Electronic Colloquium on Computational Complexity, Report No. 148 (2020)

Simple and fast derandomization from very hard functions: Eliminating randomness at almost no cost

Lijie Chen [*] Roei Tell [†]

September 27, 2020

Abstract

Extending the classical "hardness-to-randomness" line-of-works, Doron et al. (FOCS 2020) recently proved that derandomization with *near-quadratic time overhead* is possible, under the assumption that there exists a function in $\mathcal{DTIME}[2^n]$ that cannot be computed by randomized SVN circuits of size $2^{(1-\epsilon)\cdot n}$ for a small ϵ.

In this work we extend their inquiry and answer several open questions that arose from their work. Our main result is that **derandomization with almost no time overhead is possible**, under a plausible hypothesis. Specifically, we show that probabilistic algorithms that run in time $T(n)$ can be deterministically simulated in time $n \cdot T(n)^{1+\epsilon}$, under a hypothesis that is formally incomparable to the one of Doron et al., but is arguably more standard: We assume that there exist non-uniformly secure one-way functions, and that for $\delta = \delta(\epsilon)$ and $k = k_T(\epsilon)$ there exists a problem in $\mathcal{DTIME}[2^{k\cdot n}]$ that is hard for algorithms that run in time $2^{(k-\delta)\cdot n}$ and use $2^{(1-\delta)\cdot n}$ bits of advice. We also show that the latter hypothesis (or, more accurately, a relaxation of it that we use) is in fact *necessary* to obtain the derandomization conclusion if one relies on a PRG construction (as is indeed our approach).

For sub-exponential time functions $T(n) = 2^{n^{o(1)}}$ we further improve the derandomization time to $n^{1+\epsilon} \cdot T(n)$, under a mildly stronger hypothesis. We also show that **the multiplicative time overhead of n is essentially optimal**, conditioned on a counting version of the non-deterministic strong exponential-time hypothesis (i.e., on #NSETH). Nevertheless, we show that **in the average-case setting a faster derandomization is possible**: Under hypotheses similar to the ones in our main result, we show that for every $L \in \mathcal{BPTIME}[n^k]$ there exists a deterministic algorithm A_L running in time $n^{\epsilon} \cdot n^k$ such that for every distribution $\mathcal{D}$ over $\{0,1\}^n$ samplable in time n^k it holds that $\Pr_{x\sim\mathcal{D}}[A_L(x) = L(x)] \geq 1 - n^{-\omega(1)}$.

Lastly, we present an alternative proof for the result of Doron et al. using a **proof paradigm that is both considerably simpler and more general**; in fact, we show how to simplify the analysis of *any construction* that "extracts randomness from a pseudoentropic string". We use this simpler proof to extend their result, deducing a mildly slower derandomization (i.e., with cubic or quadratic overhead) from weaker hardness assumptions (i.e., for SVN circuits that do not use randomness).

[*]Massachusetts Institute of Technology, Cambridge MA. Part of the work was done while L.C. was visiting the Weizmann Institute of Science. Email: `lijieche@mit.edu`

[†]Massachusetts Institute of Technology, Cambridge MA. Part of the work was done while R.T. was a student at the Weizmann Institute of Science. Email: `roeitell@gmail.com`

ISSN 1433-8092

Contents

1 Introduction

Can we replace all randomized algorithms for decision problems by deterministic algorithms with roughly similar runtime? The long line of works typically referred to as "hardness-to-randomness", which was initiated by [Yao82; BM84; NW94], gives one way of answering the foregoing question: These works show that certain *lower bounds* for non-uniform circuits imply derandomization with bounded *runtime overhead*.

The fastest conditional derandomization in the classical line-of-works was proved by Impagliazzo and Wigderson [IW99], who showed that if $\mathcal{E} \not\subset \mathcal{SIZE}[2^{.01\cdot n}]$, then $pr\mathcal{BPP} = pr\mathcal{P}$. By a padding argument, this conclusion implies that randomized time-T algorithms can be simulated in deterministic time $T(n)^c$ for some large constant $c \in \mathbb{N}$. In other words, they showed that when solving decision problems, randomness can be deterministically simulated with a *polynomial runtime overhead*.

In a recent exciting work, Doron, Moshkovitz, Oh, and Zuckerman [DMO+20] asked if an *even faster* derandomization is possible: Could we prove that derandomization with a *small* polynomial overhead (i.e., derandomization in time $T(n)^c$ for a small value of c) follows from plausible hypotheses? Taken to the extreme, could it be that randomized algorithms for decision problems can be deterministically simulated with *almost no runtime overhead*? Their main result is that derandomization with only a *quadratic* time overhead (i.e., $c \approx 2$) is possible under a certain lower bound hypothesis; specifically, this conclusion follows from the hypothesis that $\mathcal{DTIME}[2^n]$ is hard for *randomized and non-deterministic circuits* of very large size $\approx 2^{.99\cdot n}$. That is:

Theorem 1.1 (derandomization with quadratic overhead [DMO+20])**.** *For every constant $\epsilon > 0$ there exists a constant $\delta > 0$ such that the following holds. Assume that there exists $L \in \mathcal{DTIME}[2^n]$ that cannot be computed by randomized SVN circuits*[1] *of size $2^{(1-\delta)\cdot n}$, even infinitely-often. Then, for every time-constructible $T\colon \mathbb{N} \to \mathbb{N}$ such that $T(n) \geq n$ we have that $pr\mathcal{BPTIME}[T(n)] \subseteq pr\mathcal{DTIME}[T(n)^{2+\epsilon}]$.*

The result of Doron *et al.* [DMO+20] provides evidence that extremely fast derandomization might be possible, and moreover opens the door to asking if an even *faster* derandomization, namely with overhead $\approx T(n)^{1.01}$, might also be possible. However, the hypothesis in Theorem 1.1 is considerably stronger than the hypotheses in the classical works: It refers to circuits that are not only larger (i.e., of size $2^{.99\cdot n}$ rather than $2^{.01\cdot n}$), but that also use randomness and non-determinism; that is, it refers to a lower bound for the non-uniform analogue of $\mathcal{MA} \cap co\mathcal{MA}$. Needless to say, these additional resources might significantly increase the power of non-uniform circuits.[2] Moreover, in contrast to classical "hardness-to-randomness" works, the hypothesis in Theorem 1.1 is not known to be *necessary* for the existence of the corresponding PRG that they construct.

1.1 Our contributions: Bird's eye

The current paper extends the line of inquiry opened by [DMO+20], focusing on the possibility of extremely fast derandomization, and provides answers to several open questions that arose from their work. We now describe our results informally and in high-level, and later on we will elaborate in more detail.

[1]Randomized SVN circuits are the non-uniform analogue of $\mathcal{MA} \cap co\mathcal{MA}$; see Definition 3.4 for details.

[2]For a recent demonstration of the power of $\mathcal{MA}$ algorithms that run in exponential time, see [Wil16]; we refer the reader to the corresponding discussion in [DMO+20, Section 1.6].

1. Simulating randomness almost "for free". The conditional derandomization in Theorem 1.1 works in *quadratic* time $T(n)^{2+\epsilon}$, and the main open question following this result is whether or not one we can derandomize probabilistic algorithms in *near-linear* time $T(n)^{1+\epsilon}$ (i.e., with almost no time overhead).

We provide an affirmative answer to the foregoing question, conditioned on a plausible hypothesis, which is formally incomparable to the one in Theorem 1.1 but is arguably more standard. Specifically, we show that probabilistic algorithms running in time $T(n)$ can be deterministically simulated in time $n \cdot T(n)^{1+\epsilon}$, conditioned on the following: There exist one-way functions secure against polynomial-sized circuits, and there exists a problem in time $2^{k \cdot n}$ that is hard for algorithms that run in time $2^{(k-.01)\cdot n}$ and use $2^{.99n}$ bits of non-uniform advice, where k is a sufficiently large constant (see Theorem 1.2, and see the subsequent discussion for a comparison with Theorem 1.1). The second assumption (or, more accurately, a relaxation of it that we use) is essentially *necessary* to obtain the derandomization conclusion using PRGs (see Section 1.2.3), and is a natural extension of hardness hypotheses from classical "hardness-to-randomness" results.

We further improve the derandomization time to $n^{1+\epsilon} \cdot T(n)$ for natural special cases, conditioned on hypotheses that are slightly more technically involved and/or a mildly stronger (see Theorems 4.8 and 4.12). This time bound almost matches the straightforward *non-uniform* derandomization, and we prove that it is essentially optimal, under a "counting" version of the non-deterministic strong exponential time hypothesis (i.e., under #NSETH; see Theorem 1.3). We show a faster *average-case* derandomization for polynomial-time algorithms, namely in time $n^{\epsilon} \cdot T(n)$, for a natural class of efficiently-samplable distributions and with success probability $1 - n^{-\omega(1)}$ (see Theorem 1.4).

2. Fast derandomization from weaker hypotheses. Can we prove that derandomization with a small fixed polynomial overhead follows only from hypotheses that generalize the ones in classical "hardness-to-randomness" results (i.e., without referring to non-uniform analogues of $\mathcal{MA} \cap co\mathcal{MA}$ and without any cryptographic assumptions)? Such a hypothesis would not only be formally weaker, but – as mentioned above – is also necessary for any derandomization that uses a PRG.

A natural first step towards this goal would be to deduce fast derandomization from hardness for SVN circuits that do not use randomness, which are non-uniform analogues of $\mathcal{NP} \cap co\mathcal{NP}$. We show that derandomization with either *cubic* or *quartic* overhead is possible, under hardness assumptions that are similar to the one in Theorem 1.1 but refer only to SVN circuits that *do not use randomness* (see Theorem 1.7). In addition, we show a near-optimal *quantified derandomization*[3] under a stronger hypothesis, which still refers only to SVN circuits that do not use randomness (see Theorem 5.4).

3. Fast derandomization via simple proof paradigms. The proof of Theorem 1.1 in [DMO+20] is highly non-trivial, relying on refined technical notions and on complicated analyses (see Section 2.2.2 for details). Can we rely on simpler "hardness-to-randomness" proofs to obtain derandomization with very small overhead, in order to better understand the core observations that allow for such derandomization?

[3]The problem of quantified derandomization, which was introduced by Goldreich and Wigderson [GW14], calls for derandomizing algorithms that err *extremely rarely* (instead of with probability at most 1/3, as in standard derandomization).

All of our results rely on *simple and intuitive proofs* that use only standard technical tools. In particular, our results rely on proof strategies that are significantly different than the ones in [DMO+20]. In fact, one of our contributions is a considerably simpler proof for Theorem 1.1, which relies on an observation of independent interest: Any PRG construction that is analyzed as "extracting randomness from a pseudoentropic string" (in particular, the construction of [DMO+20]) can be analyzed via a proof strategy that is both *simpler* and *more general*; loosely speaking, the alternative proof strategy relies on "error-reduction then quantified derandomization". For more details see Section 1.3.

1.2 Derandomization with almost no slowdown

Our first result is that under a plausible hardness hypothesis, probabilistic algorithms that run in time $T(n)$ can be deterministically simulated in time $n \cdot T(n)^{1+\epsilon}$, for an arbitrarily small constant $\epsilon > 0$ and for all time bounds $T(n)$.

