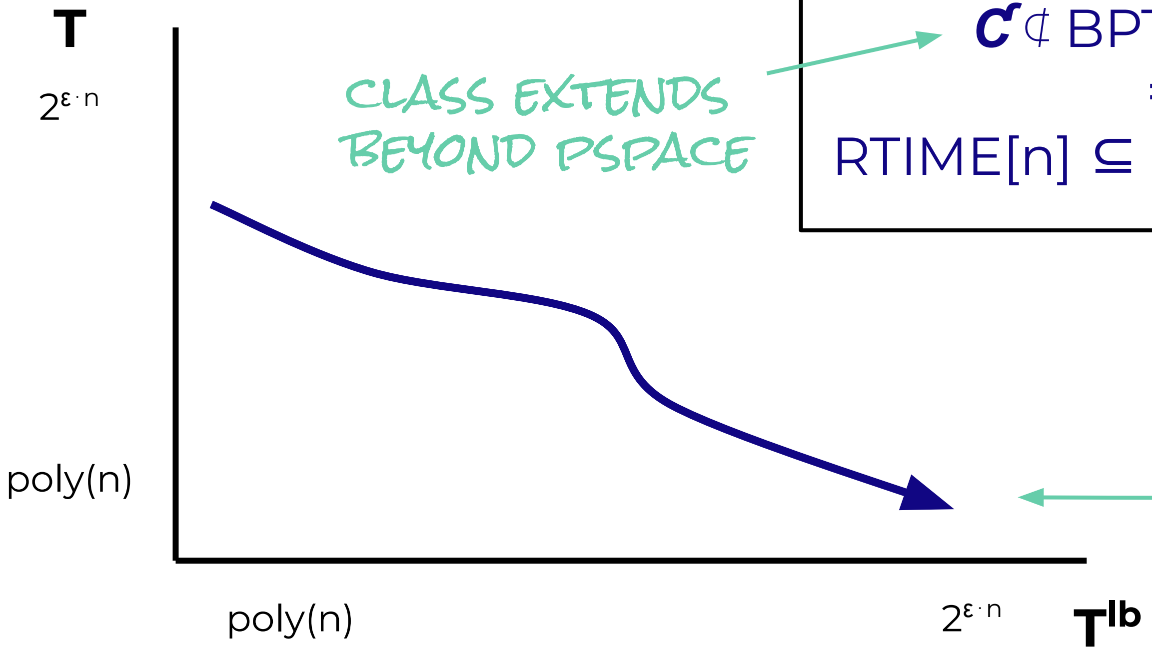
- informal diagram



$$\mathcal{C} \not\subset \mathrm{BPTIME}[T^{lb}]$$
$$\Rightarrow$$
$$\mathrm{RTIME}[n] \subseteq \text{avg-}\mathrm{DTIME}[T]$$

