Electronic Colloquium on Computational Complexity, Report No. 94 (2023)

New ways of studying the $\mathcal{BPP} = \mathcal{P}$ conjecture

Lijie Chen [*] Roei Tell [†]

June 29, 2023

Abstract

What's new in the world of derandomization? Questions about pseudorandomness and derandomization have been driving progress in complexity theory for many decades. In this survey we will describe new approaches to the $\mathcal{BPP} = \mathcal{P}$ conjecture from recent years, as well as new questions, algorithmic approaches, and ways of thinking. For example: Do we really need pseudorandom generators for derandomization, or can we get away with weaker objects? Can we prove free lunch theorems, eliminating randomness with zero computational overhead? What hardness assumptions are necessary and sufficient for derandomization? And how do new advances in this area interact with progress in cryptography and in interactive proof systems?

Note: A version of this text originally appeared as an ACM SIGACT News Complexity Theory Column [CT23a].

[*] Miller Institute for Basic Research in Science at University of California, Berkeley, CA. Email: `wjmzbmr@gmail.com`

[†] The Institute for Advanced Study at Princeton NJ and the DIMACS Center at Rutgers University, NJ. Email: `roeitell@gmail.com`

ISSN 1433-8092

Contents

1 The classical question of derandomization

> *"We use randomness, because we must".* (Shafi Goldwasser, Weizmann Complexity Seminar, 2017.)

The authors of this column know how to add two n-bit integers without flipping any random coins. In fact, we know how to do it *well*: By a simple and fast algorithm. There are many textbook problems like that, which can be solved by algorithms that are fast, simple to describe and to analyze, and that do not need to flip random coins during their execution.

Yet, as Goldwasser's quote above asserts, some problems force us to use randomness. If we want to solve a problem on big data – data that is so big that is infeasible to read all of it – we must use randomness; this is the case in learning theory, property testing, and (more generally) sublinear-time algorithms. A similar situation arises in cryptography: In standard cryptographic settings, we are facing an adversary who is computationally stronger than us, and who knows all of our secret plans (i.e., the algorithm that we'll use); we must act unpredictably to thwart it.

There are computational tasks for which the picture is less clear: They can be solved efficiently using randomness, and they *may or may not* be solvable by efficient deterministic algorithms (that aren't allowed to flip random coins); that is, a deterministic algorithm might exist, but we haven't found one yet. A natural example is the polynomial identity testing problem, and there are many others (see, e.g., [Wig19, Section 7.1]). Nevertheless, in many cases over the years, when researchers highlighted such a computational task, someone came along and designed a good deterministic algorithm for it, robbing us of the illustrative example. Two celebrated instances have been deciding primality [AKS04], and solving undirected connectivity in logspace [Rei08].

Can we derandomize general classes? Since we seem to be able to derandomize algorithms for specific computational tasks, maybe we can do so more generally, for *every* task in some large class? To the best of our knowledge, this question was first formally studied by Gill [Gil77] (see also [AM77; Adl78; AKL+79; BG81]), and already in his initial paper he conjectured that fast general derandomization is possible.[1]

The $\mathcal{BPP} = \mathcal{P}$ conjecture, which asserts that all languages decidable in probabilistic polynomial time are also decidable in deterministic polynomial time, is one of the main open problems in complexity theory. And, as one might expect, it is connected to basically every other major theoretical area: For example, to $\mathcal{P}$ vs $\mathcal{NP}$, lower bounds for Turing machines and for non-uniform circuits, cryptography, interactive proof systems, fine-grained complexity, learning theory, meta-complexity, and more.

[1] Our understanding of his conjecture is that every function (possibly with multiple output bits) that is computable in probabilistic time $T(x)$ on every input x is also computable in deterministic time $O(T(x))$, on infinitely many inputs x; see [Gil77, Section 3].

These days, a more general conjecture is studied, asserting that derandomization holds also for the corresponding classes of *promise problems*, i.e. $pr\mathcal{BPP} = pr\mathcal{P}$.[2] The reason that we study the more general conjecture is that we understand the structure of $pr\mathcal{BPP}$ better than that of $\mathcal{BPP}$,[3] and that all of the known derandomization techniques apply to promise problems.

2 A recent shift in approach

This column describes a new approach towards studying the $pr\mathcal{BPP} = pr\mathcal{P}$ conjecture that has been developed in recent years, and its initial fruit. In a gist, instead using classical pseudorandom generators, recent works focused on targeted pseudorandom generators, which are objects whose goals are more modest, but nevertheless still suffice for derandomization. (In fact, targeted PRGs are the *right* object to study: They are sufficient and necessary for derandomization.)

Well, why is it good to have more modest goals? As it turns out, these goals can be achieved from *weaker assumptions*, and using *less computational resources*, compared to classical PRGs. But this is all jumping ahead. We first briefly recall what are classical PRGs and what are their shortcomings (in Section 2.1), and introduce targeted PRGs (in Section 2.2).

In Section 3 we describe the recent results obtained by constructing targeted PRGs, and their advantages. In Section 4 we present some broader applications of these new developments, for example in the areas of explicit constructions, cryptography, interactive proof systems, and algorithm design. Also, throughout the text we suggest many interesting open problems to pursue, and we include a "roadmap" for studying these open problems in Section 5.

2.1 The textbook approach: PRGs and their discontents

To set the stage, let's very briefly describe the classical theoretical evidence for the $pr\mathcal{BPP} = pr\mathcal{P}$ conjecture, which can be found in any relevant textbook (see, e.g. [Gol08, Chapter 8], [AB09, Chapter 20], and [Wig19, Chapter 7]). This is the celebrated "hardness vs randomness" framework.

2.1.1 Derandomization from PRGs

The theoretical evidence for $pr\mathcal{BPP} = pr\mathcal{P}$ originated in *cryptography* in the 1980's, from the study of pseudorandom generators (PRGs) (the earliest pioneers have been [Sha81;

[2] Recall that a promise problem is a pair $\Pi = (\mathsf{Y}, \mathsf{N}) \subseteq \{0,1\}^* \times \{0,1\}^*$ such that $\mathsf{Y} \cap \mathsf{N} = \emptyset$, and an algorithm solves Π if it accepts every $x \in \mathsf{Y}$ and rejects every $x \in \mathsf{N}$ (we do not care how the algorithm behaves on $x \notin (\mathsf{Y} \cup \mathsf{N})$).

[3] Specifically, we know of a complete problem for $pr\mathcal{BPP}$ under $\mathcal{P}$-reductions, called CAPP (see, e.g., [Gol08, Exercise 6.14]); we have hierarchy theorems for $pr\mathcal{BPTIME}$ (see, e.g., [Bar02, Section 3.2]); we have a search-to-decision reduction for $pr\mathcal{BPP}$ (see [Gol11]); and we have a reduction of $pr\mathcal{BPP}$ to the "one-sided error" version, i.e. $pr\mathcal{BPP} = pr\mathcal{RP}^{pr\mathcal{RP}}$ (see [BF99]). None of these are known for $\mathcal{BPP}$.

BM84; Yao82]). These are objects that stretch a short random seed to a long pseudorandom output, where this output needs to look random to *every* procedure from some predetermined class $\mathcal{C}$:

Definition 2.1 (PRGs)**.** *A PRG with seed length $\ell(n)$ and error $\epsilon(n)$ for a class $\mathcal{C}$ of Boolean functions is an algorithm G satisfying, for every $C \in \mathcal{C}$ that takes n input bits,*

$$\Pr_{s \in \{0,1\}^{\ell(n)}}[C(G(1^n, s)) = 1] \in \Pr_{r \in \{0,1\}^n}[C(r) = 1] \pm \epsilon(n) .$$

The textbook approach uses PRGs for derandomization, by considering the class $\mathcal{C}$ of all circuits of linear size:[4] A circuit $C\colon \{0,1\}^T \to \{0,1\}$ of size $O(T)$ can represent the computation of a probabilistic T-time machine M on an input x, by letting $C(r) = M(x, r)$. Then, a PRG for such circuits allows to reduce the number of random coins used by M from T to $\ell(T)$. Finally, if $\ell(T)$ is small enough – say, $\ell(T) = O(\log(T))$ – then we can obtain a completely deterministic algorithm, by enumerating over all choices of $s \in \{0,1\}^{\ell(T)}$.

In contrast to cryptography, in derandomization we allow G to use more computational resources than the procedures in the adversarial class $\mathcal{C}$. For example, to derandomize linear-time machines in polynomial time (and deduce that $pr\mathcal{BPP} = pr\mathcal{P}$), it suffices to have a PRG for linear-sized circuits that has seed length $O(\log(n))$, error $< 1/6$, and running time $\text{poly}(n)$.

2.1.2 Hardness vs randomness: PRGs are equivalent to strong circuit lower bounds!

Here's one of the most celebrated theorems in complexity theory, which builds on earlier breakthroughs [Nis91; NW94]:

Theorem 2.2 (hardness vs randomness [IW97])**.** *There exists a PRG for linear sized circuits that has seed length $\ell(n) = O(\log(n))$ and error $1/\text{poly}(n)$ and running time $\text{poly}(n)$* if and only if *there is some constant $\epsilon > 0$ and $L \in \mathcal{DTIME}[2^n]$ such that no circuit of size $2^{\epsilon \cdot n}$ can compute L correctly on infinitely many input lengths.*

Thus, if you believe in certain strong circuit lower bounds, then you get the PRGs you need for $pr\mathcal{BPP} = pr\mathcal{P}$! And moreover, we're not asking you to believe in too much: This is *exactly* the right hypothesis for the existence of such PRGs (i.e., sufficient and necessary).

Theorem 2.2 is the poster-child of a rich line of works on hardness vs randomness. For example, with much effort, Theorem 2.2 is now known to scale well: Quantitatively weaker circuit lower bounds yield slower derandomization (i.e., hardness for smaller circuits yields slower PRGs with longer seed); see, e.g., [BFN+93; STV01; TSZS06; SU05; Uma03]. Also, Theorem 2.2 extends to hardness-vs-randomness tradeoffs for interactive proof systems: Lower bounds for circuits that use non-determinism yield generators for co-nondeterministic circuits, which suffice to prove that $pr\mathcal{AM} = pr\mathcal{NP}$,

[4] To avoid confusion, we note that some sources consider PRGs for circuits of polynomial size, but to deduce that $pr\mathcal{BPP} = pr\mathcal{P}$ it suffices to consider circuits of linear size.

and these results also scale quantitatively; see, e.g., [KM02; MV05; IKW02; SU05; GSTS03; SU07]. Lastly, we also know of *uniform* hardness vs randomness tradeoffs, in which lower bounds for probabilistic Turing machines (e.g., $\mathcal{EXP} \neq \mathcal{BPP}$) yield derandomization "on average" (e.g., for every $L \in \mathcal{BPP}$ there is $L' \in \mathcal{P}$ such that $\Pr_{x\in\{0,1\}^n}[L(x) = L'(x)] \geq 1 - o(1)$ for all $n \in \mathbb{N}$); this line of works on uniform tradeoffs is, in fact, not complete yet, and see, e.g. [IW98; CNS99; Kab01; TV07; GV08; CIS18; CRT+20; CRT22].

2.1.3 Why not stop here? Why can't complexity theorists ever just be happy?

Going back to the 1980's, why did we rush to use PRGs in the first place? If one looks at the problem of derandomization from first principles, then PRGs seem like an overkill: We are trying to design a *single algorithm* G that works against an *entire class* $\mathcal{C}$.