Similarly to other results in the "hardness-to-randomness" line-of-works, our derandomization relies on a PRG construction. However, the standard approach of constructing a PRG that "fools" non-uniform circuits of size $T(n)$ cannot work here, because such a PRG requires a seed of length at least $\log(T(n))$ (and evaluating a time-T algorithm at each output of the PRG requires time $T(n)^2$). Our way to bypass this obstacle is to observe that the standard approach is an "overkill": When transforming a probabilistic algorithm into a distinguisher for the PRG, the distinguisher runs in time $T(n)$ but *only uses n bits of non-uniform advice* (i.e., the advice corresponds to the input, which is of length n rather than $T(n)$). Thus, it suffices to "fool" the foregoing class of distinguishers (see Proposition 4.7), and indeed there exists a non-explicit PRG for this class with *sub-logarithmic* seed length; that is, the distinguisher class can be modeled by $\mathcal{DTIME}[N]/T^{-1}(N)$ (where the notation alludes to $N = T(n)$), and it can be "fooled" by a (non-explicit) PRG with seed length $(1 + o(1)) \cdot \log(T^{-1}(N))$. [4]

In our first result we construct a PRG that yields derandomization in time $n \cdot T(n)^{1+\epsilon}$ (rather than $n^{1+\epsilon} \cdot T(n)$), conditioned on the following. First, we assume that there exist non-uniformly secure one-way functions; recall that this assumption is not known to imply (by itself) derandomization in less than sub-exponential time. Secondly, we assume that there exists a problem decidable in time $2^{k \cdot n}$ that cannot be solved in time $2^{(k-\delta)\cdot n}$ with $2^{(1-\delta)\cdot n}$ bits of non-uniform advice, where δ is sufficiently small and k is sufficiently large; indeed, this assumption extends hardness assumptions from classical "hardness-to-randomness" results in a natural way (cf., e.g., [IW99]). In more detail:

Theorem 1.2 (derandomization with almost no slowdown for all probabilistic algorithms)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that the following holds. Let $T\colon \mathbb{N} \to \mathbb{N}$ be any time-constructible non-decreasing function, and let $k = k_{\epsilon,T} \geq 1$ be a sufficiently large constant.*[5] *Assume that there exist one-way functions that are secure against polynomial-sized circuits, and that there exists $L \in \mathcal{DTIME}[2^{k\cdot n}]$ such that $L \notin \mathtt{i.o.}\mathcal{DTIME}[2^{(k-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$. Then, we have that $pr\mathcal{BPTIME}[T(n)] \subseteq pr\mathcal{DTIME}[n \cdot T(n)^{1+\epsilon}]$.*

[4]To see this, for every input length N we "fool" the first $\epsilon(N)$ machines that run in time N, instantiated with every possible advice, where ϵ is any super-constant function. This yields at most $\epsilon(N) \cdot 2^{T^{-1}(N)}$ distinguishers, and therefore a seed of length $\log(T^{-1}(N)) + \log(\epsilon(N))$ suffices.

[5]When T is a polynomial the constant k will be very close to the polynomial power of T, and when T is super-polynomial the constant k will be linear in $1/\epsilon$; see Theorem 4.10.

We also further relax the non-cryptographic hypothesis in Theorem 1.2, and only require that the *amortized* time-complexity of L when printing its entire truth-table will be $2^{k\cdot n}$ (rather than requiring each entry in the truth-table to be computable in such time; see Theorem 4.10). The reason that we mention this relaxation is that *the existence of such an L is necessary for the PRG conclusion* (see Proposition 1.6 and Theorem 4.11). This relaxation will apply to all results in the current section, but for simplicity we will avoid mentioning it explicitly, and defer the full result statements to the technical section.

Thus, one of our assumptions in Theorem 1.2 is *necessary* for the conclusion, and the other is a standard and fundamental one. Moreover, both assumptions refer only to standard computational models that *do not use non-determinism or randomness* (rather than to non-uniform analogues of $\mathcal{MA} \cap co\mathcal{MA}$). Thus, we argue that our hypothesis is more standard and appealing than the one in Theorem 1.1. Intuitively, the main part in our hypothesis that is stronger (and allows for faster derandomization) is that our hardness assumption refers to a separation of uniform algorithms from non-uniform procedures in a *"higher" time bound* (i.e., in time $2^{k\cdot n}$ rather than in time 2^n).

The benefit in derandomization in near-linear time (compared to, say, derandomization in quadratic time) is particularly salient when considering randomized algorithms that run in time close to that of a "brute-force" algorithm. Many such "better-than-brute-force" randomized algorithms are known, for example for $\mathcal{NP}$-complete graph problems (see, e.g., [Bjö14; CKN18]), for satisfiability of formulas and circuits (see, e.g., [PPS+05; CSS16]), and for $\mathcal{NP}$-complete algebraic problems (see, e.g., [LPT+17]). For all these problems, derandomization in time $T(n)^{1+\epsilon}$ would yield a better-than-brute-force deterministic algorithm, but derandomization in time $T(n)^2$ is trivial.

1.2.1 Further minimizing the derandomization overhead

The derandomization time overhead of $T(n) \mapsto n \cdot T(n)^{1+\epsilon}$ is small, but is it as small as it can be? We prove several additional results that address this question, both by improving the upper-bound and by showing a near-matching conditional lower bound.

Recall that the known *non-uniform* derandomization of $pr\mathcal{BPTIME}[T]$ yields circuits of size $O(n \cdot T(n))$.[6] We first show that such an overhead is unavoidable, at least for uniform algorithms,[7] conditioned on a *counting* version of the Non-Deterministic Strong Exponential-Time Hypothesis (NSETH), which is weaker than NSETH itself. Specifically, extending a result of Williams [Wil16], we prove that derandomization in time $O(n \cdot T(n))$ is optimal if for every $\epsilon > 0$ there does not exist a non-deterministic machine that counts the number of satisfying assignments of a k-SAT formula over n bits in time $2^{(1-\epsilon)\cdot n}$, assuming that $k = k_\epsilon$ is sufficiently large (see Section 6).

Secondly, for a large class of probabilistic algorithms, we improve the derandomization overhead in Theorem 1.2 to $n^{1+\epsilon} \cdot T(n)$, which almost matches this lower bound. For probabilistic polynomial-time algorithms this improvement can be proved under hypotheses that are similar to the one in Theorem 1.2, yet are slightly more technically involved to state (see Theorem 4.8). For probabilistic algorithms that run in time

[6]This is since for every probabilistic algorithm and $n \in \mathbb{N}$, by a Chernoff bound there exist $O(n)$ fixed random strings that lead the algorithm to a correct decision on all inputs.

[7]Note that many problems can be solved in constant probabilistic time $T(n) = O(1)$ but require linear deterministic time $O(n)$ (e.g., estimating the Hamming weight of the input). Nevertheless, the lower bound for this specific time bound does not rule out an *additive* derandomization overhead of n.

$T(n) = 2^{n^{o(1)}}$ this improvement follows under hypotheses that are similar to the ones required for polynomial-time algorithms, while strengthening the cryptographic assumption to assume that the one-way function is secure against circuits of *subexponential size*.

For convenience, let us state the three foregoing results (i.e., the conditional upper-bounds and lower-bound) together. This yields a suboptimal result that is nevertheless relatively clean and easy to state (the suboptimality is mainly since we will assume a subexponentially-secure one-way function even to derandomize polynomial-time algorithms; see Theorems 4.8, 4.12, and 6.4 for the full technical statements.)

Theorem 1.3 (derandomization with near-optimal overhead; informal)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that for every "nice" function $T\colon \mathbb{N} \to \mathbb{N}$ satisfying $T(n) = 2^{n^{o(1)}}$ the following holds. Assume that for some $\gamma > 0$ there exist one-way functions that are secure against circuits of size 2^{n^γ}, and that there exists $L \in \mathcal{DTIME}[2^{\delta \cdot n} \cdot T'(n)]$ such that $L \notin \mathtt{i.o.}\mathcal{DTIME}[T']/2^{(1-\delta)\cdot n}$, where $T'(n) = 2^{O(\delta\cdot n)} \cdot T(2^{(1-\delta)\cdot n})$. Then, $pr\mathcal{BPTIME}[T] \subseteq pr\mathcal{DTIME}[n^{1+\epsilon} \cdot T(n)]$. Moreover, under the hypothesis* #NSETH *it holds that $\mathcal{BPTIME}[T] \not\subseteq \mathcal{DTIME}[n^{1-\epsilon} \cdot T(n)]$, for every polynomial T and $\epsilon > 0$.*

1.2.2 Faster average-case derandomization

Bypassing the conditional lower bound in Theorem 1.3, we show that a faster derandomization is possible if we are willing to settle for derandomization that *succeeds only on most inputs*, rather than in worst-case. Recall that for any $L \in \mathcal{BPTIME}[T]$ and any distribution $\mathcal{D}$ over $\{0,1\}^n$ there exists a circuit family of size $O(T(n))$ that correctly decides L_n with high probability over choice of input from $\mathcal{D}$ (say, with probability 0.99). Moreover, when we restrict the class of distributions only to those that are samplable in time $T(n)$, then there exists a *single* circuit family of size $\tilde{O}(T(n))$ that correctly decides L with high probability with respect to *all* distributions in this class.[8] (Note that this class of distributions is quite natural, and in particular contains the uniform distribution.)

We show a near-matching explicit (i.e., uniform) derandomization under a hypothesis that is similar to the one in Theorem 1.2. Specifically, under this hypothesis, for every $L \in \mathcal{BPTIME}[T]$ we construct a deterministic algorithm A_L that runs in time $n^\epsilon \cdot T(n)$ and succeeds with high probability with respect to every T-time samplable distribution. (In particular, with respect to uniform distribution.) We will focus on the setting of polynomial time bounds $T(n) = n^k$, since this is the more interesting setting for improving the worst-case bound of $n^{1+\epsilon} \cdot T(n)$ to the average-case bound $n^\epsilon \cdot T(n)$.

Theorem 1.4 (average-case derandomization with overhead n^ϵ; see Theorem 4.13)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that for every $T(n) = n^k$ the following holds. Assume that there exist one-way functions secure against polynomial-sized circuits, and that there exists $L_0 \in \mathcal{DTIME}[2^{\delta\cdot n} \cdot T'(n)]$ such that $L_0 \notin \mathtt{i.o.}\mathcal{DTIME}[T']/2^{(1-\delta)\cdot n+1}$, where $T'(n) = 2^{(1-\delta)\cdot(2k/\epsilon)\cdot n + O(\delta\cdot n)}$. Then, for every $L \in \mathcal{BPTIME}[T]$ there exists a deterministic algorithm A_L that runs in time $n^\epsilon \cdot T(n)$ such that for every T-time samplable distribution $\mathcal{D}$ it holds that $\Pr_{x\sim\mathcal{D}_n}[A_L(x) = L(x)] = 1 - n^{-\omega(1)}$.*

[8] To see this, for every $n \in \mathbb{N}$, consider the first n Turing machines that run in time T. By a Chernoff bound, there exist $O(\log(n))$ random strings that lead the probabilistic algorithm to be correct with probability 0.99 on each of the n distributions sampled by the n Turing machines. The average-case error (of 0.99) can be decreased by first applying error-reduction to the original probabilistic machine.