First main result

- informal diagram

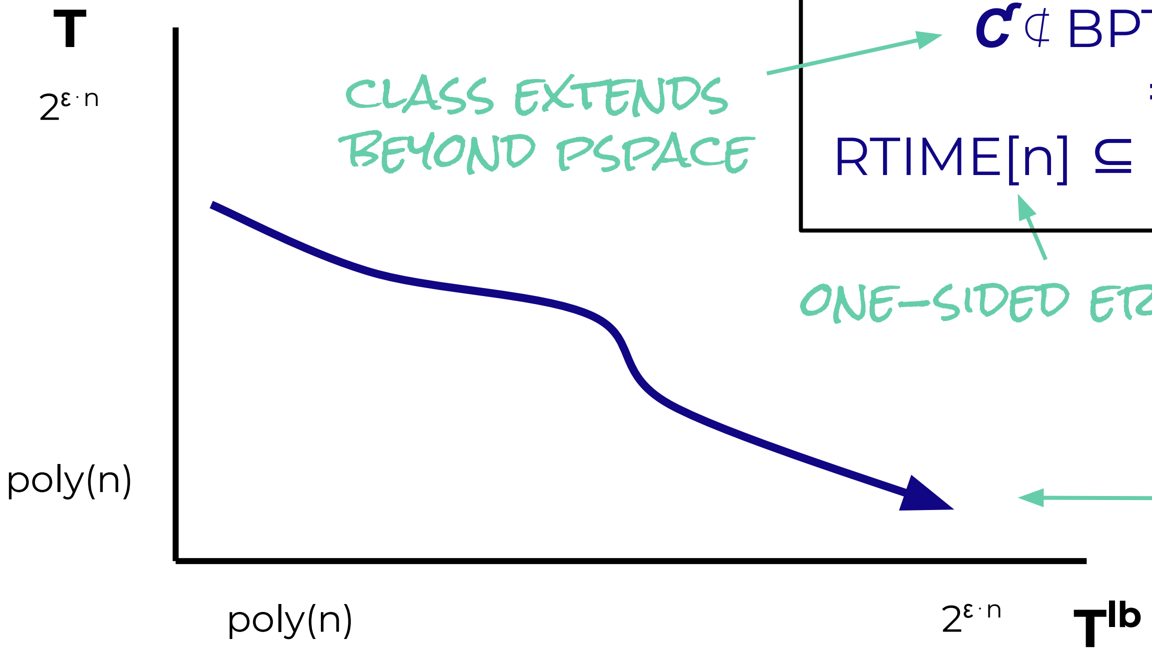


$$\boxed{\begin{gathered} \mathcal{C} \not\subset \mathrm{BPTIME}[T^{lb}] \\ \Rightarrow \\ \mathrm{RTIME}[n] \subseteq \text{avg-}\mathrm{DTIME}[T] \end{gathered}}$$

POLYTIME DERAND

First main result

- informal diagram



CLASS EXTENDS BEYOND PSPACE

$$\mathcal{C} \not\subset \mathrm{BPTIME}[T^{lb}]$$
$$\Rightarrow$$
$$\mathrm{RTIME}[n] \subseteq \text{avg-}\mathrm{DTIME}[T]$$

ONE-SIDED ERROR

POLYTIME DERAND

First main result

- breaking the PSPACE barrier, getting polytime derand

  - **Thm 1:** For $\mathcal{C}$ = lu-SIZEDEPTH[$2^{O(n)}$, $2^{o(n)}$]

    $\mathcal{C} \not\subseteq$ BPTIME[$2^{\varepsilon \cdot n}$] $\Rightarrow$ RP $\subseteq$ avg-P

1 lu = logspace-uniform = printable by a machine using space logarithmic in the circuit size

First main result

› breaking the PSPACE barrier, getting polytime derand

› **Thm 1:** For $\mathcal{C}$ = lu-SIZEDEPTH[$2^{O(n)}$, $2^{o(n)}$]

$$\mathcal{C} \not\subseteq \text{BPTIME}[\, 2^{\varepsilon \cdot n} \,] \;\Rightarrow\; \text{RP} \subseteq \text{avg-P}$$

$$\mathcal{C} \not\subseteq \text{BPTIME}[\, T \,] \;\Rightarrow\; \text{RP} \subseteq \text{avg-TIME}[\, 2^{t(n)} \,]$$

$$t(n) = T^{-1}(\text{poly}(n))^2 / O(\log n)$$

1 lu = logspace-uniform = printable by a machine using space logarithmic in the circuit size

First main result

› breaking the PSPACE barrier, getting polytime derand

› **Thm 1:** For $\mathcal{C}$ = lu-SIZEDEPTH[$2^{O(n)}$, $2^{o(n)}$]

$$\mathcal{C} \not\subseteq \text{BPTIME}[\, 2^{\varepsilon \cdot n} \,] \;\Rightarrow\; \text{BPP} \subseteq \text{avg-P} / O(\log n)$$

$$\mathcal{C} \not\subseteq \text{BPTIME}[\, T \,] \;\Rightarrow\; \text{BPP} \subseteq \text{avg-P} / a(n)$$

$$t(n) = T^{-1}(\text{poly}(n))^2 / O(\log n)$$

$$a(n) = o(T^{-1}(\text{poly}(n)) + O(\log T^{-1}(n))$$

First main result

- breaking the PSPACE barrier, getting polytime derand

- **Thm 1:** For $\mathcal{C}$ = lu-SIZEDEPTH[$2^{O(n)}$, $2^{o(n)}$]

  $\mathcal{C} \not\subset$ BPTIME[$2^{\varepsilon \cdot n}$] $\Rightarrow$ RP $\subseteq$ avg-P