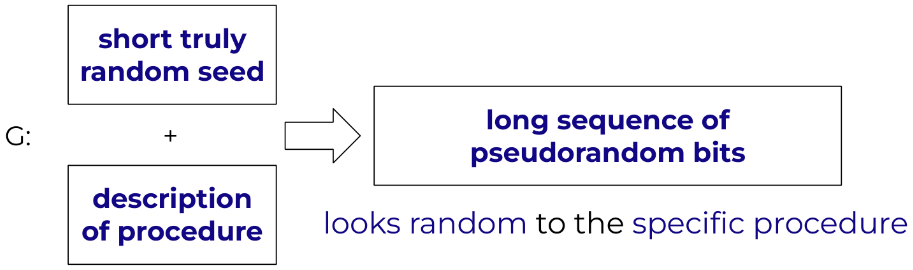
This is a non-obvious thing to do. Consider the textbook algorithm that uses PRGs to derandomize a probabilistic machine M. This algorithm gets an input x, computes a set S of pseudorandom strings while completely ignoring x (i.e., by running the PRG on all seeds), and only then remembers that it had x in the first place, and evaluates $M(x, s)$ for every $s \in S$.

This textbook PRG approach has two well-known shortcomings. First, to get a PRG against the class of linear-sized circuits, we *need strong circuit lower bounds*, provably (recall that Theorem 2.2 asserts an equivalence); this is a strong assumption, which we don't understand well. Secondly, the textbook PRG approach incurs *considerable computational overheads*. This is because: (1) The PRG needs a seed of length $\ell(T) \geq \log(T)$ to fool all circuits of size $O(T)$; and (2) The derandomization enumerates over the ℓ-bit seeds. When derandomizing T-time machines, this yields a multiplicative time overhead of T; when derandomizing T-time interactive protocols, this yields a multiplicative time overhead of T *per round* (so for 100 rounds, the overhead is T^{100}).

A daunting open problem. Maybe PRGs are, nevertheless, *necessary* for $pr\mathcal{BPP} = pr\mathcal{P}$? Equivalently, does $pr\mathcal{BPP} = pr\mathcal{P}$ imply strong circuit lower bounds as in Theorem 2.2?

This is essentially the case (with some caveats) for non-deterministic derandomization; that is, showing that $pr\mathcal{MA} = pr\mathcal{NP}$ requires PRGs that are computable in non-deterministic polynomial time (see, e.g., [Wil13; MW18; Che19; CR20; CLW20]). However, there has been virtually zero unconditional progress on this question for the $pr\mathcal{BPP} = pr\mathcal{P}$ setting, despite the importance of this problem and decades of interest. (See [CRT+20] for a *conditional* affirmative answer.)

Most frustratingly, until a few years ago, we also didn't know of any *barriers* to solving the open problem: If someone would've popped up and just *solved* the problem (as the authors have certainly tried), we didn't know of any other major consequences that would follow. These days we can say a bit more about surprising consequences (see Section 5).



G:
short truly random seed
+
description of procedure
long sequence of pseudorandom bits
looks random to the specific procedure

Figure 1: A visual depiction of a targeted PRG G, which gets as input a description of a procedure C, and maps a random seed of $\ell(n)$ bits to a long sequence of n bits that is pseudorandom for C. An alternative useful way to think of G is as mapping C to the pseudorandom set $\{G(\langle C\rangle, s)\}_{s\in\{0,1\}^\ell}$.

2.2 A non-black-box approach: Replacing PRGs with targeted PRGs

The cornerstone for recent progress is a pseudorandom object that is weaker than classical PRGs, and still suffices for derandomization. In fact, it is the *right* object, in the sense that it's both sufficient and *necessary* for derandomization of $pr\mathcal{BPP}$!

2.2.1 Defining targeted PRGs

In 2011, Goldreich [Gol11] introduced what he called targeted pseudorandom generators. Recall that a classical PRG stretches an $\ell(n)$-bit seed to an n-bit sequence that looks random to every procedure from a class $\mathcal{C}$ of computational procedures. A targeted PRG works just the same – it stretches $\ell(n)$ random bits to n pseudorandom bits – except that it also gets a description of a *specific procedure* $C \in \mathcal{C}$, and we're only asking that the n-bit output will look random to *this particular* C. For concreteness, let us define a targeted PRG for circuits:

Definition 2.3 (targeted PRG). *A* targeted pseudorandom generator (targeted PRG) *with seed length $\ell(n)$ and error $\epsilon(n)$ is an algorithm G that gets as input a description of a circuit $C\colon \{0,1\}^n \to \{0,1\}$ of size $O(n)$, and a random seed of length $\ell(n)$, and satisfies*

$$\Pr_{s\in\{0,1\}^{\ell(n)}}[C(G(\langle C\rangle, s)) = 1] \in \Pr_{r\in\{0,1\}^n}[C(r) = 1] \pm \epsilon(n) \, .$$

Intuitively, what we are asking the targeted PRG to do is read the *description* of the specific procedure C, understand something from the description, and use what it learned to produce pseudorandomness that is targeted specifically to C. This is why we think of targeted PRGs as *non-black-box algorithms*: They don't treat the procedure C as a black-box.

One can also define a targeted PRG for uniform classes. For example, fix a machine M (say, M solves the complete problem CAPP); then, the targeted PRG gets input

$x \in \{0,1\}^n$, and satisfies

$$\Pr_s[M(x, G(x,s)) = 1] \in \Pr_r[M(x,r) = 1] \pm \epsilon \ .$$

Note that we didn't bother to give a description of M to G as explicit input. This is because the description of M consists of constantly many bits of information – which can be hard-wired into G – whereas the interesting information is the input x, which can be arbitrarily long.

2.2.2 These are exactly the right object for $pr\mathcal{BPP} = pr\mathcal{P}$!

It is clear that targeted PRGs suffice for derandomization, because in derandomization we get an input x, and we only want to fool the specific procedure $M(x,\cdot)$; using the terminology of Definition 2.3, we have an explicit description of $C(r) = M(x,r)$. In his original work, Goldreich [Gol11] showed that targeted PRGs are also necessary for derandomization of $pr\mathcal{BPP}$:

Theorem 2.4. *A targeted PRG as in Definition 2.3 with seed length $O(\log(n))$, error $1/\mathrm{poly}(n)$, and running time $\mathrm{poly}(n)$ exists if and only if $pr\mathcal{BPP} = pr\mathcal{P}$.*

The second author of this column did not think highly of Theorem 2.4 during his PhD, because the proof is technically simple. Needless to say, the second author is eating his hat these days.

An indication for the non-triviality of Theorem 2.4 is the fact that we don't know whether or not an analogous result holds in the setting of $\mathcal{AM}$ vs $\mathcal{NP}$. Recall that a hitting-set generator (HSG) H is the one-sided error version of a PRG, where we require that if $\Pr_r[C(r) = 1] \geq 1/2$ then $\Pr_s[C(H(s)) = 1] > 0$. HSGs for co-nondeterministic circuits[5] suffice for derandomization of $pr\mathcal{AM}$ (see [FGM+89]), and in fact targeted HSGs, which are defined analogously to Definition 2.3, also suffice for this purpose. In his original paper, Goldreich asked:

Open Problem 2.5. *Is is true that $pr\mathcal{AM} = pr\mathcal{NP}$ if and only if there exists a targeted HSG for co-nondeterministic circuits of linear size that has seed length $O(\log(n))$ and runs in non-deterministic time $\mathrm{poly}(n)$?*[6]

The non-trivial direction is showing that derandomization of $pr\mathcal{AM}$ implies the existence of targeted HSGs for co-nondeterministic circuits. One step towards this was taken by Sdroievski and van Melkebeek [SM23], who showed that derandomization of $pr\mathcal{AMTIME}[2^{\mathrm{polylog}(n)}]$ yields a targeted HSG for co-nondeterministic circuits, albeit one that uses some non-uniform advice bits.

[5] A co-nondeterministic circuit C gets input r and a (non-deterministic) witness w. We say that C accepts r if for all w it holds that $C(r,w) = 1$, and that C rejects r if there exists w such that $C(r,w) = 0$.

[6] The meaning of the generator G being computable in *non-deterministic* time $\mathrm{poly}(n)$ is that there is a non-deterministic w such that $\Pr_s[C(G(\langle C \rangle, s, w)) = 1] \approx \Pr_r[C(r) = 1]$, in which case we call w good; and that for every w that is not good, we have $G(\langle C \rangle, s, w) = \perp$.

Theorem 2.6 (derandomization of $pr\mathcal{AM}$ requires targeted HSGs)**.** *If* $pr\mathcal{AMTIME}[n^{O(\log(n)^3)}] \subseteq$ $\texttt{i.o.}$-$pr\mathcal{NEXP}$*, then there exists a targeted HSG for co-nondeterministic circuits that yields the result* $pr\mathcal{AM} \subseteq \texttt{i.o.}$-$pr\mathcal{NTIME}[2^{n^c}]/n^\epsilon$ *for some* $c > 1$ *and all* $\epsilon > 0$.

3 Non-black-box hardness vs randomness

Having defined targeted PRGs, and explained why they seem to be the right object of study for derandomization, the natural question is: *Can we build one?*

3.1 Constructing targeted PRGs from hard functions

The pivotal development in derandomization in recent years has been

> **Constructions of targeted PRGs from functions that are hard for probabilistic (uniform) machines.**

The known constructions are sufficiently general, scalable, and robust that they can be viewed as a *non-black-box version of the classical hardness vs randomness framework*. The common thread in all of them is an instance-wise tradeoff of hardness vs randomness.

Instance-wise hardness vs randomness. Classical results assume the existence of a function f that is hard for small (non-uniform) circuits, and construct a PRG $G = G_f$ based on f. Correctness is established by proving that "if a small circuit can break the generator, then a small circuit can compute the hard function"; since f is hard for small circuits, the generator is secure.

Results from recent years assume the existence of a function f that is hard for *uniform probabilistic machines*, and construct a targeted PRG $G = G_f$. Correctness is established by proving a very similar statement, but *when all the players in this game are uniform, and are given access to a fixed auxiliary input x.* In more detail, fixing a function f and a machine M, we define a targeted PRG $G = G_f$ and a probabilistic uniform machine $F = F_{M,f}$, and prove that:

> For every $\underline{\text{fixed input } x}$, if $F(x)$ fails to compute $f(x)$, then $G(x, \cdot)$ fools $M(x, \cdot)$.

What does this give us? If we only assume that f is hard (for F) in the worst-case, i.e. hard on one input $x \in \{0,1\}^n$, then the derandomization succeeds only on x, which is obviously not useful at all. However, if f is hard on 99% of inputs, then the derandomization succeeds on 99% of inputs, which is definitely better. And if f is hard for F on 100% of inputs – indeed, such functions exist! see Section 3.3 – then the derandomization succeeds in the worst-case.

Current lines of work. There are, at the time of writing, two lines of constructions of targeted PRGs and of targeted HSGs from hard functions. Each line of works optimizes a different goal, addressing one of the two main shortcomings of classical PRGs (that were described in Section 2.1.3):

1. Constructions that **optimize the computational overhead**, and in particular avoid the overheads that are inherent to classical PRGs. Targeted PRGs with no computational overhead allow to obtain **free lunch theorems**, eliminating randomness at no (observable) cost. These constructions rely on strong (and quite non-standard) hardness assumptions.