An appealing interpretation for the derandomization in Theorem 1.4 is that there exists a deterministic algorithm A_L such that every T-time algorithm that tries to find an input x such that $A_L(x) \neq L(x)$ succeeds only with negligible probability.

1.2.3 Batch-computable PRGs and problems with bounded *amortized* complexity

Recall that the results in this section are obtained by constructing PRGs for the class $\mathcal{DTIME}[O(N)]/T^{-1}(N)$, which (ideally) have seed length $\ell \approx \log(T^{-1}(N)) = \log(n)$. Also recall that our goal is to obtain derandomization in time $n^{1+\epsilon} \cdot N$.

Unlike classical results, we do not prove that our PRGs are computable in time close to N on each of the 2^ℓ seeds, but rather only *"batch-computable"* in time close to $2^\ell \cdot N$ on *all seeds* at once. Indeed, such PRGs still suffice for the standard derandomization approach of enumerating over all seeds. And moreover, as mentioned after Theorem 1.2, the "batch-computable" PRGs that we construct also follow from the relaxed hypothesis that asserts an upper-bound on the *amortized* time-complexity of the hard problem when computing its entire truth-table (rather than a worst-case time bound); and the existence of such a problem is in fact necessary to get "batch-computable" PRGs.

The point is that the relationship that we show between the latter two objects is *quantitatively tighter* than the known relationship between standard PRGs (that are efficiently-computable on each seed) and hard problems (with bounded worst-case complexity). Thus, in the context of derandomization with almost no time overhead, it turns out to be more fruitful to study the two weaker objects. To be more explicit about this point, let us state a special case of the connection between these two objects. (The result follows from the technical versions of results that were already stated or mentioned above, but its parametrization is less clean since we wish to highlight the parametric tightness.)

Definition 1.5. *For $f\colon \{0,1\}^* \to \{0,1\}$, we say that $f \in \texttt{amort-}\mathcal{DTIME}[T]$ if for every $n \in \mathbb{N}$, the truth-table of f on n-bit inputs can be printed in time $2^n \cdot T(n)$.*

Proposition 1.6 (near-equivalence between batch-computable PRGs and problems with bounded amortized time complexity, the polynomial setting)**.** *There exists a universal constant $c > 1$ such that the following holds. Assume there exists one-way functions that are secure against polynomial-sized circuits. Then, for every $\epsilon > 0$ there exist $\delta, \delta' > 0$ such that for any fixed constant $k \geq 1$:*

$$\exists L \in \texttt{amort-}\mathcal{DTIME}[2^{\delta \cdot n} \cdot T_k(n)] \setminus \texttt{i.o.}\mathcal{DTIME}[T_k(n)]/2^{(1-\delta)\cdot n}\,,$$

where $T_k(n) = 2^{(1-\delta)\cdot kn + c\delta \cdot n}$

$$\Downarrow \qquad \text{(Theorem 4.8)}$$

$\exists$ $(1/8)$-*PRG for* $\mathcal{DTIME}[O(n)]/n^{1/k}$ *with seed length* $(1+\epsilon)\cdot(1/k)\cdot\log(n)$

that is batch-computable on all seeds in time $n^{(1+2\epsilon)/k} \cdot n$

$$\Downarrow \qquad \text{(Theorem 4.11)}$$

$$\exists L \in \texttt{amort-}\mathcal{DTIME}[2^{\delta' \cdot n} \cdot T_k(n)] \setminus \texttt{i.o.}\mathcal{DTIME}[T_k(n)]/2^{(1-\delta')\cdot n}\,,$$

where $T_k(n) = 2^{(1-\delta')\cdot kn}$

1.3 Fast derandomization via a simple paradigm

The starting point for our other contributions is an alternative and *considerably simpler* proof for Theorem 1.1. Using a different high-level proof strategy, we rely on the hypothesis to construct a very simple PRG, and analyze it in a way that avoids essentially all of the involved technical work that was carried out in [DMO+20].

Our contributions here are two-fold. First, our simple proof is flexible, and allows us to extend the original result in several directions (see below). And secondly, we show that our high-level proof strategy simplifies a well-known proof strategy *in general* as well as sheds new light on a well-known challenge. In a gist (and jumping ahead), the proof strategy of [DMO+20] (following [HIL+99; BSW03; FSU+13] and others) is to "extract randomness from a pseudoentropic string". We show that any PRG construction that is analyzed using the foregoing strategy can be analyzed in a way that is both *simpler* and *more general*, without any loss in parameters. To do so we follow an old idea of Sipser [Sip88], which was recently highlighted again in [GW14] and in a sequence of follow-up works (see, e.g., [Tel17; Tel18; KL18; CT19; CJW20]): Specifically, we show an analysis that first reduces the derandomization problem to the problem of quantified derandomization, albeit via a non-standard reduction, and then solves the latter problem. (For further details and a comparison between the two strategies see Section 2.2.2.)

Our simpler proof of Theorem 1.1 follows the high-level strategy above, while solving the quantified derandomization problem using a new construction of a very simple PRG that "fools" extremely biased distinguishers (for details on the latter see Section 2.2.1). Building on this simpler proof, we show how to mildly relax the hypothesis of Theorem 1.1 while deducing an only mildly slower derandomization. Specifically, we show that the hypothesis in Theorem 1.1 can be relaxed to refer only to SVN circuits, which are non-uniform analogues of $\mathcal{NP} \cap co\mathcal{NP}$, while deducing derandomization with either *cubic* overhead $T(n)^{3+\epsilon}$ or *quartic* overhead $\approx T(n)^{4+\epsilon}$ (rather than quadratic overhead), depending on the specific hardness hypothesis. In more detail:

Theorem 1.7 (fast derandomization from hardness for SVN circuits)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that the following holds.*

1. *Assume that there exists $L \subseteq \{0,1\}^*$ whose entire truth-table on n-bit inputs can be printed in time $2^{(3/2)\cdot n}$, but that cannot be computed (on an input-by-input basis) by SVN circuits of size $2^{(1-\delta)\cdot n}$, even infinitely-often. Then, for every time-constructible $T\colon \mathbb{N} \to \mathbb{N}$ we have that $pr\mathcal{BPTIME}[T(n)] \subseteq pr\mathcal{DTIME}[T(n)^{3+\epsilon}]$.*

2. *Assume that there exists $L \in \mathcal{DTIME}[2^n]$ such that L cannot be computed by SVN circuits of size $2^{(1-\delta)\cdot n}$, even infinitely-often. Then, for every time-constructible $T\colon \mathbb{N} \to \mathbb{N}$ we have that $pr\mathcal{BPTIME}[T(n)] \subseteq pr\mathcal{DTIME}[T(n)^{4+\epsilon}]$.*

Using our new PRG, we also construct a near-optimal algorithm for a natural setting of quantified derandomization, under a plausible hypothesis that refers to hardness for SVN circuits. Loosely speaking, we show that randomized time-T algorithms that err on at most $2^{T(n)^{.99}}$ of their random choices can be deterministically simulated in time $T(n)^{1.01}$, if there exists a function whose truth-table on n-bit inputs can be printed in time $2^{(1.01)\cdot n}$, but that is hard to compute (on an input-by-input basis) for SVN circuits of size $2^{.99\cdot n}$. See Theorem 5.4 for details.

New light on bypassing the hybrid argument. A well-known challenge in proving results such as the ones in Theorems 1.1 and 1.2 is that the analysis of the PRG construction needs to avoid a certain hybrid argument (for a detailed explanation see [BSW03, Sec. 1.2], and also [FSU+13]). Indeed, the proofs in [BSW03; DMO+20] as well as our proofs avoid such an argument. However, we believe that the observations in the current section allow to better understand *how* this barrier was bypassed in all these works, which sheds new light on this challenge. Further details appear in Section 2.2.2.

1.4 Organization

In Section 2 we describe our proofs in high-level. In Section 3 we present preliminary definitions and state some well-known results. In Section 4 we show the PRG constructions that correspond to the results in Section 1.2, and in Section 5 we show the PRG constructions that correspond to the results in Section 1.3. Finally, in Section 6 we show the conditional lower bound for derandomization that was mentioned in Section 1.2.

2 Proof overviews

Throughout the section we will be somewhat informal with respect to the precise values of parameters, and in particular denote by $\epsilon > 0$ an unspecified constant that should be thought of as arbitrarily small. Also, for simplicity we explain how to construct PRGs with constant error; the extensions to PRGs with error $n^{-.01}$ are straightforward.

2.1 Proof of Theorems 1.2, 1.3 and 1.4

Let us first see what is the core difficulty that yields a large polynomial overhead in existing "hardness-to-randomness" proofs. Recall that for a probabilistic algorithm M with running time $T(n)$ and a fixed input $x \in \{0,1\}^n$, we consider a distinguisher D_x that gets as input random coins $r \in \{0,1\}^N$, where $N = T(n)$, runs in time N, and outputs the decision of M at input x with randomness r. If we can construct a PRG that "fools" all potential distinguishers D_x, then we can use this PRG to derandomize M.

Classical "hardness-to-randomness" proofs (following [NW94]) rely on *reconstructive* PRGs. Such a PRG is an oracle machine G that gets a random seed and access to a function f and satisfies the following: Any efficient procedure D that distinguishes the output distribution of G^f from the uniform distribution can be transformed to an efficient procedure C_f that computes f. As a contrapositive, if f is hard for every efficient procedure, then G^f "fools" every potential efficient distinguisher D.

The key bottleneck in the reconstructive proof approach is the overhead in transforming D into C_f. In the state-of-the-art construction of Umans [Uma03] (following [NW94; IW99; STV01; TSZS06; SU05]), the transformation overhead is a large polynomial, say $|C_f| = |D|^c = N^c$. Thus, we must assume that f is hard for circuits of size N^c, which means that the time it takes to compute G^f is larger than $N^c = T(n)^c$. This is where the large derandomization overhead comes from, and this is what we want to avoid.

The first idea: Composing two "low-cost" PRGs. We will construct a PRG with optimal seed length $(1+\epsilon) \cdot \log(N)$ and running time $N^{1+\epsilon}$ by composing two suboptimal yet "low-cost" PRGs; that is, each of the two PRGs falls short of achieving the parameters we need by itself, but is nevertheless computable in time $N^{1+\epsilon}$. The first PRG will have short seed length $(1+\epsilon)\cdot\log(N)$ but also short output length N^{ϵ}; and the second PRG will have a relatively-long seed N^{ϵ} but a sufficient output length N.

The first PRG follows from the observation that the *transformation of hardness into randomness entails very little overhead when the PRG outputs a relatively-short string*. In fact, for this purpose we can even use an instantiation of the classical construction of Nisan and Wigderson [NW94] (with variations a-la [RRV02]): Using a function f whose truth-table is of size $|f| = N^{1+O(\epsilon)}$, a suitable instantiation of the NW PRG outputs $M = N^{\epsilon}$ pseudorandom bits using seed length $(1+O(\epsilon))\cdot\log(N)$, in time $N^{1+O(\epsilon)}$, and with a small reconstruction overhead such that $|C_f| = |f|^{\epsilon}\cdot|D| + |f|^{1-2\epsilon} < |f|^{1-\epsilon}$ (i.e., the reconstruction procedure is an oracle machine that gets access to D, runs in time $|f|^{\epsilon}$, and uses $|f|^{1-2\epsilon}$ bits of advice; see Theorem 4.1).[9] Thus, if f is hard for circuits of size $|f|^{1-\epsilon}$, then the PRG "fools" D.