- $\mathcal{C} \ni$ TQBF, likely contains funcs outside PSPACE
- no "special structure" needed, any func in $\mathcal{C}$ will do
- polytime derandomization

1 lu = logspace-uniform = printable by a machine using space logarithmic in the circuit size

First main result

› breaking the PSPACE barrier, getting polytime derand

› **Thm 1:** For $\mathcal{C}$ = lu-SIZEDEPTH[$2^{O(n)}$, $2^{o(n)}$]

$$\mathcal{C} \not\subset \text{BPTIME}[\, 2^{\varepsilon \cdot n} \,] \;\Rightarrow\; \text{RP} \quad \subseteq \text{avg-P}$$

› derand only of RP, or of BPP but with advice

› $\mathcal{C}$ seems like a proper subset of TIME[2^n]

› tradeoff isn't perfectly smooth

1 lu = logspace-uniform = printable by a machine using space logarithmic in the circuit size

2 Derand from fine-grained hardness assumptions

1. Classical missing piece
2. Fine-grained assumptions
3. Constructions and proof ideas

The general question

- natural hardness assumptions

- Motivating question: Can we deduce derandomization from hardness for natural problems?
- Nice setting: Fine-grained hardness for problems in P
  - rich study of k-clique, k-OV, k-SUM, …
- Key point: This is trying to do something harder
  - hardness in TIME[2^n] is stronger assumption

Typically correct derandomization

- non-uniform hardness [GW02, MS05, Sha10, Sha11, KvMS12]
  - **Thm [KvMS12]:** Assume that $\forall c$ there's $L_c \in P$ that's $(1/n)$-hard for circuits of size n^c. Then, $BPP \subseteq$ avg-P.

Typically correct derandomization

- non-uniform hardness [GW02, MS05, Sha10, Sha11, KvMS12]

- **Thm [KvMS12]:** Assume that $\forall c$ there's $L_c \in P$ that's (1/n)-hard for circuits of size n^c. Then, $BPP \subseteq$ avg-P.
  - clean and general result …
  - … but hardness is for non-uniform circuits
  - … and also requires mild average-case hardness

Typically correct derandomization

- non-uniform hardness [GW02, MS05, Sha10, Sha11, KvMS12]

- **Thm [KvMS12]:** Assume that $\forall c$ there's $L_c \in P$ that's (1/n)-hard for circuits of size n^c. Then, $BPP \subseteq$ avg-P.

- Goal: Get a uniform analogue
  - lower bounds for machines, no advice
  - uniform hypothesis is necessary

Problem-centric derandomization

› hardness for specific problems

› **Thm [CIS18]:** Assume that $\forall k$, counting k-cliques is hard for probabilistic algorithms that run in time $n^{(\frac{1}{2}+\varepsilon)\cdot k}$.

Then, BPP $\subseteq$ avg-P.

Problem-centric derandomization

› hardness for specific problems

› **Thm [CIS18]:** Assume that $\forall k$, counting k-cliques is hard for probabilistic algorithms that run in time $n^{(\frac{1}{2}+\varepsilon)\cdot k}$.

Then, BPP $\subseteq$ avg-P.

- uniform hardness assumption …
- … but for specific problems with "nice structure"
- … and requires specific hardness

Problem-centric derandomization

› hardness for specific problems

› **Thm [CIS18]:** Assume that $\forall k$, counting k-cliques is hard for probabilistic algorithms that run in time $n^{(1/2+\varepsilon)\cdot k}$.

Then, BPP $\subseteq$ avg-P.

› Goal: Be more general wrt function & time bound

Second main result

- derandomization from weak fine-grained hardness

- **Thm 2:** Assume that for every $c \in N$ there's $L_c \in$ lu-TIMEDEPTH$[\, n^{O(1)}, n^2 \,]$ that's (1/n)-hard for probabilistic time n^c.

  Then, RP = P on average.

Third main result

› derandomization from weak fine-grained hardness

› **Thm 3:** Assume that for every $c \in N$ there's L_c computable by lu arithmetic formulas of polysize and degree n^2 over $GF(n^{O(1)})$ that's hard for probabilistic time n^c.

Then, RP = P on average.