2. Constructions that **optimize the hardness assumption**, and in particular allow obtaining **full equivalences** between $pr\mathcal{BPP} = pr\mathcal{P}$ and various assumptions from cryptography, complexity theory, learning theory, and more. Another construction optimizes the hardness assumption for derandomization of $\mathcal{AM}$, coming close to proving an equivalence.

The first line of works, and the resulting free lunch theorems, are presented in Section 3.2. The second line of works, with the resulting equivalence theorems, are presented in Section 3.3.

3.2 A lunch that looks free: Derandomizing without (noticeably) paying for it

In 2020, before the first construction of a targeted PRG was presented, Doron, Moshkovitz, Oh, and Zuckerman [DMO+20] asked the following question: Can we derandomize $pr\mathcal{BPP}$ faster than in classical results? Theorem 2.2 only guarantees that $pr\mathcal{BPP} = pr\mathcal{P}$, which implies (by a padding argument) that there exists $c \in \mathbb{N}$ such that $pr\mathcal{BPTIME}[T] \subseteq pr\mathcal{DTIME}[T^c]$ for every reasonable time bound $T(n)$. Certainly, this is unsatisfying when $c = 10^6$.

How small a constant c can we get? This question lies in the realm of *fine-grained complexity*. Doron *et al.* [DMO+20] showed that $c \approx 2$ might be possible, under strong hardness assumptions:

Theorem 3.1 (derandomization with quadratic overhead; see [DMO+20])**.** *For every $\epsilon > 0$ there is $\delta > 0$ such that the following holds. Assume that there is a problem in $\mathcal{DTIME}[2^n]$ that is hard on almost all input lengths for $\mathcal{MA}$ protocols that run in time $2^{(1-\delta)\cdot n}$ and receive $2^{(1-\delta)\cdot n}$ bits of non-uniform advice. Then, for every "nice" $T(n)$ it holds that $pr\mathcal{BPTIME}[T] \subseteq pr\mathcal{DTIME}[T^{2+\epsilon}]$.*

Their proof of Theorem 3.1 used classical PRGs. A follow-up work by the current authors showed that the result can be refined: Instead of paying an overhead of $T \mapsto T^2$, under strong assumptions it is possible to pay an overhead of $T \mapsto n \cdot T$, where n is the input length:

Theorem 3.2 (derandomization with overhead $T \mapsto n \cdot T$; see [CT21b]). *For every $\epsilon > 0$ there is $\delta > 0$ such that for every "nice" $T(n)$ there exists $k = k_{T,\epsilon} \in \mathbb{N}$ for which the following holds. Assume that there are one-way functions secure against polynomial sized circuits, and that there is a problem in $\mathcal{DTIME}[2^{k \cdot n}]$ that is hard on almost all input lengths for algorithms running in time $2^{(k-\delta)\cdot n}$ and receiving $2^{(1-\delta)\cdot n}$ bits of non-uniform advice. Then, $pr\mathcal{BPTIME}[T] \subseteq pr\mathcal{DTIME}[n \cdot T(n)^{1+\epsilon}]$.*

The proof of Theorem 3.2 also used classical PRGs. As explained in Section 2.1.3, the textbook PRG approach cannot do better than an overhead of $T \mapsto n \cdot T$. This is because the approach relies on a PRG that fools the class "T-time machines that have access to an arbitrary n-bit input", and fooling this class requires a seed of length $\ell(n) \geq \log(n)$.

Trying to get a free lunch with targeted PRGs. The overhead $T \mapsto n \cdot T^{1+\epsilon}$ in Theorem 3.2 is significant, for example when $T(n) = O(n)$ or for algorithms that are only slightly faster than brute-force (where the "$1 + \epsilon$" matters). The obvious question motivating further progress is:

> **Can we get derandomization "for free", eliminating randomness without paying in computational resources?**

Of course, we study this question using targeted PRGs. But there is still an obstacle along the way: Under natural assumptions, there is *no derandomization* that succeeds in the worst-case and avoids the computational overheads of PRGs, regardless of the algorithm used to obtain it.

Proposition 3.3 (a conditional lower bound; see [CT21b], following [Wil16]). *Suppose that* #NSETH *is true.*[7] *Then, for every $T(n) = \text{poly}(n)$ and $\epsilon > 0$ it holds that $\mathcal{BPTIME}[T] \not\subseteq \mathcal{DTIME}[n^{1-\epsilon} \cdot T]$.*

The overhead of $T \mapsto n \cdot T$ is most significant when T is small (e.g., $T(n) = O(n)$). Moreover, when derandomizing *interactive proof systems*, under the same assumption #NSETH, the necessary overhead is a large polynomial, specifically $T \mapsto n \cdot T^{\lceil c/2 \rceil}$ where c is the number of messages exchanged in the interaction (see Section 3.2.2 for details).

The specific problem that is hard to derandomize in the proof of Proposition 3.3 is essentially a polynomial identity testing problem (see [CT21b, Section 6]). There is another natural candidate hard problem, which we call the Time-Bounded Approximate Majority Problem (TAMP):

> For a time-computable $T = T(n) \geq n$, we are given as input n descriptions of Turing machines $M_1, ..., M_n$, where each description is of size $O(\log(n, T(n)))$.

[7] The assumption #NSETH asserts that for every constant $\epsilon > 0$, and for a sufficiently large $k = k_\epsilon > 0$, there does *not* exists a non-deterministic machine that gets an n-bit k-SAT formula Φ, runs in time $2^{(1-\epsilon)\cdot n}$, and outputs the number of assignments that satisfy Φ (for a precise formulation see, e.g., [CT21b, Section 6]).

> For each $i \in [n]$, define $\mu_i \in \{0,1\}$ as follows: If, when M_i is executed on an empty input, it halts after at most $T(n)$ steps and outputs 1, then $\mu_i = 1$; otherwise, $\mu_i = 0$. Our goal is to output "yes" if $\sum_{i\in[n]} \mu_i \geq 2n/3$, and "no" if $\sum_{i\in[n]} \mu_i \leq n/3$.

Note that TAMP with time bound T can be solved probabilistically in time $O(T)$ (by randomly choosing $i \in [n]$ and executing M_i for T steps). On the other hand, it seems plausible that TAMP cannot be solved deterministically in time $o(n) \cdot T$. We do not have any concrete evidence for this possibility, though, and we pose obtaining such evidence (e.g., deducing this lower bound from well-studied assumptions) as an open problem.

Derandomization that costs nothing and looks correct. Even if #NSETH is true, or if TAMP is hard for deterministic algorithms, there is still a way to bypass this obstacle: Instead of insisting on derandomization that is correct on *all inputs*, we relax the requirement and try to get derandomization that succeeds on *"most" inputs*. This type of "average-case derandomization" was considered as far back as [IW98], and can be viewed as part of the study of average-case complexity (see, e.g., [Gol08, Chapter 10.2], [AB09, Chapter 18]).

Recent works obtained a strong form of average-case derandomization: Under strong assumptions, they deduced **overhead-free derandomization such that no efficient algorithm can find inputs on which the derandomization errs**, except with negligible probability. In other words, these results do not necessarily yield a truly free lunch, but they yield *a lunch that looks free* to all efficient observers. Yet another way of putting it is that, under strong assumptions, randomness is *indistinguishable from useless* in the broad settings to which the results apply.

We will mention three such results: A free lunch theorem for derandomization of $\mathcal{BPP}$, in Section 3.2.1; a free lunch theorem for derandomization of interactive proof systems, in Section 3.2.2; and derandomization of space-bounded machines that induces minimal memory overhead, in Section 3.2.3. We discuss the hardness assumptions needed for free lunch theorems in Section 3.2.4.

3.2.1 A free lunch theorem for derandomization of $\mathcal{BPP}$

Our goal is to show that for every $L \in \mathcal{BPTIME}[T]$ there is $L' \in \mathcal{DTIME}[\approx T]$ such that no efficient algorithm can find an input on which L and L' disagree. Let's define this notion:

Definition 3.4 (indistinguishable languages). *We say that $L \subseteq \{0,1\}^*$ and $L' \subseteq \{0,1\}^*$ are* indistinguishable *if for every probabilistic algorithm A that gets input 1^n and runs in time* $\mathrm{poly}(n)$*, the probability that $A(1^n)$ prints $x \in \{0,1\}^n$ such that $L(x) \neq L'(x)$ is at most $n^{-\omega(1)}$.*

It turns out that this is possible under a strong non-batch-computability assumption for probabilistic algorithms. Specifically, consider the assumption that there is a function $f\colon \{0,1\}^n \to \{0,1\}^k$ such that each individual output bit of $f(x)$ is computable in time T', but printing the entire k-bit string $f(x)$ requires more time than T'; specifically, assume that printing $f(x)$ cannot be done in probabilistic time $k^{.001} \cdot T$. (A trivial algorithm can print $f(x)$ in time $k \cdot T$.)

So far, this is quite a standard assumption. For example, if we start from $g\colon \{0,1\}^{n/k} \to \{0,1\}$ and let $f = g^{\times k}$ be the k-wise direct product of g, then this assumption seems very appealing. The main issue is that we do not only want worst-case hardness, but rather hardness over a class of distributions over the inputs; that is, we will require that for every distribution $\mathbf{x}_n$ over the inputs (from a certain class of distributions), and every probabilistic algorithm A running in time $k^{.001} \cdot T$, with high probability over choice of $x \sim \mathbf{x}_n$, the algorithm A fails to print $f(x)$.[8]

A direct-product assumption where $f = g^{\times k}$ yields average-case hardness over the uniform distribution $\mathbf{x}_n = \mathbf{u}_n$; this implies derandomization that succeeds on .99 of the inputs over the uniform distribution (see [CT21a, Theorem 1.6]). We want derandomization that succeeds, with high probability, over all polynomial-time samplable distributions. Therefore, we will require that the average-case hardness holds over all polynomial-time samplable distributions, as follows.

Theorem 3.5 (free lunch derandomization of $\mathcal{BPP}$; see [CT21a])**.** *Let $T(n)$ be a polynomial. Assume that one-way functions exist, and that for every $\epsilon > 0$ there is $\delta > 0$ and a function $f\colon \{0,1\}^n \to \{0,1\}^{n^\epsilon}$ such that:*

1. *There is an algorithm that gets input $(x,i) \in \{0,1\}^n \times [n^\epsilon]$ and outputs the i^{th} bit of $f(x)$ in time $T'(n)$, where $T'(n) = T(n) \cdot n^\epsilon$.*
2. *For every probabilistic algorithm A that runs in time $T'(n) \cdot n^\delta$ and every polynomial-time samplable distribution $\mathbf{x} = \{\mathbf{x}_n\}_{n \in \mathbb{N}}$ and every large enough $n \in \mathbb{N}$, with probability $1 - n^{-\omega(1)}$ over $x \sim \mathbf{x}_n$ it holds that that $\Pr_{i,A}\left[A(x)_i = f(x)_i\right] < .99$.*

Then, for every $L \in \mathcal{BPTIME}[T]$ there is $L' \in \mathcal{DTIME}[T \cdot n^{O(\epsilon)}]$ such that L and L' are indistinguishable.

We do not know if the assumption about f in Theorem 3.5 is necessary for the conclusion. However, we do know that a certain non-batch-computability assumption is necessary for derandomization in one special case: An overhead-free derandomization of probabilistic balanced DLOGTIME-uniform formulas. See [CT21a, Section 6.3] for details.