The observation underlying the second PRG is that *standard cryptographic assumptions yield a very fast PRG with polynomial stretch*. Specifically, assuming the existence of a one-way function that is secure against polynomial-sized circuits, there exists a PRG G^{cry} with stretch $M \mapsto N$ that can be computed in time $N^{1+\epsilon}$ and "fools" any distinguisher of size N (see Section 4.1). The composed PRG will take a seed $w \in \{0,1\}^{(1+O(\epsilon))\cdot\log(N)}$, map it to $NW(w) \in \{0,1\}^M$, and output $G(w) = G^{\mathrm{cry}}(NW(w)) \in \{0,1\}^N$. The composition of G^{cry} and of NW indeed "fools" D, where the crucial point is that applying the "outer" PRG G^{cry} can be thought of as yielding a distinguisher $D' = D \circ G^{\mathrm{cry}}$ of approximately the same size as D (and therefore NW "fools" D' and hence D).[10]

This argument yields derandomization with *quadratic overhead*; that is, with running time $N^{2+O(\epsilon)} = T(n)^{2+O(\epsilon)}$. This is because we will evaluate D, which is of size N, on each of the $N^{1+O(\epsilon)}$ outputs of the PRG. Thus, so far we deduced the conclusion of Theorem 1.1 from incomparable hypotheses (which refer only to standard circuits), but still fall short of proving the stronger conclusion of Theorem 1.2.

The second idea: Using small and extremely hard truth-tables to fool distinguishers with "a little" non-uniformity. The bottleneck in the argument above is that the seed length of the NW PRG is $(1+O(\epsilon))\cdot\log(N)$. In general, the seed length of the NW PRG is proportional to the truth-table size of the hard function, and in our instantiation of the NW PRG the seed length is $(1+\epsilon)\cdot\log(|f|)$. However, it a-priori seems impossible to use a smaller truth-table, since our distinguisher is a non-uniform circuit of size N, and we cannot assume hardness of a function with truth-table size $|f| < N$ for circuits of size N (i.e., it is trivial to compute such a function with N bits of advice).

As mentioned in Section 1, the pivotal observation to solve this problem is that when the distinguisher $D = D_x$ models the execution of a probabilistic algorithm with running-time $N = T(n)$ on an input x, then D can be thought of as a time-N algorithm

[9]The parameters stated here are not fully accurate (e.g., we need $M = N^{\Omega(\epsilon^2)}$ for this to be true), but the difference is immaterial for this high-level discussion. See Section 4 for the full details.

[10]To see this in more detail, observe that for $\ell = (1+O(\epsilon))\cdot\log(N)$, the acceptance probability of D on the distribution $G^{\mathrm{cry}}(NW(\mathbf{u}_\ell))$ equals the acceptance probability of $D' = D\circ G^{\mathrm{cry}}$ on the distribution $NW(\mathbf{u}_\ell)$. Since D' is of approximately the same size as D (i.e., size $N^{1+\epsilon}$ instead of N), we can instantiate the NW PRG with parameters very similar to the ones above, and assuming that f is hard for circuits of size $|f|^{1-\epsilon}$, the distribution $NW(\mathbf{u}_\ell)$ is pseudorandom for D'. Since D' has approximately the same acceptance probability as D, the distribution $NW(\mathbf{u}_\ell)$ is also pseudorandom for D.

that uses only $|x| = n = T^{-1}(N)$ bits of non-uniform advice. In other words, to derandomize time-T algorithms it suffices to "fool" the class $\mathcal{DTIME}[N]/T^{-1}$, rather than all circuits of size N. This opens the door to instantiating the PRG using a *very hard function whose truth-table is of size significantly smaller than the running time of our distinguisher*; in other words, we will be using a function with super-exponential time complexity.

Let us spell out the resulting argument, and for concreteness let us focus on the setting of $T(n) = n^k$ for some constant k. The outer PRG $G^{\texttt{cry}}$ will have stretch $n^\epsilon \mapsto n^k$, and the inner PRG NW will use a function of truth-table size $|f| = n^{1+O(\epsilon)}$ and a seed of length $(1 + O(\epsilon)) \cdot \log(n)$ to output n^ϵ bits. Recall that the reconstruction procedure of NW is an oracle machine that runs in time $|f|^\epsilon$ and uses $|f|^{1-2\epsilon}$ bits of non-uniform advice; thus, a distinguisher that runs in time $T(n)$ and uses n bits of advice yields an algorithm A_f for f that runs in time $T(n) \cdot n^{O(\epsilon)}$ and uses $n + |f|^{1-2\epsilon} < |f|^{1-\epsilon}$ bits of advice (see Theorem 4.1). When we consider the complexity of A_f as a function of the input size $\ell = \log(|f|)$ to f, we get an algorithm that runs in time $T(2^{(1-O(\epsilon))\cdot\ell}) \cdot 2^{O(\epsilon\cdot\ell)}$ and uses $2^{(1-\epsilon)\cdot\ell+1}$ bits of advice. Thus, for this to work we need to assume that $f \notin \texttt{i.o.}\mathcal{DTIME}[T(2^{(1-\epsilon)\cdot\ell}) \cdot 2^{O(\epsilon\cdot\ell)}]/2^{(1-\epsilon)\cdot\ell+1}$. [11]

The last necessary missing piece: Batch-computable PRGs and hard functions with bounded *amortized* time complexity. The resulting derandomization algorithm computes the output-set of the PRG, and evaluates D at each of the strings in this set. Now, recall that we want this algorithm to run in time $n^{1+O(\epsilon)} \cdot T(n)$. It is not clear if we can compute the PRG at each seed in time close to $T(n)$ (as is the standard approach); nevertheless, we can still *compute the entire output-set of the PRG "in a batch" in time* $n^{1+O(\epsilon)} \cdot T(n)$, which suffices for our purposes.

For concreteness, let us still focus on the setting of $T(n) = n^k$. We first compute the entire truth-table of f, and we need to do so in time at most $n^{1+O(\epsilon)} \cdot T(n) = 2^\ell \cdot 2^{O(\epsilon\cdot\ell)} \cdot T(2^{(1-O(\epsilon))\cdot\ell})$. Thus, we have to assume that when computing the entire truth-table of f, the amortized time cost per entry is no more than $2^{O(\epsilon\cdot\ell)} \cdot T(2^{(1-O(\epsilon))\cdot\ell})$. (Note that the constant hidden inside the O-notation in $2^{O(\epsilon\cdot\ell)}$ may be larger than the constant hidden inside the O-notation in $2^{O(\epsilon\cdot\ell)}$ in our lower bound.) As mentioned in Section 1, the existence of such a function is necessary in order to construct a "batch-computable" PRG as the one that we are constructing (see Theorem 4.11). Now, given access to f, we can compute the output-set of the NW PRG in time $n^{1+O(\epsilon)}$ (i.e., we compute the combinatorial designs once in advance, and similarly apply an error-correcting code to the truth-table of f once in advance; see Appendix A). Finally, we evaluate $D' = D \circ G^{\texttt{cry}}$ on each of the $n^{1+O(\epsilon)}$ resulting strings. The PRG $G^{\texttt{cry}}$ can be assumed to run in time $T(n) \cdot n^\epsilon$ (see Section 4.1), and thus our final running-time is indeed $n^{1+O(\epsilon)} \cdot T(n)$.

The above argument proves the special case of Theorem 1.2 for polynomial time functions $T(n) = n^k$. (In fact, it even proves the stronger result for this special case, in which the derandomization time is $n^{1+\epsilon} \cdot T(n)$.) In Section 4 we explain how to extend the argument to super-polynomial time functions $T(n) = n^{\omega(1)}$. In a gist, we will either reduce the super-polynomial case to the polynomial case using a padding argument, which yields derandomization in time $n \cdot T(n)^{1+\epsilon}$; or rely on a stronger cryptographic hypothesis to obtain an outer PRG $G^{\texttt{cry}}$ with super-polynomial stretch, which yields derandomization in time $n^{1+\epsilon} \cdot T(n)$. See further details in Section 4.

[11] To see that this is consistent with the time bound stated in Theorem 1.2, plug-in $T(n) = n^k$ to the time bound to see that $T(2^{(1-\epsilon)\cdot\ell}) \cdot 2^{O(\epsilon\cdot\ell)} = 2^{(1-\epsilon)\cdot k\ell+O(\epsilon\cdot\ell)} = 2^{(1-(1-O(1/k))\cdot\epsilon)\cdot k\ell}$.

Proof of Theorem 1.4: Faster average-case derandomization. For simplicity, let us prove that we can derandomize algorithms that run in time $T(n) = n^k$ in time $n^{\epsilon} \cdot T(n)$ with respect to the *uniform* distribution (rather than any T-time samplable distribution).

Given a probabilistic machine M running in time T, consider the machine M' that gets input $w \in \{0,1\}^m$, maps it to $x = G^{\mathtt{cry}}(w) \in \{0,1\}^n$ where $n = m^{1/\epsilon}$, and outputs $M(x)$. We now instantiate Theorem 1.2 with the time function $T'(m) = m^{k/\epsilon}$, which bounds the running time of M'. Using appropriate hypotheses (to apply Theorem 1.2 with the function T'), we deduce there exists a PRG G with seed length $(1+\epsilon) \cdot \log(m)$ that is batch-computable in time $m^{1+\epsilon} \cdot m^{k/\epsilon}$ and can be used to replace the random coins of M'. Now, a key point to note is that M' only uses its random coins as random coins for M. Hence, we deduce that *the PRG G can be used to replace the random coins of M on any input $x \in \{0,1\}^n$ in the output-set of $G^{\mathtt{cry}}$*. Note that as a function of $|x| = n$, the PRG has seed length $(1+\epsilon) \cdot \log(n^{\epsilon}) < 2\epsilon \cdot \log(n)$ and running time at most $n^{\epsilon(1+\epsilon)} \cdot n^k < n^{k+2\epsilon}$.

Of course, this still doesn't mean that the G can be used to replace the random coins of M on inputs $x \in \{0,1\}^n$ that are not in the output-set of $G^{\mathtt{cry}}$. The crucial observation is that we can now *reuse the pseudorandomness properties of $G^{\mathtt{cry}}$ to "fool" a second test* (the first use of the pseudorandomness of $G^{\mathtt{cry}}$ was in our instantiation of Theorem 1.2). Specifically, recall that $G^{\mathtt{cry}}$ is a cryptographic PRG that was obtained relying on the existence of one-way functions, and hence it can be shown to "fool" circuits of arbitrary polynomial size (see Proposition 4.3). Now, consider an algorithm T that gets input $x \in \{0,1\}^n$ and checks whether or not the pseudorandom coins produced by G can be used to replace the random coins of M at x.[12] Since T can be implemented by a circuit of size $\mathrm{poly}(n^k)$, it is "fooled" by $G^{\mathtt{cry}}$. And since for all x in the output-set of $G^{\mathtt{cry}}$ it holds that $T(x) = 1$, it follows that for almost all $x \in \{0,1\}^n$ it holds that $T(x) = 1$. Hence, for almost all $x \in \{0,1\}^n$ we can use G to replace the random coins of M at x.