Third main result

- derandomization from weak fine-grained hardness

  - **Proof idea:**

    ⇒ efficiently balance low-degree formulas → low depth

    ⇒ amplify worst-case hard ⇒ mild avgcase hard

    (because these are still low-degree formulas)

    ⇒ appeal to Thm 2 as a black box

3 Constructions and proof ideas

Natural hardness assumptions

› unstructured hardness in P

› **Thm 2:** Assume that for every $c \in \mathbb{N}$ there's $L_c \in$ lu-TIMEDEPTH[$n^{O(1)}$, n^2] that's (1/n)-hard for probabilistic time n^c.

Then, RP = P on average.

Basic building block

- refinement of a construction from [CT21]

- **Prop:** For every $f \in$ lu-SIZEDEPTH[n^{100}, n^2] there is a targeted HSG H_f that gets input x, prints a list of n-bit strings, and: ∀ time-n machine M & ∀ fixed x,

  $H_f(x)$ isn't pseudorandom for $M(x,\cdot)$

  $\Rightarrow \Pr[\, F_M(x) = f(x) \,] \geq 2/3$, where F_M runs in time n^{10}

Basic building block

- refinement of a construction from [CT21]

- **Prop:** For every $f \in$ lu-SIZEDEPTH$[\,T, d\,]$ there is a targeted HSG H_f that gets input x, prints a list of $T^{.01}$-bit strings, and: $\forall$ time-$T^{.01}$ machine M & $\forall$ fixed x,

  $H_f(x)$ isn't pseudorandom for $M(x,\cdot)$

  $\Rightarrow \Pr[\,F_M(x) = f(x)\,] \geq 2/3$, F_M runs in time $\mathrm{poly}(d,n) \cdot T^{.02}$

Basic building block

- refinement of a construction from [CT21]

- Key points:
  - f is hard on x for time n^{10}

    $\Rightarrow H_f$ pseudorandom on x
  - any $f \in$ lu-SIZEDEPTH$[\, n^{100}, n^2 \,]$ will do

Proof idea

- use the tarHSG for derandomization

  - Idea: Use the targeted HSG, rely on instance-wise hardness vs randomness

f hard in the worst case $\Rightarrow$ f hard on 99% of inputs $\Rightarrow$ derand on 99% of inputs

Proof idea

› use the tarHSG for derandomization

› Idea: Use the targeted HSG, rely on instance-wise hardness vs randomness

f hard in the worst case $\Rightarrow$ f hard on 99% of inputs $\Rightarrow$ derand on 99% of inputs

NOT KNOWN FOR THE RELEVANT CLASSES!

Proof idea

› getting strong average-case hardness

› <u>High-level approach:</u>

1. L is hard, every n^{20}-time alg fails on 1/n of inputs
2. encode L to L' by an efficient code (approximately locally list-decodable), say direct product
3. L' is hard, every n^{10}-time alg fails on 1-1/n of inputs

… *right?*

Proof idea

› getting strong average-case hardness

- What goes wrong?

  ⇒ fix an algorithm A' computing L' correctly on more than 1/n of inputs

  ⇒ local approximate list-decoder $A = \mathrm{Dec}^{A'}$ should compute L correctly on more than 1 - 1/n of inputs

  ⇒ problem: Dec only succeeds with **low** probability

Worst-case to average-case reductions

- via new instance checkers

- **Def:**

M is an instance checker for f if

- $\Pr[\ M^f(x) = f(x)\ ] = 1$
- $\forall O,$ $\Pr[\ M^O(x) \notin \{ f(x), \perp \}\ ] \leq .01$

Worst-case to average-case reductions

› via new instance checkers

› <u>Revised approach:</u> (after building an instance checker)

⇒ L is hard, every n^{20}-time alg fails on 1/n of inputs

⇒ reduce L to L' that has a good instance checker, argue that hardness is preserved

⇒ encode L' to L'' with (say) direct product, argue that every n^{10}-time alg fails on 1-1/n of inputs … *right?*

Worst-case to average-case reductions

- via new instance checkers

- **Def:**

  M is an $(\varepsilon, \varepsilon')$-tolerant instance checker for f if

  - f' agrees with f on $1-\varepsilon$ of inputs

    $\Rightarrow$ for $1-\varepsilon'$ of inputs, $\Pr[\, M^{f'}(x) = f(x) \,] \geq 2/3$