3.2.2 A free lunch theorem for derandomization of interactive proof systems

Can we derandomize *interactive proof systems* without overhead? The story here is very similar to the one for derandomization of $pr\mathcal{BPP}$. Assuming strong lower bounds

[8] We also require that A fails to print even an *approximate* version of $f(x)$. This is a stronger requirement, but it is obtained in the direct-product setting; see [CT21b, Section 1.2.1] for precise details.

for non-uniform models, and using classical PRGs, we can derandomize $pr\mathcal{MA}$ with a *quadratic overhead*, and derandomize $pr\mathcal{AM}$ protocols in which $c = O(1)$ messages are exchanged with time overhead $T \mapsto n \cdot T^{\lceil c/2 \rceil + \epsilon}$ (see [CT23b, Theorems 1.1 and 1.2]). These two results are optimal when using the PRG approach; and assuming #NSETH, they are optimal regardless of the algorithm involved, when insisting on derandomization that succeeds in the worst-case (see [CT23b, Theorems 6.1 and 6.2]).

We will take an approach similar to that in Section 3.2.2, of relaxing the worst-case requirement. In the current setting, we are trying to derandomize an interactive proof system into an $\mathcal{NP}$-type verifier V. Instead of requiring that no efficient adversary can find an input on which V *might* err with some proof, we require that **no efficient adversary can find an input x and a proof π that mislead the derandomized verifier** V. Let us define the appropriate notion:

Definition 3.6 (computationally sound $\mathcal{NP}$)**.** *We say that a promise problem $\Pi = (\mathsf{Y}, \mathsf{N})$ is in* computationally sound $\mathcal{NTIME}[T]$ *(or* cs-$\mathcal{NTIME}[T]$*, in short) if there exists a deterministic verifier V running in time T and an honest prover P running in time T^c (for some constant $c \geq 1$) such that the following holds:*

1. **(Completeness.)** *For every $x \in \mathsf{Y}$ it holds that $V(x, P(x)) = 1$.*

2. **(Computational soundness.)** *For every adversary $\hat{P}$ running in probabilistic time* $\mathrm{poly}(T)$ *(this polynomial may be arbitrarily large), the probability that $\hat{P}(1^n)$ finds $x \in \{0,1\}^n$ and $\pi \in \{0,1\}^{T(n)}$ such that $x \in \mathsf{N}$ and $V(x, \pi) = 1$ is $T(n)^{-\omega(1)}$.*

Definition 3.6 puts forward a version of $\mathcal{NP}$ that has computational soundness, rather than information-theoretic soundness.[9] Indeed, Definition 3.6 can be viewed as an *argument system*, in the cryptographic sense, in which the verifier is deterministic.[10] The notion should not be confused with Micali's [Mic00] notion of "computationally sound proofs".

Now, recall that a doubly efficient proof system (as introduced in [GKR15]) is an interactive proof system in which the honest prover is efficient, and the verifier is extremely efficient. We denote by $\mathsf{de}\mathcal{IP}^{\leftrightharpoons c}[T]$ the set of languages decidable by doubly efficient interactive proof systems with a verifier running in time T, an honest prover running in time $\mathrm{poly}(T)$, and $c = O(1)$ turns of interaction. The following free lunch theorem asserts that under strong assumptions, such systems can be derandomized with essentially no overhead into computationally sound $\mathcal{NP}$:

Theorem 3.7 (free lunch derandomization of $\mathsf{de}\mathcal{IP}$; informal, see [CT23b])**.** *Under strong hardness assumptions, for every $\epsilon > 0$ and $c \in \mathbb{N}$ and "nice" time bound $T(n)$, it holds that*

$$\mathsf{de}\mathcal{IP}^{\leftrightharpoons c}[T] \subseteq \text{cs-}\mathcal{NTIME}[T^{1+\epsilon}] \, .$$

[9] The original definition in [CT23b] used a different name for the same notion, but we believe that the name "computationally sound $\mathcal{NP}$", which was suggested to us by Oded Goldreich, is better.

[10] Moreover, in comparison to standard cryptographic proof systems, the adversary is *uniform*, and the definition allows errors on *some* inputs, as long as such inputs are infeasible to find; see more details in Section 4.

The hardness assumptions in Theorem 3.7 are cumbersome to state, and can be found in [CT23b, Theorem 1.8]. At a high-level, the theorem relies on two assumptions. The first is a strengthening of the assumption in Theorem 3.5, now asserting that the probabilistic algorithm A fails to (approximately) print $f(x)$ even when it is given oracle access to an Arthur-Merlin protocol that takes T-bit inputs, runs in linear time $O(T)$, and uses c turns of interaction. This is a plausible strengthening, since computing even a single output bit of $f(x)$ takes time $T' = T \cdot n^{\epsilon} > T$, and thus it is not clear why a proof system running in time T should be helpful.

The first hardness assumption suffices to derandomize $\mathrm{de}\mathcal{IP}^{\leftrightharpoons c}[T]$ verifiers that use $T^{o(1)}$ random coins (see [CT23b, Theorem 1.5]). To handle the general case, the theorem relies on an additional assumption: There is a problem whose truth-tables on n-bit inputs can be printed in time $2^{(1+\epsilon/3)\cdot n}$, but that is hard for constant-round interactive protocols with a verifier that runs in time $2^{(1-\delta)\cdot n}$ and uses $2^{(1-\delta)\cdot n}$ bits of non-uniform advice (where $\delta = \delta_{\epsilon} > 0$ is sufficiently small).

3.2.3 Forgetting to pay: Derandomizing $\mathcal{BPL}$ with minimal memory footprint

In Sections 3.2.1 and 3.2.2 we tried to minimize the *time* overhead for derandomization. How about minimizing the *space* overhead? It turns out that this, too, can be done, under assumptions.

Doron and the second author [DT23] showed two algorithms yielding such derandomization, one that uses classical PRGs and one that uses targeted PRGs. For simplicity, let us only present the former. Recall that a function f is computable in *catalytic space* S if there is a machine that gets input x and access to a worktape of length S that is filled with arbitrary information, computes $f(x)$ (while possibly manipulating the contents of the worktape), and is required to return the contents of the worktape to its original state before halting. Usually we consider machines that have access both to a standard worktape and to a "catalytic" worktape. Then:

Theorem 3.8 (derandomization with minimal memory footprint; see [DT23])**.** *For every $\epsilon > 0$ there is $C > 1$ such that the following holds. Suppose that there is a PRG for linear-sized circuits that has arbitrarily long polynomial stretch and is computable in logspace, and that there exists L computable in space $(C+1)\cdot n$ that is hard on almost all input lengths for algorithms that run in space $C \cdot n$ and get $2^{n/2}$ bits of non-uniform advice. Then, for $S(n) = \Omega(\log(n))$ we have $\mathcal{BPSPACE}[S] \subseteq \mathcal{DSPACE}[(2+\epsilon)\cdot S]$. Moreover, if L is computable in space n using additional $(C+1)\cdot n$ auxiliary catalytic space, then $\mathcal{BPSPACE}[S] \subseteq \mathcal{DSPACE}[(1+\epsilon)\cdot S]$.*

The second construction [DT23, Theorem 3] yields the same conclusion using a uniform hardness assumption: A function $f\colon \{0,1\}^n \to \{0,1\}^{n^2}$ computable in space $(C+1)\cdot n$ such that every probabilistic algorithm R that runs in space $C \cdot \log(n) + O(\log(n))$ fails to *compress* $f(x)$ (i.e., to print a small circuit whose truth-table is $f(x)$). This result uses a targeted PRG and an instance-wise hardness-vs-randomness tradeoff, and thus if hardness holds over a certain distribution $\mathbf{x}_n$, we get derandomization that succeeds over precisely the same distribution.

3.2.4 Free lunch from weaker hardness assumptions?

The results in this section give the first theoretical evidence for free lunch theorems, but they rely on strong hardness assumptions. Moreover, they rely on many *different types* of assumptions: For example, non-batch-computable functions (in Theorem 3.5), hardness for non-uniform algorithms with bounded time or space (as in Theorems 3.1, 3.2, 3.8), variants of these assumptions asserting hardness for interactive proofs (as in Theorem 3.7), and hardness of compressing outputs of functions (mentioned after Theorem 3.8). It is reasonable that free lunch theorems will rely on strong hardness assumptions, but can we understand which assumptions are really needed?

Open Problem 3.9. *Prove any of results mentioned in this section using weaker assumptions, or alternatively prove that the assumptions are necessary for the conclusions.*

Some free lunch theorems also relied on *cryptographic assumptions*, such as the existence of one-way functions, which were used to bypass the well-known *hybrid argument* in analyses of PRGs (see, e.g., [DMO+20] for an explanation). An alternative way to bypass the hybrid argument, without using cryptographic assumptions, is to rely on hardness for non-uniform versions of $\mathcal{NP}$ or of $\mathcal{MA}$ (see, e.g., [CT23b]). Intuitively, these assumptions do not seem necessary for free lunch derandomization of probabilistic algorithms.

Open Problem 3.10. *Prove conclusions as in Theorems 3.2 and 3.5 without relying on cryptographic assumptions, or on hardness for non-uniform $\mathcal{NP}$ or $\mathcal{MA}$.*

The crucial missing part is a construction of a (classical) PRG that stretches n^{ϵ} bits to n bits, runs in time $n^{1+\epsilon}$, and is pseudorandom for circuits of size $O(n)$ (for an arbitrarily small $\epsilon > 0$).

A black-box barrier. Shaltiel and Viola [SV22] studied the necessity of the hardness assumptions in Theorems 3.1 and 3.2. They showed that when using certain *black-box techniques*, such as pseudoentropy generators (equivalently, black-box algorithms for quantified derandomization [Tel22, Section 9]) and black-box hardness amplification, standard circuit lower bounds (i.e., functions in time 2^n that are hard for circuits of size $\approx 2^{.99 \cdot n}$) do not suffice for the conclusion.

3.3 Proving that $pr\mathcal{BPP} = pr\mathcal{P}$ from uniform hardness assumptions

If PRGs and the corresponding circuit lower bounds (as in Theorem 2.2) are an overkill for derandomization, what are the *right hardness assumptions* for $pr\mathcal{BPP} = pr\mathcal{P}$? The following observation from [CT21a] paves way for progress:

Observation 3.11. *If $pr\mathcal{BPP} = pr\mathcal{P}$, then for every $c \in \mathbb{N}$ there is a length-preserving function $f\colon \{0,1\}^* \to \{0,1\}^*$ computable in deterministic polynomial time such that for every probabilistic machine M running in time n^c, for* **all but finitely many** *inputs $x \in \{0,1\}^*$ it holds that $\Pr[M(x,r) = f(x)] < 2/3$.*

Proof idea. For every $n \in \mathbb{N}$ and $x \in \{0,1\}^n$ and $i \in [n]$, print an n-bit string such that the i^{th} output bit is different than the one that the i^{th} Turing machine prints on input x. Rely on the fact that $pr\mathcal{BPP} = pr\mathcal{P}$ to deterministically simulate probabilistic machines. ■

We refer to hardness as in Observation 3.11 as hardness on almost all inputs (where "almost all" = "all but finitely many"). Note that the function f has multiple output bits, and this is necessary for it to be hard on almost all inputs.[11] This type of hardness seems to better model what we need in derandomization, compared to circuit lower bounds. Recall that in derandomization we get an input x, and our "adversary" is a probabilistic machine M equipped with x (i.e., we want to "fool" $M(x, \cdot)$). Instead of a circuit lower bound, which is a single truth-table that is hard for all small circuits (representing all possible x's), a function f that is hard on almost all inputs lets us compute a value $f(x)$ that is hard for machines equipped with this particular input x.