The foregoing argument extends naturally to any T-time samplable distribution. For any T-time sampling algorithm S, we use $G^{\mathtt{cry}}$ with stretch $m \mapsto T(n)$, and define $M'(x) = M(S(G^{\mathtt{cry}}(x)))$ (this generalizes the foregoing uniform case, which is obtained using $S(z) = z$). The running time of M' is still less than $T'(m) = m^{k/\epsilon}$, and therefore the rest of the argument proceeds without change.

2.2 Proofs of Theorems 1.1 and 1.7

As mentioned in Section 1.3, our alternative proof of Theorem 1.1, which is also the basis of our proof of Theorem 1.7, is based on two key components. First, we replace the high-level proof strategy of [DMO+20] (following [HIL+99; BSW03] and others) of "extracting from a pseudoentropic string" with a simpler and more general strategy that involves reducing the derandomization problem to a problem of quantified derandomization (in a non-standard way) and then solving the latter. Secondly, to instantiate the simpler high-level strategy in our setting, we present a very simple reconstructive PRG for quantified derandomization.

[12] To be more accurate, T solves a promise-problem wherein the "yes" instances are x such that the gap between the acceptance probability of $M(x,\cdot)$ with random coins and the acceptance probability of $M(x,\cdot)$ with pseudorandom coins is less than $1/6$, and "no" instances are those in which the said gap is at least $1/8$. This problem can be solved probabilistically in time $O(n^{k+2\epsilon})$ (by sampling from the seeds of G and from uniform coins for M and comparing the two estimates), and the probabilistic algorithm can then be converted to a deterministic circuit of size $\mathrm{poly}(n^k)$. See the proof of Theorem 4.13 for details.

We first present the construction of the PRG for quantified derandomization in Section 2.2.1, since it is a very simple and self-contained algorithmic component. Then, in Section 2.2.2 we describe our high-level strategy of using this PRG to obtain very fast derandomization, and also explain why this strategy is both simpler and more general than "extracting from a pseudoentropic string".

2.2.1 A reconstructive PRG for quantified derandomization

We first show how to construct a hitting-set generator (HSG) with seed length $\epsilon \cdot \log(N)$ that "hits" any distinguisher $D\colon \{0,1\}^N \to \{0,1\}$ of size N that accepts all but at most $2^{N^{1-\epsilon}}$ of its inputs; later on we will explain how to easily adapt the construction to obtain a PRG that "fools" all distinguishers with such extreme bias (see below). Our HSG and PRG will be reconstructive, and their reconstruction procedure will be very efficient but will use use non-determinism;[13] thus, they should be compared with known reconstructive PRGs (e.g., [NW94; Uma03]), whose reconstruction procedure does not use non-determinism but has a large polynomial overhead.

The most naive construction of a HSG would be an algorithm H that evaluates f at inputs that are indexed by its random seed. Known constructions are, of course, more complicated, first encoding the truth-table of f by some useful encoding, and then using the seed in a clever way to choose bits from this encoding (see, e.g., [Nis91; NW94; RRV02; TSZS06; SU05; Uma03]). In contrast, in the current context *we show that the naive construction suffices:* The algorithm H gets a seed s of length $\epsilon \cdot \log(N)$ and access to a function f over $(1+\epsilon) \cdot \log(N)$ bits, partitions the truth-table of f into N^ϵ parts of length N, and outputs the corresponding part f_s that is indexed by s.

Why does this naive construction work? Assume that a distinguisher D rejects all parts f_s of the truth-table of f, and recall that D rejects at most $2^{N^{1-\epsilon}}$ strings. The intuition is that *we can describe f information-theoretically by $|D| + N^\epsilon \cdot \log(|D^{-1}(0)|) = O(N) = o(|f|)$ bits*, using the description of D and the N^ϵ indices of the parts f_s in the set $D^{-1}(0)$. We now show how to leverage this information-theoretic argument to obtain an *efficient non-deterministic circuit that computes f* (i.e., gets input $x \in \{0,1\}^{(1+\epsilon)\cdot\log(N)}$ and outputs $f(x)$), which would contradict the hypothesized hardness of f.

Since $|D^{-1}(0)| \le 2^{N^{1-\epsilon}}$, there exists a quasilinear-time computable hash function $h\colon \{0,1\}^N \to \{0,1\}^{N^{1-\epsilon/2}}$ that maps every distinct $y, y' \in D^{-1}(0)$ to different images.[14] Fixing such a function h, we construct a circuit C_f that has "hard-coded" the N^ϵ values $\{z_s = h(f_s)\}_s$. Given an input x, the circuit C_f *non-deterministically constructs* the relevant part f_s of f that contains the location indexed by x (see below) and outputs the corresponding bit of f_s. Specifically, denoting by s the seed such that f_s contains the location indexed by x, the circuit C_f non-deterministically guesses a string f'_s, verifies that $h(f'_s)$ equals the hard-coded value z_s and that $D(f'_s) = 0$, and if these two conditions hold then it outputs the relevant location in f'_s; otherwise, it outputs $\perp$. (Indeed, this non-deterministic circuit never outputs a wrong answer.) This circuit is of size

[13]Thus, when instantiating this PRG with a function f that is hard for non-deterministic circuits, we deduce that the PRG "fools" all potential distinguishers D. We also comment that the reconstruction procedure actually yields an SVN circuit (rather than an arbtirary non-deterministic circuit), but for simplicity we ignore this fact in the high-level overview.

[14]To be more accurate, the function h will be a standard quasilinear-time computable pairwise-independent hash function (see Theorem 3.12), and will satisfy a weaker property that suffices for our purposes: For every $s \in [N^\epsilon]$, there does not exist $y \in D^{-1}(0) \setminus \{f_s\}$ such that $h(y) = h(f_s)$.

$|C_f| = O(N^{1+\epsilon/2})$, and correctly computes f (since the only string $f'_s \in \{0,1\}^N$ such that $D(f'_s) = 0$ and $h(f'_s) = z_s$ is $f'_s = f_s$). Thus, if f (whose truth-table is of size $N^{1+\epsilon}$) is hard for such circuits, then the HSG "hits" D.

Extending the HSG to a PRG. Let us now show how to construct a PRG for all distinguishers D that that evaluate to some $\sigma \in \{0,1\}$ on all but $2^{N^{1-\epsilon}}$ of their inputs. Note that the required pseudorandomness property is that any such D will evaluate to σ on almost all of the outputs of the PRG; thus, if D violates this property (i.e., D is indeed a distinguisher), then we know that D evaluates to $\neg\sigma$ on a noticeable fraction (say, .01) of the output-set of the PRG. This is a weaker property than in the HSG setting, in which we could assume that D evaluates to 0 on *all* of the pseudorandom strings.

Recall that in our reconstruction algorithm for the HSG, after guessing f'_s we check that $D(f'_s) = \neg\sigma$, and otherwise output $\perp$. Applying the same approach to the current setting, we will only be able to compute f on a .01-fraction of the inputs, rather than on all inputs; that is, we reconstruct a "corrupted" version of f, denoted $\tilde{f}$. To solve this problem we simply add an initial step of encoding f by an arbitrary locally list-decodable error-correcting code that is computable in near-linear time (e.g., the Reed-Muller code, as in [STV01]; see Theorem 3.19). Then, our actual reconstruction algorithm runs the local list-decoding procedure of this code, while answering its queries using the original reconstruction algorithm to simulate access to the "corrupted" version $\tilde{f}$.

The construction above already suffices to prove our conditional optimal quantified derandomization, which was mentioned in the end of Section 1.3. For further details and a high-level explanation, see Section 5.

2.2.2 Replacing the "extract from a pseudoentropic string" strategy with the "error-reduction and quantified derandomization" strategy

The proof of Theorem 1.1 in [DMO+20] follows a well-known strategy for constructing PRGs, which dates back to [HIL+99; BSW03]. Specifically, to construct a PRG they first construct a *pseudoentropy generator* (PEG), which takes as input a small random seed and outputs a string that appears to any efficient distinguisher as though it came from a distribution with high min-entropy; and then apply a *seeded extractor* to this pseudoentropic string. An extractor converts distributions with high *statistical* min-entropy to distributions that are *statistically* close to uniform, and the main idea is that we expect that a complexity-theoretic analogue of this statement will also hold: When the input distribution to the extractor *looks* to any efficient distinguisher as though it has high min-entropy, we intuitively expect its output distribution to *look* to any efficient distinguisher as close to uniform. This yields a PRG of the form

$$G(s_0, s_1) = \mathtt{Ext}\left(G_0(s_0), s_1\right) , \tag{2.1}$$

where G_0 is a pseudoentropy generator and Ext is an extractor.

The main challenge in materializing this approach is that constructing PEGs for standard notions of pseudoentropy (e.g., HILL pseudoentropy [HIL+99]) is challenging, and few constructions are known (see, e.g., [HIL+99; BSW03; STV01]). Doron *et al.* [DMO+20] bypassed this obstacle by constructing a PEG for a weaker notion of pseudoentropy, called *metric pseudoentropy*;[15] however, they then faced the problem of proving that extracting from a string with high *metric* pseudoentropy yields a pseudorandom string, which required a lot of technical work. (In fact, to prove this they needed the PEG to "fool" a stronger and non-standard class of distinguishers; see [DMO+20].)

We show that *any construction as in Eq.* (2.1) *is a special case of a class of constructions that can be analyzed in a different and significantly simpler way*. To see this, consider any potential distinguisher $D\colon \{0,1\}^N \to \{0,1\}$ of size $O(N)$. We show that G from Eq. (2.1) is ϵ-pseudorandom for D, as follows:

1. **Non-standard reduction to quantified derandomization:** Let $\bar{D}$ be a circuit that accepts its input z iff $\Pr_r[\mathtt{Ext}(z,r) \in D^{-1}(1)] \in \mu \pm \epsilon/2$, where $\mu = \Pr_r[D(r) = 1]$. Note that $\bar{D}$ is *not* the circuit that is obtained by applying standard (extractor-based) error-reduction to D, but rather a circuit that tests whether or not its input causes the extractor to sample the event $D^{-1}(1)$ correctly, up to error $\epsilon/2$; to decide the latter predicate the circuit $\bar{D}$ has the value μ hard-wired, but this does not cause a problem since $\bar{D}$ is only part of the analysis.

   The two crucial points are that $\bar{D}$ accepts all but a tiny number of exceptional inputs (since `Ext` samples any event correctly, with extremely high probability); and that any $(\epsilon/2)$-PRG G_0 for $\bar{D}$ yields an ϵ-PRG $G(s,r) = \mathtt{Ext}(G_0(s),r)$ for D. [16]

2. **Solving the quantified derandomization problem:** Thus, the only missing part in the proof is to construct a generator G_0 that is $(\epsilon/2)$-pseudorandom for the extremely-biased circuit $\bar{D}$ (i.e., such that $\Pr_s[\bar{D}(G(s)) = 1] \geq 1 - \epsilon/2$). This is indeed a quantified derandomization problem, where the number of exceptional inputs (of $\bar{D}$) is dictated by the parameters of the extractor `Ext`.