  - $\forall O, \Pr[\, M^{O}(x) \notin \{ f(x), \perp \} \,] \leq .01$

Worst-case to average-case reductions

- via new instance checkers

- because of infinitely-often vs almost-always issues,

  which are under the rug,

  the instance checker will need to tolerate very high corruption, think $\varepsilon \approx 1/n$ instead of $1 - \varepsilon \approx 1-1/n$

Worst-case to average-case reductions

› via new instance checkers

› Working approach: (w/ tolerant instance checker)

⇒ L is hard, every n^{20}-time alg fails on 1/n of inputs

⇒ reduce L to L' that has a good tolerant instance checker, argue that hardness is preserved

⇒ encode L' to L'' with (say) direct product, argue that every n^{10}-time alg fails on 1-1/n of inputs

Worst-case to average-case reductions

- via new instance checkers

- **Prop:** Every $L \in$ lu-SIZEDEPTH[T, d] is reducible in linear time to $L' \in$ lu-SIZEDEPTH[$T^{O(1)}$, $d \cdot$ polylog(T)] that has a same-length tolerant instance checker running in time poly(d, log(T), n).

Worst-case to average-case reductions

- via new instance checkers

- **Prop:** There is L that's complete for SPACE[O(n)] under linear-time reductions and has a same-length tolerant instance checker running in time poly(n).

  ⇒ optimal wc2ac for computing SPACE[O(n)] by probabilistic algorithms

Technical contribution

- optimal worst-case to average-case results

- **Thm:** $\forall$ nice $\varepsilon > 0$

$$\mathrm{SPACE}[\,O(n)\,] \not\subset \text{io-}\mathrm{BPTIME}[\,T\,]$$

$$\Rightarrow \mathrm{SPACE}[\,O(n)\,] \not\subset \text{io-avg}_{\frac{1}{2}+\varepsilon}\mathrm{BPTIME}[\,T'\,],$$

where $T' = T(n/c) \cdot \mathrm{poly}(\varepsilon/n)$.

Instance checker construction

› ideas

› Instance checker based on [GKR'15] proof system

$$x \mapsto (\, x, i, j \,)$$

ROUND

VERIFIER MESSAGE

› pretend that the proof system is history-independent

(in actuality it depends only on last O(1) rounds)

Instance checker construction

› ideas

- Instance checker based on [GKR'15] proof system

$$x \mapsto (\, x, i, j \,)$$

ROUND

VERIFIER MESSAGE

- #rounds $\approx d < T$ $\Rightarrow$ $|i| \leq O(\log T)$
- #coins $\approx O(\log T)$ $\Rightarrow$ $|j| = O(\log T)$

Instance checker construction

› ideas

› Instance checker based on [GKR'15] proof system

$$x \mapsto (\, x, i, j \,)$$

ROUND

VERIFIER MESSAGE

› trivial linear-time reduction from original problem

› prover is efficient ⇒ complexity upper-bound is preserved

› verifier is super-efficient ⇒ fast same-length instance checker

Instance checker construction

› ideas

› Instance checker based on [GKR'15] proof system

$$x \mapsto (\, x, i, j \,)$$

ROUND

VERIFIER MESSAGE

› <u>problem:</u> it's **not tolerant!**

› adversary can corrupt (say) only the last round

Instance checker construction

› ideas

› Instance checker based on [GKR'15] proof system

$$x \mapsto (\, x, i', j \,)$$

$$p_x(i',j) = \text{interpolates the} \approx n^2 \text{ polynomials}$$

› p_x can self-correct from errors

⇒ doesn't matter if error concentrated on one i

4 Open problems

1. Classical missing piece
2. Fine-grained assumptions
3. Results, constructions, proof ideas

Open problems

- classical “hardness vs randomness” framework still isn’t complete!

1. Polytime derand from hardness in TIME[2^n]

   (and then!) from fine-grained hardness in P

2. Strengthen conclusion to BPP = P on avg
   - goal: construct a computational merger

3. Prove smooth tradeoffs
   - match the non-uniform setting

Thank you!

⇒ breaking through a classical barrier
⇒ derand from natural fine-grained hardness
⇒ lots of open questions in hardness vs randomness