Open Problem 3.12. *Prove a converse to Observation 3.11.*

Open Problem 3.12 is our favorite challenge arising from works in recent years. Beyond its elegance as a conjecture, an additional motivation suggesting that it might be provable is a *partial converse* that was shown in [CT21a]:

Theorem 3.13 (derandomization from aai-hardness)**.** *There is a constant $c > 1$ such that the following holds. Assume that there is a length-preserving function f computable by logspace-uniform circuits of polynomial size and depth $d(n) = n^2$, such that f is hard on almost all inputs for probabilistic algorithms running in time n^c. Then, $pr\mathcal{BPP} = pr\mathcal{P}$.*

Note that the only meaningful restriction on f (i.e., the gap between Theorem 3.13 and resolving Open Problem 3.12) is the depth restriction $d(n) = n^2$. That is, without the depth restriction, logspace-uniformity would not be an issue (because functions computable in time T are computable by logspace-uniform circuits of size and depth $\tilde{O}(T)$). Moreover, Theorem 3.13 "scales down" to weak circuit classes, in fact as down as $\mathcal{TC}^0$ (see [CTW23]); and extends to "low-end" parameter settings (i.e., obtaining slow derandomization from weaker lower bounds), albeit with significantly suboptimal parameters (see [CT21a] for discussion).

So far there hasn't been further progress on Open Problem 3.12 directly. However, several works were able to prove *full equivalences* between $pr\mathcal{BPP} = pr\mathcal{P}$ and assumptions that are (arguably) less clean. Most interestingly, these assumptions come from different areas (e.g., cryptography, learning theory, and complexity theory). We describe these recent works in Section 3.3.1, and describe a work dealing with the setting of $pr\mathcal{AM} = pr\mathcal{NP}$ in Section 3.3.2.

[11] For simplicity, we fixed the output length to equal the input length, but any super-constant output length suffices.

3.3.1 Characterizing $pr\mathcal{BPP} = pr\mathcal{P}$ by assumptions from across theory

The first to show an equivalence between $pr\mathcal{BPP} = pr\mathcal{P}$ and natural hardness assumptions were Liu and Pass [LP22a; LP22b], who proved two equivalences with assumptions from *meta-complexity* and from *cryptography*. The first assumption refers to conditional Levin's Kolmogorov complexity. Fixing a universal machine U, denote $\mathsf{cKt}(f, x) = \min_{\Pi\in\{0,1\}^*, t\in\mathbb{N}} \{|\Pi| + \lceil\log(t)\rceil : U(\Pi(x), 1^t) = f\}$. The assumption is for the following problem, denoted $\mathsf{gap\text{-}cKt}[s, S]$:

1. YES inputs are (f, x) such that $|f| = |x|$ and $\mathsf{cKt}(f, x) \leq s(|x|)$.
2. NO inputs are (f, x) such that $|f| = |x|$ and $\mathsf{cKt}(f, x) \geq S(|x|)$.

When $s(n) = O(\log(n))$ the problem $\mathsf{gap\text{-}cKt}[s, S]$ is solvable in deterministic polynomial time, since we can enumerate all size-s programs and run them for 2^s steps. Liu and Pass [LP22a] showed that hardness of $\mathsf{gap\text{-}cKt}[O(\log n), n - 1]$ on almost-all-*conditions* characterizes $pr\mathcal{BPP} = pr\mathcal{P}$:

Theorem 3.14 (almost-all-conditions hardness of $\mathsf{gap\text{-}cKt}$ characterizes $pr\mathcal{BPP} = pr\mathcal{P}$; see [LP22a])**.** *There is a constant $c > 1$ such that $pr\mathcal{BPP} = pr\mathcal{P}$ if and only if the following holds. There is $c' \geq 1$ such that for every probabilistic machine M running in time n^c, for all but finitely many z there exists $x \in \{0,1\}^{|z|}$ such that M fails to compute* $\mathsf{gap\text{-}cKt}[c' \cdot \log(n), n - 1](x, z)$.

Tangentially, Theorem 3.14 fits nicely with two other results of Liu and Pass [LP20; LP22c]. There is a related problem about conditional time-bounded Kolmogorov complexity, denoted $\mathsf{cMK^tP}$, such that worst-case hardness of $\mathsf{cMK^tP}$ characterizes $\mathcal{NP} \not\subseteq \mathcal{BPP}$, certain average-case hardness of $\mathsf{cMK^tP}$ characterizes one-way functions, and almost-all-conditions hardness of $\mathsf{cMK^tP}$ characterizes derandomization![12] We also note that in [LP22a] they showed that it suffices to assume hardness for some fixed ("universal") sequence of conditions $x_1, ..., x_n, ...$ where $|x_i| \in \{0,1\}^i$.

The second characterization is with the notion of leakage-resilience from cryptography (see, e.g., [RS85; Mau92; ISW03; MR04; DP08; AGV09]). Recall that a function is leakage-resilient hard if it is hard to compute even when given "leakage", i.e. a few bits of the output $f(x)$. More precisely, $f\colon \{0,1\}^n \to \{0,1\}^n$ is (ℓ, T)-leakage-resilient hard on input $x \in \{0,1\}^n$ if there is no pair of T-time algorithms M and L such that L gets input $f(x)$ and prints $\ell(n)$ bits $b_1, ..., b_\ell$ of $f(x)$; and M gets input $(x, b_1, ..., b_\ell)$ and prints $f(x)$. Liu and Pass [LP22b] showed:

Theorem 3.15 (almost-all-inputs leakage-resilience characterizes $pr\mathcal{BPP} = pr\mathcal{P}$; see [LP22b])**.** *There is a constant $c > 1$ such that $pr\mathcal{BPP} = pr\mathcal{P}$ if and only if the following holds. There exists $\epsilon \in (0, 1)$ and a length-preserving $f\colon \{0,1\}^* \to \{0,1\}^*$ such that f is compuable in deterministic polynomial time, and is (n^ϵ, n^c)-leakage-resilient hard on almost all inputs.*

[12] The result statements iin [LP22a] refer to $\mathsf{gap\text{-}cKt}$, but the proofs immediately imply similar results for $\mathsf{cMK^tP}$.

Theorems 3.14 and 3.15 were proved in [LP22a; LP22b] by reinterpreting a standard analysis of [IW98] of the classical Nisan-Wigderson [NW94] construction,[13] and recasting the construction as a targeted PRG. The same approach yields a characterization of $pr\mathcal{BPP} = pr\mathcal{P}$ in terms of almost-all-inputs hardness of *learning* the truth-table $f(x)$ (see [CTW23] for details).

Generalizing these results using the terminology of refuters. The result above show that derandomization can be characterized by lower bounds for gap-cKt, by leakage-resilience lower bounds, and by learning lower bounds. It turns out that all these results can be cast as characterizing derandomization by constructive lower bounds (see, e.g., [Kab01; GSTS07; CJS+21]).

Consider a lower bound $f \notin \mathcal{C}$ for some function f and class $\mathcal{C}$. A refuter for f against $\mathcal{C}$ is an algorithm that get as input a description of $C \in \mathcal{C}$ and finds x such that $C(x)$ fails to compute $f(x)$. When an efficient refuter exists, we say that the lower bound $f \notin \mathcal{C}$ is *constructive*. As proved in [CTW23], derandomization is equivalent to *deterministically constructivizing known lower bounds for probabilistic algorithms*. For example, denoting by $\mathsf{str}\mathcal{TISP}[T,S]$ the class of non-uniform probabilistic streaming algorithms running in time T and space S, we have:

Theorem 3.16 (derandomization vs refutation, a special case; see [CTW23])**.** *For any constant $\epsilon > 0$ and function $f\colon \{0,1\}^* \to \{0,1\}^*$ computable in time $T(n) = \mathrm{poly}(n)$ such that there is a $\mathcal{BPP}$-refuter[14] for f against $\mathsf{str}\mathcal{TISP}[T^{1+\epsilon}, n^\epsilon]$, the following are equivalent.*

1. *$pr\mathcal{BPP} = pr\mathcal{P}$.*

2. *There is a refuter in $\mathcal{FP}$ against $\mathsf{str}\mathcal{TISP}[T^{1+\epsilon}, n^\epsilon]$.*

There are decades-old unconditional lower bounds in polynomial time for n^ϵ-space streaming algorithms with *unbounded* running time (see, e.g., [AMS99]), and these lower bounds indeed have $\mathcal{BPP}$-refuters (since the lower bounds hold on average). The point of Theorem 3.16 is that constructing deterministic refuters for such lower bounds is equivalent to derandomization.

Theorem 3.16 (in a more technical form that appears in [CTW23]) generalizes and strengthens Theorems 3.14 and 3.15, as well as the result about learning mentioned after Theorem 3.15 and a result of Korten [Kor22, Theorem 40] (who showed that $pr\mathcal{BPP} = pr\mathcal{P}$ if and only if a certain problem is solvable in deterministic polynomial time). For example, the hardness assumption in Theorem 3.15 is implied by the existence of a refuter for probabilistic one-way communication protocols that run in time n^c and send n^ϵ bits. The assumption in Theorem 3.16 is weaker, since it only requires refuting the weaker class $\mathsf{str}\mathcal{TISP}[n^c, n^\epsilon]$. Details appear in [CTW23].

[13] The analysis of [IW98] shows that the reconstruction algorithm of the Nisan-Wigderson PRG can be viewed as a learning algorithm for the hard function.

[14] A $\mathcal{BPP}$-refuter for f against $\mathcal{C}$ is a probabilistic algorithm that, on input $C \in \mathcal{C}$, with high probability outputs an input x such that $C(x)$ fails to compute $f(x)$

Consequences: Amplification theorems, and a new network of connections. An obvious consequence of all the equivalences listed in this section is that the mentioned problems are equivalent not only to derandomization, but *they are also equivalent to each other*. For example, refuters for streaming algorithms (as in Theorem 3.16) are equivalent to almost-all-conditions hardness of $\mathsf{gap}\text{-}\mathsf{cKt}$ (as in Theorem 3.14), which is equivalent to almost-all-input leakage-resilience (as in Theorem 3.15). This is a new and interesting network of equivalences between notions from various theoretical areas that is centered around the $pr\mathcal{BPP} = pr\mathcal{P}$ conjecture.

An additional implication (which wasn't mentioned above) is *amplification theorems* that follow from these results. As one example, Liu and Pass [LP22b] showed that leakage-resilience with n^{ϵ} bits of leakage is equivalent to derandomization, which in turn is equivalent to leakage-resilience with $n^{1-\epsilon}$ bits of leakage. More generally, in [CTW23] it is shown that refuters for *weak classes* (e.g., streaming algorithms) imply refuters for *stronger classes* (e.g., general RAMs). At the moment, we only know how to prove these amplification theorems by proving equivalences to derandomization.