   Now, it follows from [BSW03] that metric PEGs are *equivalent* to PRGs that solve the quantified derandomization problem (i.e., a metric PEG for min-entropy k is equivalent to a PRG for distinguishers with at most 2^k exceptional inputs, where in both cases this parameter corresponds to the min-entropy of `Ext`; see Proposition 3.11). Thus, any PEG G_0 is also a PRG for quantified derandomization with the parameters induced by `Ext`, and therefore G is indeed an ϵ-PRG.

The above shows that any construction that relies on "extracting from a pseudoentropic string" – i.e., constructs a PEG and analyzes its composition with an extractor – can also be analyzed via the simpler "error-reduction then quantified derandomization" strategy.[17] However, the converse direction is not known to be true: The proof that metric PEGs can be composed with extractors to yield standard PRGs requires the metric PEG to "fool" a stronger class of distinguishers (see [DMO+20, Sec. 6]).

Applying the proof strategy in our setting. The alternative proof of Theorem 1.1 and our proof of Theorem 1.7 follow by applying the strategy above with the PRG construc-

[15]This means that for every potential distinguisher $D\colon \{0,1\}^N \to \{0,1\}$ there *exists* some distribution $\mathbf{w}$ over $\{0,1\}^N$ with high entropy such that D does not distinguish between $\mathbf{w}$ and the output distribution of G_0^f (for a precise definition see [DMO+20, Sec. 2]).

[16]To see this, call a string z good if $\Pr_r\left[\mathsf{Samp}(z,r) \in D^{-1}(1)\right] \in \mu \pm \epsilon/2$. Then, for any $\sigma \in \{0,1\}$ it holds that $\Pr_{s,r}[\mathsf{Samp}(G(s),r) \in D^{-1}(\sigma)] \leq \Pr[G(s) \text{ is not good}] + \mu + \epsilon/2 < \epsilon$.

[17]This holds for any construction that uses the weak notion of *metric* PEG, and therefore also for constructions that use stronger notions.

tion that was described in Section 2.2.1. The main difference between the two proofs boils down to the construction of a circuit for the function $\bar{D}$. In more detail, note that $\bar{D}$ is a more complicated function than D, and that we need to solve the quantified derandomization problem for $\bar{D}$ rather than for D; intuitively, the overhead in implementing $\bar{D}$ as a circuit yields an overhead in our derandomization time.

The straightforward way of implmenting $\bar{D}$ yields derandomization either in cubic time or in quartic time, depending on the specific hardness hypothesis (see Theorem 5.5). An alternative way is to implement $\bar{D}$ using *randomness*, which mitigates the overhead and allows to obtain derandomization in quadratic time, at the cost of having to assume that the underlying hard function is hard for *randomized SVN circuits* (this yields the alternative proof of Theorem 1.1; see Theorem 5.6). For further details see Section 5.

New light on bypassing the hybrid argument. As mentioned in Section 1.3, it is well-known that proofs of results such as Theorems 1.1 and 1.2 need to avoid a certain hybrid argument (see, e.g., [BSW03; FSU+13]). In [BSW03, Sec 1.2] it was suggested that the proof strategy of "extracting from a pseudoentropic string" allows to bypass this barrier, since the analysis of a reconstruction procedure for a PEG might be easier than the analysis of a reconstruction procedure for a PRG. (To be more accurate, [BSW03] suggested that the connection of pseudoentropy to unpredictability might be closer than the connection of pseudorandomness to unpredictability.)

Our main point is that their suggestion can be reframed as suggesting that *the analysis of PRGs for quantified derandomization might allow avoiding a hybrid argument* more easily than the analysis of PRGs for standard derandomization.[18] This suggestion is at least as plausible as their original suggestion, since any proof that avoids a hybrid argument using the PEG-based approach also yields a proof that avoids a hybrid argument using the quantified-derandomization-based approach. Moreover, all the reconstruction procedures whose analyses avoid a hybrid argument in [BSW03; DMO+20] and in our work can be viewed as solving a corresponding quantified derandomization problem.[19]

We also point out the fact that in all three works, the reconstruction procedures that avoid a hybrid argument used "strong" resources (i.e., either used non-determinism or referred only to space-bounded computation, disregarding time). Thus, it is useful to recall that an additional potential explanation for the success so far might simply be that the analysis of such "strong" reconstruction procedures is easier.

[18]This is consistent with the fact that the unconditionally-known constructions of PRGs for quantified derandomization indeed avoid a hybrid argument; see, e.g., [GW14; Tel17].

[19]In the current work this fact is explicit, and in [DMO+20] this is because the reconstruction procedure is part of a construction of a metric PEG, which is equivalent to a PRG for quantified derandomization. In [BSW03] there is no direct construction of a PEG or of a PRG for quantified derandomization, but the reconstruction procedures are ones that correspond to such construction: This is since the reconstruction algorithms in [BSW03] transform a *very biased distinguisher* (of a pseudoentropic distribution $\mathbf{w}$ over $\{0,1\}^n$ from the uniform distribution $\mathbf{u}_n$) into an algorithm that predicts a bit in the pseudoentropic distribution with high success probability. (Indeed, the point is that the reconstruction algorithm only works for very biased distinguishers; see [BSW03, Section 7] for further details.)

3 Preliminaries

3.1 Complexity classes

We first recall standard definitions of $pr\mathcal{BPTIME}[T]$ and of $\mathcal{DTIME}[T]/s$. Our reason for formally stating these definitions is to clarify that we refer to problems solvable with time and randomness complexity *precisely* $T(n)$ (rather than $O(T(n))$ as in some sources).

Definition 3.1. *For a time-constructible function $T\colon \{0,1\}^{\mathbb{N}} \to \{0,1\}^{\mathbb{N}}$, we say a promise problem $\Pi = (Y, N)$ is in $pr\mathcal{BPTIME}[T]$ if there is a randomized RAM machine M such that for every $x \in \{0,1\}^*$, the machine runs in time $T(|x|)$ and satisfies the following:*

1. *If $x \in Y$ then $\Pr[M(x) = 1] \geq 2/3$.*
2. *If $x \in N$ then $\Pr[M(x) = 0] \geq 2/3$.*

Definition 3.2. *We use $\mathcal{DTIME}[T]/s$ to denote the class of languages that are computable by a deterministic RAM machine that on inputs of length $n \in \mathbb{N}$ runs in time $T(n)$ and uses $s(n)$ bits of non-uniform advice.*

We also define the following notion, which generalizes $\mathcal{NP} \cap co\mathcal{NP}$ to computational models other than Turing machines (e.g., to circuits, oracles machines, etc.). We then extend this notion to a general form of $\mathcal{MA} \cap co\mathcal{MA}$.

Definition 3.3 (non-deterministic unambiguous computation). *We say that a non-deterministic procedure R* computes a function f non-deterministically and unambiguously *if for every $x \in \{0,1\}^*$ the following holds:*

1. *There exist non-deterministic choices for R such that $R(x) = f(x)$.*
2. *For all non-deterministic choices for R it holds that $R(x) \in \{f(x), \perp\}$.*

Definition 3.4 (SVN circuits). *We say that $f\colon \{0,1\}^N \to \{0,1\}$ can be* computed by an SVN circuit of size *S if there exists a non-deterministic circuit D of size S that computes f non-deterministically and unambiguously.*

Definition 3.5 (randomized SVN circuits). *We say that f can be* computed by randomized SVN circuits of size *S if the there exists a randomized non-deterministic circuit D of size S such that for every $x \in \{0,1\}^N$ the following holds:*

1. *There exists w such that $\Pr[D(x, w) = f(x)] \geq 2/3$.*
2. *For every w it holds that $\Pr[D(x, w) \in \{f(x), \perp\}] \geq 2/3$.*

3.2 Pseudorandomness, PRGs and HSGs

We recall the standard definition of pseudorandom generators and of hitting-set generators. For simplicity, when we do not specify the size of a distinguisher, we assume that this size is identical to the number of output bits of the PRG.

Definition 3.6 (distinguisher)**.** *For a distribution* $\mathbf{w}$ *over* $\{0,1\}^n$*, we say that* $D\colon \{0,1\}^n \to \{0,1\}$ distinguishes $\mathbf{w}$ from the uniform distribution with advantage $\epsilon > 0$ *(or that* D *is an* ϵ*-distinguisher for* $\mathbf{w}$*, in short) if* $\left|\Pr_{x\in\{0,1\}^n}[D(x)=1] - \Pr[D(\mathbf{w})=1]\right| \geq \epsilon$*. If* D *is* not *an* ϵ*-distinguisher for* $\mathbf{w}$*, we say that* D *is* ϵ-fooled *by* $\mathbf{w}$*.*

Definition 3.7 (pseudorandom generator)**.** *Let* G *be an algorithm that gets input* 1^n *and a random seed of length* $\ell(n)$ *and outputs an* n*-bit string, and let* $\mathcal{F}$ *be a class of Boolean functions. We say that* G *is a* pseudorandom generator with error μ for $\mathcal{C}$ *(or* μ*-PRG for* $\mathcal{C}$*, in short) if for every* $f \in \mathcal{F}$ *and sufficiently large* $n \in \mathbb{N}$ *it holds that* f *is* μ*-fooled by* $G(1^n, \mathbf{u}_{\ell(n)})$*.*

Definition 3.8 (hitting-set generator)**.** *Let* H *be an algorithm that gets input* 1^n *and a random seed of length* $\ell(n)$ *and outputs an* n*-bit string, and let* $\mathcal{F}$ *be a class of Boolean functions. We say that* H *is a* hitting-set generator for $\mathcal{C}$ with density $\mu > 0$ *(or* μ*-HSG for* $\mathcal{C}$*, in short) if for every* $f \in \mathcal{F}$ *and sufficiently large* $n \in \mathbb{N}$*, either* $\Pr_{x\in\{0,1\}^n}[f(x)=1] < \mu$ *or there exists* $s \in \{0,1\}^{\ell(n)}$ *such that* $f(H(s)) = 1$*.*

For both PRGs and HSGs, when we omit an explicit mention of a class $\mathcal{F}$ (of potential distinguishers) we implicitly refer to the class $\mathcal{F}$ of functions that are computable by circuits of size that is identical to the number of input bits (i.e., of size $S(n) = n$).

We will also refer to "batch-computable" PRGs/HSGs, in which we do not measure the computation time on a single seed, but rather the time that it takes to print the entire output-set of the PRG (on all seeds). That is:

Definition 3.9 (batch-computable PRGs and HSGs)**.** *Let* G *be a* μ*-PRG (resp.,* μ*-HSG) with seed length* ℓ *for some class* $\mathcal{C}$*. We say that* G *is* batch-computable in time T *if for every* $n \in \mathbb{N}$*, the entire set of strings* $\{G(1^n, s) : s \in \ell(n)\}$ *can be printed in time* $T(n)$*.*

We also mention the notion of metric pseudoentropy, which was introduced by Barak, Shaltiel, and Wigderson [BSW03]. The reason for doing so is that we want to explicitly state and prove the claim, which was mentioned in Section 2.2.2, that PRGs for very biased circuits are *equivalent* to metric PEGs. (The proof follows [BSW03], and one direction of it was used in [DMO+20].)