3.3.2 The $\mathcal{AM}$ setting

Can we prove that $pr\mathcal{AM} = pr\mathcal{NP}$ from weaker assumptions than the classical circuit lower bounds in [KM02; MV05; SU05]? In this setting we can do significantly better than in the $\mathcal{BPP}$ setting, and almost get a full equivalence. Specifically, while we still don't know if targeted HSGs are the right object of study in this setting, Sdroievski and van Melkebeek [SM23] managed to show a near-equivalence between derandomization of $pr\mathcal{AM}$ and an almost-all-inputs hardness assumption.

Theorem 3.17 (near-equivalence between derandomization of $pr\mathcal{AM}$ and almost-all-inputs hardness; see [SM23])**.** *The following two statements hold:*

1. *If $pr\mathcal{AM} = pr\mathcal{NP}$, then for every $c \in \mathbb{N}$ there is a length-preserving function that is computable in non-deterministic polynomial time with $\log^*(n)$ bits of advice and that is hard on almost all inputs for $\mathcal{AMTIME}[n^c]$.*[15]

2. *If there exists a length-preserving function that is computable in non-deterministic time n^a and that is hard on almost all inputs for $pr\mathcal{AMTIME}[n^{O(\log a)^2}]$, then $pr\mathcal{AM} = pr\mathcal{NP}$.*

The only two gaps between Theorem 3.17 and a full equivalence are the $\log^*(n)$ bits of advice, and the fact that the hardness in one direction is for $pr\mathcal{AM}$ whereas the hardness in the other direction is for $\mathcal{AM}$. These seem smaller than the gap between Observation 3.11 and Theorem 3.13.

[15] The advice length can be any increasing function, rather than $\log^*(n)$ specifically.

4 Broader effects: Explicit constructions, cryptography, algorithms, interactive proof systems

Given that classical PRGs have a myriad of applications, one can expect the new constructions of targeted PRGs (and targeted HSGs) to have significant broader effects. In this section we mention several applications known so far:

1. In Section 4.1 we describe a new algorithm for **finding prime numbers**.
2. In Section 4.2 we describe interactions with **cryptography**: Specifically, new approaches towards the Fiat-Shamir heuristic, and new types of argument systems.
3. In Section 4.3 we describe a new conditional construction of a fast **algorithm for solving a $\#\mathcal{P}$-hard problem**, in the model of computationally sound $\mathcal{NP}$.
4. In Section 4.4 we describe interactions between the area of **proof systems** and new constructions of targeted PRGs and targeted HSGs.
5. In Section 4.5 we describe an application to the classical open problem of **uniform hardness vs randomness** tradeoffs.

4.1 Finding prime numbers

It is 2023 and we still don't know how to efficiently find prime numbers, i.e. get input 1^n and deterministically print a prime number $p \in [2^n, 2^{n+1})$. Of course, using randomness, we can just try $O(n)$ uniformly chosen n-bit integers (relying on the density of primes), testing each one for primality. But this doesn't guarantee that we'll get the *same* prime in each execution.

Pseudodeterministic algorithms, introduced in [GG11], are randomized algorithms that produce a fixed answer, with high probability (i.e., for every x there exists y such that $\Pr_r[A(x,r) = y] \geq 2/3$). Oliveira and Santhanam [OS17] used classical PRG constructions to build a pseudodeterministic algorithm that, for infinitely many $n \in \mathbb{N}$, finds an n-bit prime in subexponential time. A recent work by the first author, Lu, Oliveira, Ren, and Santhanam [CLO+23] was able to do better:

Theorem 4.1 (pseudodeterministically finding primes in polynomial time; see [CLO+23])**.** *There is a probabilistic polynomial-time algorithm A such that for infinitely many $n \in \mathbb{N}$, there exists an n-bit prime $p = p(n)$ for which $\Pr_r[A(1^n, r) = p(n)] \geq 2/3$.*

We stress that the algorithm in Theorem 4.1 is *unconditional*. (In contrast, a conditional "superfast" algorithm for a broad class of explicit construction problem appears in [CT21a, Corollary 1.9].) The proof of Theorem 4.1 in [CLO+23] uses a refined version of a targeted HSG construction from [CT21a], and applies it recursively along with a clever win-win analysis.

4.2 Cryptography: Fiat-Shamir, new argument systems, targeted PRGs vs hash functions

The study of PRGs in derandomization originated from cryptography, and some of the fast derandomization results (and free lunch theorems) mentioned in this column rely on cryptographic assumptions (see Theorems 3.2 and 3.5). Let us now focus on the *reverse* direction, in which the new developments in derandomization affect cryptography. To do so, recall Theorem 3.7:

Theorem 4.2 (free lunch derandomization of $\mathsf{de}\mathcal{IP}$; informal, see [CT23b])**.** *Under strong hardness assumptions, for every $\epsilon > 0$ and $c \in \mathbb{N}$ and "nice" time bound $T(n)$, for every language L that is decidable by a doubly efficient interactive proof system with a verifier running in time T and an honest prover running in time $\mathrm{poly}(T)$ and c turns of interaction, it holds that $L \in \mathsf{cs}\text{-}\mathcal{NTIME}[T^{1+\epsilon}]$.*

The derandomization in Theorem 4.2 eliminates the random coins of the verifier in the proof system for L, and when doing so it also eliminates the need for *interaction* between the verifier and the honest prover. This is reminiscent of the classical Fiat-Shamir heuristic [FS86] in cryptography, which is a technique to eliminate interaction in certain classes of proof systems (for recent works see, e.g., [PS19; CCH+19; HLR21; CJJ21]).

The similarity to Fiat-Shamir is substantial. First, both in Fiat-Shamir and in Theorem 4.2, we compile a *proof system* (with information-theoretic soundness) into an *argument system* (with computational soundness). Secondly, even the construction in Theorem 4.2 is reminiscent of Fiat-Shamir: In the latter, we apply a *hash function* to the transcript at each round to deterministically obtain the verifier's next message; in the former, we apply a *targeted PRG* to the transcript at each round to deterministically obtain the verifier's next message (see [CT23b] for further details).

Nevertheless, the analysis in the proof of Theorem 4.2 is completely different from the analyses in known works concerning Fiat-Shamir (e.g., in [CJJ21]), and it uses primarily derandomization-based techniques. Moreover, Theorem 4.2 only relies on complexity-theoretic assumptions, which are not known to imply even one-way functions.[16] This raises the following questions:

Open Problem 4.3. *Can we analyze the Fiat-Shamir heuristic in more settings (beyond the one in Theorem 4.2) using complexity-theoretic assumptions and analyses? Can we replace correlation intractable hash functions (which are the technical tool underlying analyses of Fiat-Shamir) with targeted PRGs in other settings in cryptography, beyond Fiat-Shamir?*

Argument systems with soundness against uniform adversaries. Another interesting element in Theorem 4.2 is the introduction (and application) of argument systems

[16] Additionally, the $\mathcal{NP}$-type verifier in Theorem 4.2 is fully deterministic, and does not use a *common random string* as in Fiat-Shamir; soundness in Theorem 4.2 is only against uniform adversaries, rather than the usual notion of non-uniform soundness in cryptography; and there are minor technical differences in the construction.

where the adversary is a *uniform algorithm*. Recall, from Definition 3.6 that the soundness requirement is that no uniform algorithm can find an input x and proof π that mislead the verifier (i.e., $x \notin L$ but the verifier accepts (x, π)). This is a natural relaxation, and we suspect that such argument systems might find further applications in cryptography.

4.3 Algorithms: Efficiently certifying #SAT with computational soundness

It is widely conjectured that 3-SAT for n-bit formulas of size $O(n)$ cannot be solved in time $2^{\epsilon \cdot n}$ for an *arbitrarily small* constant $\epsilon > 0$; this is the Exponential-Time Hypothesis (ETH) [IP01; IPZ01]. A well-studied extension asserts that even co-nondeterministic machines cannot achieve such a running time; this is the Non-Deterministic Exponential-Time Hypothesis (NETH) [CGI+16].

It turns out that, under strong assumptions, the harder problem #SAT can be solved in non-deterministic time $2^{\epsilon \cdot n}$, for an arbitrarily small $\epsilon > 0$, *if we relax the soundness requirement to be computational*; that is, it can be solved in *computationally sound* $\mathcal{NTIME}$ as in Definition 3.6.

Theorem 4.4 (efficiently certifying #SAT with computational soundness; see [CT23b]). *Under strong hardness assumptions, for every $\epsilon > 0$ there is a deterministic verifier V that gets as input an n-bit formula Φ of size at most $2^{o(n)}$, runs in time $2^{\epsilon \cdot n}$, and satisfies:*

1. **(Completeness.)** *There is a prover that, given any Φ as above, runs in time $2^{O(n)}$ and outputs a proof π such that $V(\Phi, \pi) = \#\mathsf{SAT}(\Phi)$.*
2. **(Computational soundness.)** *For every adversary $\hat{P}$ running in time $2^{O(n)}$, the probability that $\hat{P}(1^n)$ prints an n-bit formula Φ of size $2^{o(n)}$ and a proof π such that $V(\Phi, \pi) \notin \{\#\mathsf{SAT}(\Phi), \perp\}$ is $2^{-\omega(n)}$.*

Theorem 4.4 is obtained by applying the free lunch theorem in Theorem 3.7 to an interactive proof system for #SAT by Williams [Wil16]. It is natural to ask if we can do better:

Open Problem 4.5. *Can we solve* #SAT *for n-bit formulas of size* $\mathrm{poly}(n)$ *in computationally sound $\mathcal{NP}$ (i.e., with a* $\mathrm{poly}(n)$*-time verifier whose soundness is computational)?*

More broadly, the model of computationally sound $\mathcal{NP}$ is natural, and has not been studied before. We believe that other interesting problems can be solved quickly in this model.

4.4 Proof systems vs targeted PRGs

Beyond the immediate implications of Theorems 3.7 and 3.17 to interactive proof systems, another connection between the two areas came in the form of **constructions of targeted HSGs based on proof systems**. We will be brief, due to space considerations.

The construction of a targeted HSG in [CT21a] is based on the doubly efficient interactive proofs of Goldwasser, Kalai, and Rothblum [GKR15]. Loosely speaking, the targeted HSG gets input x and uses the *prover strategy functions* (thought of as "hard functions") to produce pseudorandom strings. We show that if an efficient machine can break the targeted HSG at input x, then one can efficiently *compute compressed versions of the prover strategy functions*, round-by-round, starting from the last round and ending in the first round (in which the value $f(x)$) is asserted.

Similarly, the construction of a targeted HSG for co-nondeterministic circuits by Sdroievski and van Melkebeek [SM23] is based on PCPs. Loosely speaking, the targeted HSG on input x considers a set of PCP witnesses, one for each output bit of $f(x)$, and uses the witnesses (thought of as "hard truth-tables") to produce pseudorandom strings.

Open Problem 4.6. *Discover more constructions of targeted HSGs and targeted PRGs that are based on proof systems.*

4.5 Uniform hardness vs randomness

An additional application of the new constructions of targeted PRGs and targeted HSGs concerns a long-standing open problem in derandomization: Proving that $\mathcal{BPP} = \mathcal{P}$ "on average" from *worst-case hardness assumptions for uniform algorithms*.[17] The idea here is to start from mild, natural and appealing hardness assumptions, and still obtain interesting derandomization results.

The work of Impagliazzo and Wigderson [IW98] raises the possibility we might be able to deduce that $\mathcal{BPP} = \mathcal{P}$ on average from the assumption that $\mathcal{E}$ is hard for probabilistic algorithms running in time $2^{\epsilon \cdot n}$, for some $\epsilon > 0$ (on almost all input lengths). Progress on this challenge has been slow (see, e.g., [IW98; CNS99; Kab01; TV07; GV08; CIS18; CRT+20; CRT22]), but recently, using the construction of a targeted HSG from [CT21a] and new constructions of *instance checkers*, the authors and Ron Rothblum proved a significant step towards this goal (see [CRT22] for details).

In addition, we can deduce average-case derandomization from *fine-grained hardness assumptions for natural problems in $\mathcal{P}$*. Results of this form were first proved by Carmosino, Impagliazzo, and Sabin [CIS18], and the authors and Ron Rothblum [CRT22] used targeted HSGs to prove results that rely on significantly weaker hypotheses; for example:

Theorem 4.7 (average-case derandomization from a natural weak hardness assumption; see [CRT22])**.** *Assume that for every $c \in \mathbb{N}$ there is $k = k(c) \in \mathbb{N}$ such that counting k-cliques is hard for probabilistic time n^c on almost all input lengths. Then, $\mathcal{RP} = \mathcal{P}$ on average.*

[17] For simplicity, "on average" in this context means that every problem in $\mathcal{BPP}$ can be solved by a deterministic polynomial-time algorithm that is correct on $1 - 1/n$ of the inputs, over the uniform distribution.

The hypothesis in Theorem 4.7 is natural and appealing, and yields a surprisingly strong conclusion. The result in [CRT22] is more general, and deduces the conclusion not only from hardness of counting k-cliques, but from hardness of any problem that is solvable by a (family of) logspace-uniform arithmetic formulas of polynomial size and degree n^2.

5 A roadmap for open problems

We conclude by reminding the reader of open problems that were posed throughout this column, and hopefully presenting them in a more orderly way.

Theoretical foundations for free lunch theorems. The known free lunch theorems for derandomization do not yet form a sufficiently coherent and established theory, because we do not understand their assumptions well enough. Open Problems 3.9 and 3.10 in Section 3.2.4 suggest two paths for progress. Another challenge is understanding whether it is indeed necessary to pay an overhead of $T \mapsto n \cdot T$ when derandomizing in the worst-case, and one potential path for progress is understanding the complexity of the problem TAMP, mentioned in the beginning of Section 3.2.

Non-black-box hardness vs randomness. Open Problem 3.12 calls for proving an equivalence between derandomization and almost-all-inputs hardness. There are several problems that can serve as stepping-stones towards this goal. For example, can we prove a result as in Theorem 3.13 where the hard function isn't required to be computable in depth n^2, but only with n^ϵ bits of space? The intuition is that Theorem 3.13 was proved relying on the proof system of [GKR15] for functions computable in low depth, and Reingold, Rothblum, and Rothblum [RRR18] constructed a similar proof system for functions computable in low space.

Additionally, as mentioned after Theorem 3.13, the result does not scale well to low-end setting. One reason is that construction underlying Theorem 3.13 is of a *targeted HSG* rather than of a targeted PRG.[18] Constructing a targeted PRG from the hypothesis is an open problem, and seemingly calls for constructing an object called computational mergers (see [CT21a] for a discussion).

We also remind the reader of Open Problem 2.5 (posed in [Gol11]), which asks whether derandomization of $pr\mathcal{AM}$ necessitates constructing targeted HSGs (for co-nondeterministic circuits).

[18] A targeted HSG yields derandomization of algorithms with one-sided error, and by known reductions [Lau83; Sip83; BF99] this yields derandomization of algorithms with two-sided error. These reductions incur significant time overheads when scaling the result to low-end settings. (There are known reductions of derandomization with two-sided error to derandomization with one-sided error that do not incur significant runtime overheads (see, e.g., [GVW11]), but these only apply to black-box derandomization (e.g., with HSGs), whereas Theorem 3.13 uses targeted HSGs.)

Broader effects of targeted PRG. In Section 4 we posed problems that call for applying targeted PRGs more broadly in theoretical computer science, and for further pursuing results that were recently obtained in such applications. For example, Open Problem 4.5 calls for further study of the model of *computationally sound* $\mathcal{NP}$, and in particular for trying to quickly solve hard problems (such as #SAT) in this model.

In addition, Open Problem 4.3 calls for analyzing the Fiat-Shamir heuristic in cryptography using derandomization-based tools, and more generally to try and replace correlation intractable hash functions with targeted PRGs in other cryptographic settings. After posing it we also suggested studying the potential of argument systems that are sound only against *uniform* adversaries.

The challenge posed in Open Problem 4.6, of constructing additional targeted PRGs and targeted HSGs from interactive proof systems, might be related to the challenge of making progress on Open Problem 3.12 (since using different proof systems for constructing targeted PRGs/HSGs might allow using hard functions with different properties).

And, as mentioned in Sections 2.1.2 and 4.5, the classical line of works concerning uniform hardness vs randomness is not complete yet, and using targeted PRGs in this context seems beneficial.

Are PRGs necessary for derandomization? Lastly, we cannot conclude the survey without mentioning the decades-old open problem of proving that derandomization *necessitates* classical PRGs (and the circuit lower bounds that are equivalent to classical PRGs). For example, can we prove the statement "$pr\mathcal{BPP} = pr\mathcal{P} \implies \mathcal{EXP} \not\subset \mathcal{P}/\text{poly}$"?

A new challenge for proving such a result, which arises from the works in recent years, is that such a result would imply that *lower bounds on almost all inputs imply circuit lower bounds*. (For example, by Theorem 3.13, or more generally if Open Problem 3.12 is resolved.) The former type of lower bound seems intuitively weaker than the latter.

Acknowledgments

We are very grateful to Dean Doron, who contributed several useful comments on an early draft of this column. We also thank Ben Lee Volk for the invitation to write this column for SIGACT News, and for reading the text and pointing out inaccuracies and typos. Lijie Chen is supported by a Miller Research Fellowship. Roei Tell is supported by NSF Grants CCF-1445755 and CCF-1900460.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: A modern approach*. Cambridge University Press, Cambridge, 2009.

[Adl78] Leonard Adleman. "Two theorems on random polynomial time". In: *Proc. 19th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 1978, pp. 75–83.

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. "Simultaneous hardcore bits and cryptography against memory attacks". In: *Theory of cryptography*. Vol. 5444. 2009, pp. 474–495.

[AKL+79] Romas Aleliunas, Richard M. Karp, Richard J. Lipton, László Lovász, and Charles Rackoff. "Random walks, universal traversal sequences, and the complexity of maze problems". In: *Proc. 20th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 1979, pp. 218–223.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. "PRIMES is in P". In: *Annals of Mathematics. Second Series* 160.2 (2004), pp. 781–793.

[AM77] Leonard M. Adleman and Kenneth L. Manders. "Reducibility, Randomness, and Intractability". In: *Proc. 9th Annual ACM Symposium on Theory of Computing (STOC)*. 1977, pp. 151–163.

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. "The space complexity of approximating the frequency moments". In: vol. 58. 1, part 2. Twenty-eighth Annual ACM Symposium on the Theory of Computing (Philadelphia, PA, 1996). 1999, pp. 137–147.

[Bar02] Boaz Barak. "A probabilistic-time hierarchy theorem for "slightly non-uniform" algorithms". In: *Proc. 6th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*. Vol. 2483. 2002, pp. 194–208.

[BF99] Harry Buhrman and Lance Fortnow. "One-Sided Versus Two-Sided Error in Probabilistic Computation". In: *Proc. 16th Symposium on Theoretical Aspects of Computer Science (STACS)*. 1999, pp. 100–109.

[BFN+93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. "BPP has subexponential time simulations unless EXPTIME has publishable proofs". In: *Computational Complexity* 3.4 (1993), pp. 307–318.

[BG81] Charles H. Bennett and John Gill. "Relative to a random oracle A, $\mathbf{P}^A \neq \mathbf{NP}^A \neq \text{co} - \mathbf{NP}^A$ with probability 1". In: *SIAM Journal of Computing* 10.1 (1981), pp. 96–113.

[BM84] Manuel Blum and Silvio Micali. "How to Generate Cryptographically Strong Sequences of Pseudo-random Bits". In: *SIAM Journal of Computing* 13.4 (1984), pp. 850–864.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. "Fiat-Shamir: from practice to theory". In: *Proc. 51st Annual ACM Symposium on Theory of Computing (STOC)*. 2019, pp. 1082–1090.

[CGI+16] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. "Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility". In: *Proc. 7th Conference on Innovations in Theoretical Computer Science (ITCS)*. 2016, pp. 261–270.

[Che19] Lijie Chen. "Non-deterministic Quasi-Polynomial Time is Average-case Hard for ACC Circuits". In: *Proc. 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2019, pp. 1281–1304.

[CIS18] Marco L. Carmosino, Russell Impagliazzo, and Manuel Sabin. "Fine-grained derandomization: from problem-centric to resource-centric complexity". In: *Proc. 45th International Colloquium on Automata, Languages and Programming (ICALP)*. 2018, 27:1–27:16.

[CJJ21] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. "SNARGs for $\mathcal{P}$ from LWE". In: *Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2021, pp. 68–79.

[CJS+21] Lijie Chen, Ce Jin, Rahul Santhanam, and Ryan Williams. "Constructive Separations and Their Consequences". In: *Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2021, pp. 646–657.

[CLO+23] Lijie Chen, Zhenjian Lu, Igor Carboni Oliveira, Hanlin Ren, and Rahul Santhanam. *Polynomial-Time Pseudodeterministic Construction of Primes*. Under Submission. 2023.

[CLW20] Lijie Chen, Xin Lyu, and Richard Ryan Williams. "Almost-Everywhere Circuit Lower Bounds from Non-Trivial Derandomization". In: *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 1–12.

[CNS99] Jin-Yi Cai, Ajay Nerurkar, and D. Sivakumar. "Hardness and hierarchy theorems for probabilistic quasi-polynomial time". In: *Proc. 31st Annual ACM Symposium on Theory of Computing (STOC))*. 1999, pp. 726–735.

[CR20] Lijie Chen and Hanlin Ren. "Strong average-case lower bounds from non-trivial derandomization". In: *Proc. 52nd Annual ACM Symposium on Theory of Computing (STOC)*. 2020, pp. 1327–1334.

[CRT+20] Lijie Chen, Ron D. Rothblum, Roei Tell, and Eylon Yogev. "On Exponential-Time Hypotheses, Derandomization, and Circuit Lower Bounds". In: *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 13–23.

[CRT22] Lijie Chen, Ron D. Rothblum, and Roei Tell. "Unstructured Hardness to Average-Case Randomness". In: *Proc. 63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2022.

[CT21a] Lijie Chen and Roei Tell. “Hardness vs Randomness, Revised: Uniform, Non-Black-Box, and Instance-Wise”. In: *Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2021, pp. 125–136.

[CT21b] Lijie Chen and Roei Tell. “Simple and fast derandomization from very hard functions: Eliminating randomness at almost no cost”. In: *Proc. 53st Annual ACM Symposium on Theory of Computing (STOC)*. 2021, pp. 283–291.

[CT23a] Lijie Chen and Roei Tell. “Guest Column: New Ways of Studying the BPP = P Conjecture”. In: *SIGACT News* 54.2 (2023), 44–69.