Definition 3.10 (metric pseudoentropy)**.** *We say that a distribution* $\mathbf{w}$ *over* $\{0,1\}^n$ *has* ϵ-metric-pseudoentropy at least k for circuits of size S *if for every circuit* $D\colon \{0,1\}^n \to \{0,1\}$ *of size* S *there exists a distribution* $\mathbf{x}_D$ *over* $\{0,1\}^n$ *with min-entropy at least* k *such that* $\left|\Pr[D(\mathbf{x}_D)=1] - \Pr[D(\mathbf{w})=1]\right| < \epsilon$*.*

Proposition 3.11 (quantified derandomization is equivalent to metric pseudoentropy)**.** *For any distribution* $\mathbf{w}$ *over* $\{0,1\}^n$ *and every* $k \leq n$ *and* $\epsilon > 0$ *the following holds:*

1. *If* $\mathbf{w}$ *has* ϵ*-metric-pseudoentropy at least* k *for circuits of size* S*, then for every* $\delta > 0$ *there does not exist a* $(\delta \cdot 2^k)$*-quantified* $(\epsilon + \delta)$*-distinguisher for* $\mathbf{w}$*.*

2. *If there does not exist a* 2^k*-quantified* ϵ*-distinguisher for* $\mathbf{w}$*, then* $\mathbf{w}$ *has* ϵ*-metric-pseudoentropy at least* k *for circuits of size* S*.*

Proof. Barak, Shaltiel and Wigderson [BSW03] proved that $\mathbf{w}$ has ϵ-metric-pseudoentropy at least k for circuits of size S if and only if for every circuit $D\colon \{0,1\}^n \to \{0,1\}$ of size S and every $\sigma \in \{0,1\}$ it holds that $\Pr[D(\mathbf{w}) = \sigma] \leq \Pr[D(\mathbf{u}_n) = \sigma] \cdot 2^{n-k} + \epsilon$.

Now, assume that $\mathbf{w}$ has ϵ-metric-pseudoentropy at least k for circuits of size S, and let $D\colon \{0,1\}^n \to \{0,1\}$ be a size-S circuit that evaluates to some $\sigma \in \{0,1\}$ on at most $\delta \cdot 2^k$ of its inputs. Using the result of [BSW03], it follows that $\Pr[D(\mathbf{w}) = \sigma] \leq \delta + \epsilon$. For the other direction, assume that there does not exist a 2^k-quantified ϵ-distinguisher of size S for $\mathbf{w}$, and let $D\colon \{0,1\}^n \to \{0,1\}$ be a size-S circuit. We claim that for every $\sigma \in \{0,1\}$ it holds that $\Pr[D(\mathbf{w}) = \sigma] \leq \Pr[D(\mathbf{u}_n) = \sigma] \cdot 2^{n-k} + \epsilon$. To see this, note that if $|D^{-1}(\sigma)| > 2^k$ then the bound is trivial; and otherwise, it cannot be that $\Pr[D(\mathbf{w}) = \sigma] > \Pr[D(\mathbf{u}_n) = \sigma] \cdot 2^{n-k} + \epsilon \geq \epsilon$, since that would mean that D is a 2^k-quantified ϵ-distinguisher of size S for $\mathbf{w}$. ■

3.3 Well-known algorithmic constructions

In this section we will state several well-known algorithmic results that we will use in our proofs: Namely, constructions of a hash function, of a randomness extractor, and of an error-correcting code. First, we will need the following construction of a pairwise-independent hash function that is computable in quasilinear time:

Theorem 3.12 (quasilinear-time pairwise-independent hashing)**.** *For every $m, m' \in \mathbb{N}$ there exists a family $\mathcal{H} \subseteq \left\{\{0,1\}^m \to \{0,1\}^{m'}\right\}$ of quasilinear-sized circuits such that for every distinct $x, x' \in \{0,1\}^m$ it holds that $\Pr_{h\in\mathcal{H}}[h(x) = h(x')] \leq 2^{-m'}$.*

Proof. We use convolution hashing (see [MNT93]): Any $h = h_{a,b} \in \mathcal{H}$ is uniquely defined by a pair $a \in \{0,1\}^{m+m'-1}$ and $b \in \{0,1\}^{m'}$ such that $h_{a,b}(x) = a * x + b$, where the i^{th} output bit of the convolution operator is $(a * x)_i = \sum_{j\in[m]} a_{i+j-1} \cdot x_i \pmod 2$. Using the Fast Fourier Transform, the complexity of computing $h_{a,b}$ (for a fixed a, b) is $\tilde{O}(n)$ (see [CLR+09, Thm 30.8]). ■

In the current work we will need an averaging sampler, or equivalently a seeded randomness extractor. As noted in [DMO+20], a construction of an such an extractor in [TSZS06, Thm 5] can be analyzed with sub-constant min-entropy rates, and this is indeed what we will need in the current work. We first define the corresponding notions, then state the well-known equivalence between extractors and averaging samplers, and finally state the result from [DMO+20], following [TSZS06].

Definition 3.13 (min-entropy)**.** *We say that a random variable $\mathbf{x}$ has* min-entropy *k if for every $x \in \mathsf{supp}(\mathbf{x})$ is holds that $\Pr[\mathbf{x} = x] \leq 2^{-k}$.*

Definition 3.14 (seeded extractor)**.** *A function $\mathtt{Ext}\colon [N] \times \{0,1\}^\ell \to [M]$ is a (k, δ)-extractor if for every random variable $\mathbf{x}$ over $[N]$ with min-entropy k it holds that $\mathtt{Ext}(\mathbf{x}, \mathbf{u}_\ell)$ is δ-close in statistical distance to $\mathbf{u}_m$. The value ℓ is the* seed length *of the extractor.*

Definition 3.15 (averaging samplers)**.** *A function $f : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ is an* averaging sampler with accuracy $\epsilon > 0$ and error $\delta > 0$ *(or* (ϵ, δ)-averaging sampler, *in short) if it satisfies the following. For every $T \subseteq \{0,1\}^m$, for all but a δ-fraction of the strings $x \in \{0,1\}^n$ it holds that $\Pr_{z\in\{0,1\}^t}[f(x,z) \in T] = |T|/2^m \pm \delta$. We will also identify f with a function $\mathsf{Samp}\colon \{0,1\}^n \to (\{0,1\}^m)^{2^t}$ in the natural way (i.e., $\mathsf{Samp}(x)_i = f(x,i)$).*

Proposition 3.16 (seeded extractors are equivalent to averaging samplers; see, e.g., [Vad12, Cor 6.24])**.** *Let $f : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$. Then, the following two assertions hold:*

1. *If f is a (k,ϵ)-extractor, then f is an averaging sampler with accuracy ϵ and error $\delta = 2^{k-n}$.*

2. *If f is an averaging sampler with accuracy ϵ and error δ, then f is an $(n - \log(\epsilon/\delta), 2\epsilon)$-extractor.*

Theorem 3.17 (an extractor with near-logarithmic seed length; see [DMO+19, Lem 7.10], following [TSZS06, Thm 5])**.** *There exists a constant $c \geq 1$ such that for every $\gamma < 1/2$ the following holds. There exists a strong (k,ϵ)-extractor $\texttt{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ for $k = n^{1-\gamma}$ and $\epsilon \geq c \cdot n^{-1/2+\gamma}$ and $d \leq (1 + c\cdot\gamma)\cdot\log(n) + c\cdot\log(1/\epsilon)$ and $m = \frac{1}{c}\cdot n^{1-2\gamma}$. Moreover, the extractor is computable in linear time.*

Finally, we will need a locally list-decodable code, and we will use the Reed-Muller code. We first define this notion then state the result of [STV01], which asserts that the RM code is indeed locally list-decodable.

Definition 3.18 (locally list-decodable code)**.** *We say that $Enc\colon \Sigma^N \to \Sigma^M$ is* locally list-decodable from agreement ρ with decoding circuit size s and output-list size L *if there exists a randomized oracle circuit $Dec\colon [N]\times[L] \to \Sigma$ of size s such that following holds. For every $z \in \Sigma^M$ that satisfies $\Pr_{i\in[M]}[z_i = Enc(x)_i]$ for some $x \in \Sigma^N$ there exists $a \in [L]$ such that for every $i \in [N]$ we have that $\Pr[Dec^z(i,a) = x_i] \geq 2/3$, where the probability is over the internal randomness of Dec.*

Theorem 3.19 (the Reed-Muller code is locally list-decodable; see [STV01, Thm. 29])**.** *Let $RM\colon \mathbb{F}_q^{\binom{t+d}{d}} \to \mathbb{F}_q^{q^t}$ be the t-variate Reed-Muller code of degree d over $\mathbb{F}_q$. Then, for every $\rho \geq \sqrt{8\cdot(d/q)}$ it holds that RM is locally list-decodable from agreement ρ with decoding circuit size $\mathrm{poly}(t, \log(q), d, 1/\rho)$ and output-list size $O(1/\rho)$.*

Corollary 3.20 (a locally list-decodable code)**.** *For every constant $\eta > 0$ there exists a constant $\eta' > 0$ such that the following holds. For every $m \in \mathbb{N}$ and $\rho = \rho(m)$ there exists a code $Enc\colon \{0,1\}^m \to \Sigma^{\bar{m}}$, where $|\Sigma| = O(m^{\eta'}/\rho^2)$ and $\bar{m} = O_{\eta'}\left(m/\rho^{2/\eta'}\right)$, such that:*

1. *The code is computable in time $\tilde{O}(\bar{m}\cdot\log(|\Sigma|)) = \tilde{O}(m/\rho^{2/\eta'})$.*

2. *The code is locally list-decodable from agreement ρ with decoding circuit size $m^\eta \cdot (1/\rho)^{1/\eta'}$ and output list size $O(1/\rho)$.*

Proof. For $\eta' < \eta$ be sufficiently small, let $t = 1/\eta'$, let $d = m^{\eta'}$, and let $q = (8/\rho^2)\cdot d$. We use the t-variate Reed-Muller code of degree d over $\mathbb{F}_q$, whose input (of bits) is of length $\binom{t+d}{d}\cdot\log(q) \geq (d/t)^t \cdot \log(q) \geq m$ and output (of elements in $\mathbb{F}_q$) is of length $\bar{m} = q^t = O_{\eta'}\left(m/\rho^{2/\eta'}\right)$. By Theorem 3.19, this code is locally list-decodable from agreement ρ with decoding circuit size $\mathrm{poly}(t, \log(q), d, 1/\eta) = \mathrm{poly}(m^{\eta'}/\rho) \leq m^\eta \cdot (1/\rho)^{1/\eta'}$ (choosing η' to be sufficiently small) and output list size $O(1/\rho)$. ∎

4 Derandomization with almost no slowdown

In this section we prove the results that were presented in Section 1.2, namely Theorems 1.2, 1.3, and 1.4, and Proposition 1.6, with the sole exception that the lower bound stated in Theorem 1.3 is proved in Section 6.