[CT23b] Lijie Chen and Roei Tell. “When Arthur has Neither Random Coins nor Time to Spare: Superfast Derandomization of Proof Systems”. In: *Proc. 55th Annual ACM Symposium on Theory of Computing (STOC)*. 2023.

[CTW23] Lijie Chen, Roei Tell, and R. Ryan Williams. *Derandomization vs Refutation: A Unified Framework for Characterizing Derandomization*. Under Submission. 2023.

[DMO+20] Dean Doron, Dana Moshkovitz, Justin Oh, and David Zuckerman. “Nearly Optimal Pseudorandomness From Hardness”. In: *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 1057–1068.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. “Leakage-Resilient Cryptography”. In: *Proc. 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2008, pp. 293–302.

[DT23] Dean Doron and Roei Tell. “Derandomization with Minimal Memory Footprint”. In: *Electronic Colloquium on Computational Complexity: ECCC* 30 (2023), p. 036.

[FGM+89] Martin Fürer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. “On Completeness and Soundness in Interactive Proof Systems”. In: *Advances in Computing Research* 5 (1989).

[FS86] Amos Fiat and Adi Shamir. “How to prove yourself: practical solutions to identification and signature problems”. In: *Advances in cryptology—CRYPTO*. 1986, pp. 186–194.

[GG11] Eran Gat and Shafi Goldwasser. “Probabilistic search algorithms with unique answers and their cryptographic applications”. In: *Electronic Colloquium on Computational Complexity: ECCC* 18 (2011), p. 136.

[Gil77] John Gill. “Computational complexity of probabilistic Turing machines”. In: *SIAM Journal of Computing* 6.4 (1977), pp. 675–695.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. “Delegating computation: interactive proofs for muggles”. In: *Journal of the ACM* 62.4 (2015), 27:1–27:64.

[Gol08] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. New York, NY, USA: Cambridge University Press, 2008.

[Gol11] Oded Goldreich. "In a World of P=BPP". In: *Studies in Complexity and Cryptography. Miscellanea on the Interplay Randomness and Computation*. 2011, pp. 191–232.

[GSTS03] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. "Uniform hardness versus randomness tradeoffs for Arthur-Merlin games". In: *Computational Complexity* 12.3-4 (2003), pp. 85–130.

[GSTS07] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. "If NP languages are hard on the worst-case, then it is easy to find their hard instances". In: *Computational Complexity* 16.4 (2007), pp. 412–441.

[GV08] Dan Gutfreund and Salil Vadhan. "Limitations of hardness vs. randomness under uniform reductions". In: *Proc. 12th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*. 2008, pp. 469–482.

[GVW11] Oded Goldreich, Salil Vadhan, and Avi Wigderson. "Simplified derandomization of BPP using a hitting set generator". In: *Studies in complexity and cryptography*. Vol. 6650. Lecture Notes in Computer Science. Springer, Heidelberg, 2011, pp. 59–67.

[HLR21] Justin Holmgren, Alex Lombardi, and Ron D. Rothblum. "Fiat-Shamir via list-recoverable codes (or: parallel repetition of GMW is not zero-knowledge)". In: *Proc. 53rd Annual ACM Symposium on Theory of Computing (STOC)*. 2021, pp. 750–760.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. "In search of an easy witness: exponential time vs. probabilistic polynomial time". In: *Journal of Computer and System Sciences* 65.4 (2002), pp. 672–694.

[IP01] Russell Impagliazzo and Ramamohan Paturi. "On the complexity of k-SAT". In: *Journal of Computer and System Sciences* 62.2 (2001), pp. 367–375.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. "Which problems have strongly exponential complexity?" In: *Journal of Computer and System Sciences* 63.4 (2001), pp. 512–530.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. "Private circuits: securing hardware against probing attacks". In: *Advances in cryptology—CRYPTO*. 2003, pp. 463–481.

[IW97] Russell Impagliazzo and Avi Wigderson. "P $=$ BPP if E requires exponential circuits: derandomizing the XOR lemma". In: *Proc. 29th Annual ACM Symposium on Theory of Computing (STOC)*. 1997, pp. 220–229.

[IW98] Russell Impagliazzo and Avi Wigderson. "Randomness vs. Time: De-Randomization under a Uniform Assumption". In: *Proc. 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 1998, pp. 734–743.

[Kab01] Valentine Kabanets. “Easiness assumptions and hardness tests: trading time for zero error”. In: *Journal of Computer and System Sciences* 63.2 (2001), pp. 236–252.

[KM02] Adam R. Klivans and Dieter van Melkebeek. “Graph Nonisomorphism Has Subexponential Size Proofs Unless the Polynomial-Time Hierarchy Collapses”. In: *SIAM J. Comput.* 31.5 (2002), pp. 1501–1526.

[Kor22] Oliver Korten. “Derandomization from time-space tradeoffs”. In: *Proc. 37th Annual IEEE Conference on Computational Complexity (CCC)*. Vol. 234. LIPIcs. Leibniz Int. Proc. Inform. 2022, Art. No. 37, 26.

[Lau83] Clemens Lautemann. “BPP and the polynomial hierarchy”. In: *Information Processing Letters* 17.4 (1983), pp. 215–217.

[LP20] Yanyi Liu and Rafael Pass. “On one-way functions and Kolmogorov complexity”. In: *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 1243–1254.

[LP22a] Yanyi Liu and Rafael Pass. “Characterizing derandomization through hardness of Levin-Kolmogorov complexity”. In: *Proc. 37th Annual IEEE Conference on Computational Complexity (CCC)*. Vol. 234. LIPIcs. Leibniz Int. Proc. Inform. 2022, Art. No. 35, 17.

[LP22b] Yanyi Liu and Rafael Pass. “Leakage-Resilient Hardness v.s. Randomness”. In: *Electronic Colloquium on Computational Complexity: ECCC* TR22-113 (2022).

[LP22c] Yanyi Liu and Rafael Pass. “On One-Way Functions from NP-Complete Problems”. In: *Proc. 37th Annual IEEE Conference on Computational Complexity (CCC)*. 2022, 36:1–36:24.

[Mau92] Ueli M. Maurer. “Factoring with an oracle”. In: *Proc. Advances in cryptology (EUROCRYPT)*. 1992, pp. 429–436.

[Mic00] Silvio Micali. “Computationally sound proofs”. In: *SIAM Journal of Computing* 30.4 (2000), pp. 1253–1298.

[MR04] Silvio Micali and Leonid Reyzin. “Physically observable cryptography (extended abstract)”. In: *Theory of cryptography*. 2004, pp. 278–296.

[MV05] Peter Bro Miltersen and N. V. Vinodchandran. “Derandomizing Arthur-Merlin games using hitting sets”. In: *Computational Complexity* 14.3 (2005), pp. 256–279.

[MW18] Cody Murray and R. Ryan Williams. “Circuit Lower Bounds for Nondeterministic Quasi-Polytime: An Easy Witness Lemma for NP and NQP”. In: *Proc. 50th Annual ACM Symposium on Theory of Computing (STOC)*. 2018, pp. 890–901.

[Nis91] Noam Nisan. “Pseudorandom bits for constant depth circuits”. In: *Combinatorica* 11.1 (1991), pp. 63–70.

[NW94] Noam Nisan and Avi Wigderson. “Hardness vs. randomness”. In: *Journal of Computer and System Sciences* 49.2 (1994), pp. 149–167.

[OS17] Igor C. Oliveira and Rahul Santhanam. “Pseudodeterministic constructions in subexponential time”. In: *Proc. 49th Annual ACM Symposium on Theory of Computing (STOC)*. 2017, pp. 665–677.

[PS19] Chris Peikert and Sina Shiehian. “Noninteractive Zero Knowledge for NP from (Plain) Learning with Errors”. In: *Advances in Cryptology - CRYPTO*. 2019, pp. 89–114.

[Rei08] Omer Reingold. “Undirected connectivity in log-space”. In: *J. ACM* 55.4 (2008), 17:1–17:24.

[RRR18] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. “Efficient Batch Verification for UP”. In: *Proc. 33rd Annual IEEE Conference on Computational Complexity (CCC)*. 2018, 22:1–22:23.

[RS85] Ronald L. Rivest and Adi Shamir. “Efficient factoring based on partial information”. In: *Proc. Advances in cryptology (EUROCRYPT)*. 1985, pp. 31–34.

[Sha81] Adi Shamir. “On the generation of cryptographically strong pseudorandom sequences”. In: *Automata, languages and programming (Akko, 1981)*. Vol. 115. Lecture Notes in Comput. Sci. 1981, pp. 544–550.

[Sip83] Michael Sipser. “A complexity theoretic approach to randomness”. In: *Proc. 15th Annual ACM Symposium on Theory of Computing (STOC)*. 1983, pp. 330–335.

[SM23] Nicollas Sdroievski and Dieter van Melkebeek. “Instance-Wise Hardness versus Randomness Tradeoffs for Arthur-Merlin Protocols”. In: *Electronic Colloquium on Computational Complexity: ECCC* 30 (2023), p. 029.

[STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. “Pseudorandom generators without the XOR lemma”. In: *Journal of Computer and System Sciences* 62.2 (2001), pp. 236–266.

[SU05] Ronen Shaltiel and Christopher Umans. “Simple extractors for all min-entropies and a new pseudorandom generator”. In: *Journal of the ACM* 52.2 (2005), pp. 172–216.

[SU07] Ronen Shaltiel and Christopher Umans. “Low-end uniform hardness vs. randomness tradeoffs for AM”. In: *Proc. 39th Annual ACM Symposium on Theory of Computing (STOC)*. 2007, pp. 430–439.

[SV22] Ronen Shaltiel and Emanuele Viola. “On hardness assumptions needed for “extreme high-end” PRGs and fast derandomization”. In: *Proc. 13th Conference on Innovations in Theoretical Computer Science (ITCS)*. 2022, Art. No. 116, 17.

[Tel22] Roei Tell. "Quantified derandomization: how to find water in the ocean". In: *Foundations and Trends® in Theoretical Computer Science* 15.1 (2022), Paper No 1, 125.

[TSZS06] Amnon Ta-Shma, David Zuckerman, and Shmuel Safra. "Extractors from Reed-Muller codes". In: *Journal of Computer and System Sciences* 72.5 (2006), pp. 786–812.

[TV07] Luca Trevisan and Salil P. Vadhan. "Pseudorandomness and Average-Case Complexity Via Uniform Reductions". In: *Computational Complexity* 16.4 (2007), pp. 331–364.

[Uma03] Christopher Umans. "Pseudo-random generators for all hardnesses". In: *Journal of Computer and System Sciences* 67.2 (2003), pp. 419–440.

[Wig19] Avi Wigderson. *Mathematics and computation*. A theory revolutionizing technology and science. Princeton University Press, Princeton, NJ, 2019, pp. xiii+418. ISBN: 978-0-691-18913-0.

[Wil13] Ryan Williams. "Improving Exhaustive Search Implies Superpolynomial Lower Bounds". In: *SIAM Journal of Computing* 42.3 (2013), pp. 1218–1244.

[Wil16] Richard Ryan Williams. "Strong ETH breaks with Merlin and Arthur: short non-interactive proofs of batch evaluation". In: *Proc. 31st Annual IEEE Conference on Computational Complexity (CCC)*. Vol. 50. 2016, 2:1–2:17.

[Yao82] Andrew C. Yao. "Theory and Application of Trapdoor Functions". In: *Proc. 23rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 1982, pp. 80–91.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il