As mentioned in Section 2, one of our basic ideas will be to compose two PRGs that are computable in near-linear time but have certain (respective) shortcomings, in order to obtain one PRG that does not suffer from these shortcomings. In Section 4.1 we show that the two foregoing PRGs exist, one of them unconditionally and the other under the hypothesis that non-uniformly secure one-way functions exist. Then, in Section 4.2 we show how to compose these PRGs in order to obtain a single (and better) PRG. Next, in Section 4.3 we prove Theorem 1.2 as well as a converse direction, which asserts that if PRGs with our parameters exist then there exists a hard problem as in the hypothesis of Theorem 1.2 (i.e., we prove Proposition 1.6). And finally, in Section 4.4 we prove several extensions and optimizations of Theorem 1.2, namely Theorems 1.3 and 1.4.

Throughout the section we will ignore rounding issues for simplicity, since rounding issues do not significantly affect our proofs. Nevertheless, to avoid confusion, let us state in advance that the notation $\mathcal{DTIME}[n]/T^{-1}$ will denote the class of functions computable in linear time with $\lceil T^{-1}(n) \rceil$ bits of advice (i.e., the advice length is the minimal m such that $T(m) \geq n$).

4.1 Near-linear-time computable PRGs

As described in Section 2, we will compose a PRG that has a small seed $(1.01) \cdot \log(n)$ but short output length n^{ϵ}, with a PRG that has a relatively-long seed n^{ϵ} and a long output length n. The first of the two PRGs can be constructed using an instantiation of the classic Nisan-Wigderson PRG [NW94], with modifications a-la [RRV02]; this instantiation yields the following result:

Theorem 4.1 (NW Generator for small output length)**.** *There exists a universal constant c_{nw} such that for all sufficiently small ϵ_{nw}, there exists an oracle machine G satisfies the following:*

- *When given input $1^{N^{\epsilon_{nw}}}$ and oracle access to a function $f\colon \{0,1\}^{\log(N)} \to \{0,1\}$, the machine G runs in time $N^{1+c_{nw}\cdot\sqrt{\epsilon_{nw}}}$ and outputs $2^{\ell_{nw}(N)}$ strings in $\{0,1\}^{N^{\epsilon_{nw}}}$, where $\ell_{nw}(N) = (1 + c_{nw}\sqrt{\epsilon_{nw}}) \cdot \log N$.* [20]
- *There exists an oracle machine R that, when given input $x \in \{0,1\}^{\log N}$ and oracle access to an $(N^{-\epsilon_{nw}})$-distinguisher for $G(1^N, \mathbf{u}_{\ell(N)})^f$ and $N^{1-\sqrt{\epsilon_{nw}}/c_{nw}}$ bits of advice, runs in time $N^{c_{nw}\cdot\sqrt{\epsilon_{nw}}}$ and outputs $f(x)$.*

The proof of Theorem 4.1 is essentially a different parametrization of well-known proofs, and we provide a proof sketch in Appendix A for completeness. The second of the two PRGs that we need is a near-linear time computable PRG with polynomial stretch that "fools" linear-sized circuits. Such a PRG is not known to follow from standard hardness assumptions for non-uniform circuits, where the main challenge is obtaining a near-linear runtime in the output length (cf., the PRG of [BFN+93] has polynomial stretch but runs in sub-exponential time).

[20]For simplicity, we bounded both the seed length and the running time using the same parameter (i.e., $c_{nw} \cdot \sqrt{\epsilon_{nw}}$) such that the running time is precisely $2^{\ell_{nw}(N)}$. In the actual construction the seed length is smaller than $1 + c_{nw} \cdot \sqrt{\epsilon_{nw}}$ (see Appendix A for details).

Assumption 4.2 (near-linear-time computable PRG with arbitrary polynomial stretch)**.** *For every $\epsilon > 0$, there exists a $(1/n)$-PRG with seed length $\ell(n) = n^{\epsilon}$ that is computable in time $n^{1+\epsilon}$.*

We show that a PRG as in Assumption 4.2 exists, under the hypothesis that there exist one-way functions secure against polynomial-sized circuits. The proof of this claim amounts to using the classic construction of PRGs from one-way functions (OWFs) [HIL+99], and then applying standard techniques to extend the expansion factor of PRGs (see, e.g., [Gol01, Const. 3.3.2]). That is:

Proposition 4.3 (OWF $\Rightarrow$ near-linear-time PRG)**.** *If there exists a polynomial-time computable one way function that is secure against circuits of arbitrary polynomial size, then Assumption 4.2 is true. Moreover, for some negligible function* $\texttt{neg}$ *and any polynomial p, the resulting PRG is* $\texttt{neg}$*-pseudorandom for circuits of size $p(n)$ (rather than only $1/n$-pseudorandom for circuits of linear size).*

Proof Sketch. By [HIL+99], our hypothesis implies that for some negligible function $\texttt{neg}$ there exists an s-PRG for $\mathcal{P}/\text{poly}$ with seed length $\ell(n) = n/2$ that is computable in polynomial time. In more detail, for some constant $c \in \mathbb{N}$ there exists G_1 that extends an m-bit seed to a $2m$-bit output such that G_1 is computable in time m^c, and for all $k \in \mathbb{N}$, no m^k-size circuit can distinguish between $G_1(U_m)$ and U_{2m} with advantage at least $\texttt{neg}(m)$.

Now, let $\epsilon > 0$. Our PRG G gets input 1^n and a seed x of length $m = n^{\epsilon/2c} < n^{\epsilon}$ and acts as follows:

- Let $\sigma_1 = x$.
- For all $i \in \{2, 3, \ldots, n/m\}$, we set σ_i to be the last m bits of $G_1(\sigma_{i-1})$.
- The output of $G(x)$ is the concatenation of all $\sigma_1, \sigma_2, \ldots, \sigma_{n/m}$.

Note that G outputs m bits, and its running time is at most $m^c \cdot n + O(n) \leq n^{1+\epsilon}$. A standard hybrid argument (as in [Gol01, Thm 3.3.3]) shows that for every polynomial p it holds that G is a $(1/p(n))$-PRG for circuits of size $p(n)$. ■

When extending Theorem 1.2 to super-polynomial time functions $T(n) = n^{\omega(1)}$ we will use a hypothesis stronger than Assumption 4.2; specifically, we will assume that there exists a PRG with sub-exponential (rather than polynomial) stretch that is computable in near-linear time. Analogously to Proposition 4.3, such a PRG follows from the existence of a OWF that is secure against circuits of sub-exponential size.

Assumption 4.4 (near-linear-time computable PRG with sub-exponential stretch)**.** *For some constant c there exists a $(1/n)$-PRG with seed length $\ell(n) = (\log(n))^c$ that is computable in time $n \cdot (\log(n))^c$.*

Proposition 4.5 (OWF against sub-exponential circuits $\Rightarrow$ near-linear-time PRG with sub-exponential stretch)**.** *If there exists a polynomial-time computable one-way function that is secure against circuits of size $2^{n^{\epsilon}}$ (for some constant $\epsilon > 0$), then Assumption 4.4 is true.*

We omit the proof of Proposition 4.5, since it is nearly identical to the proof of Proposition 4.3.

4.2 Composing two near-linear time PRGs

We now show that the two PRGs that were mentioned in Section 4.1 can be composed to obtain a single PRG that has both a short seed length and small running time. Moreover, we will instantiate this PRG with parameters that are suitable for our application, which involves very small truth-tables that are hard for algorithms with super-exponential time complexity. Specifically, in the following result, our goal will be to "fool" the class $\mathcal{DTIME}[O(n)]/A$, and we will do so using a PRG with seed length $1.01 \cdot \log(A(n))$ and running time $A(n)^{1.01} \cdot n$; that is:

Proposition 4.6 (super-exponential hardness to near-optimal randomness by composing two "low-cost" PRGs)**.** *There exists a universal constant c such that for every $\epsilon > 0$ there exists $\delta_0 > 0$ and $\delta_2 > \delta_1 > 0$ for which the following holds. For any time-constructible non-increasing function $A \colon \mathbb{N} \to \mathbb{N}$ such that $A(\mathbb{N}) = \mathbb{N}$ and $A^{-1}(n) = \min\{N \in \mathbb{N} : A(N) = n\}$ is time-constructible, assume that:*

1. *There exists a $(1/n)$-PRG with seed length $A(n)^{\delta_0}$ and running time $n \cdot A(n)^{\delta_0}$.*

2. *For $T_{A^{-1}}(n) = 2^{(c\cdot\delta_1)\cdot n} \cdot A^{-1}\left(2^{(1-\delta_1)\cdot n}\right)$, there exists $L \in \texttt{amort-}\mathcal{DTIME}[2^{\delta_2 \cdot n} \cdot T_{A^{-1}}(n)]$ such that $L \notin \texttt{i.o.}\mathcal{DTIME}\left[T_{A^{-1}}\right]/2^{(1-\delta_1)\cdot n+1}$.*[21]

Then, there exists an $(n^{-\delta_0/2})$-PRG for $\mathcal{DTIME}[O(n)]/A$ that on input 1^n uses a seed of length $(1+\epsilon)\cdot\log(A(n))$ and is batch-computable in time $A(n)^{1+\epsilon}\cdot n$.

Proof. Let c_{nw} be the constant from Theorem 4.1. Let $\delta_1, \delta_0 > 0$ be sufficiently small constants to be specified later, and let c be a universal constant to be specified later. We first describe the construction of the PRG G.

Construction of the PRG G. It will be more convenient to denote the input to the PRG by 1^N rather than 1^n. For $N \in \mathbb{N}$, let $n = A(N)$ and let $\ell = \frac{\log(n)}{1-\delta_1}$ (such that $2^\ell = n^{1/(1-\delta_1)}$). By our hypothesis, the truth-table of L_ℓ (i.e., the restriction of the hypothesized L to ℓ-bit inputs) can be printed in time $2^{(1+c\cdot\delta_1+\delta_2)\cdot\ell} \cdot A^{-1}\left(2^{(1-\delta_1)\cdot\ell}\right) \leq 2^{(1+c\cdot\delta_1+\delta_2)\ell} \cdot N$, whereas $L \notin \texttt{i.o.}\mathcal{DTIME}[2^{(c\cdot\delta_1)\cdot\ell} \cdot A^{-1}(n)]/2^{\ell(1-\delta_1)+1}$.

Let G_{cry} be the $(1/n)$-PRG from our first hypothesis, and let G_{nw} be the oracle machine from Theorem 4.1, instantiated with parameter $\epsilon_{nw} = (1-\delta_1)\cdot\delta_0$. Note that for $N_{nw} = |L_\ell| = n^{1/(1-\delta_1)}$, when G_{nw} gets input $1^{N_{nw}^{\epsilon_{nw}}}$ and oracle access to L_ℓ, it uses a seed of length $\ell_{nw} = (1 + c_{nw}\cdot\sqrt{\epsilon_{nw}})\cdot\ell = \frac{1+c_{nw}\cdot\sqrt{\epsilon_{nw}}}{1-\delta_1}\cdot\log n$, and outputs a string of length $N_{nw}^{\epsilon_{nw}} = 2^{\epsilon_{nw}\cdot\ell} = n^{1/(1-\delta_1)\cdot\epsilon_{nw}} = n^{\delta_0}$.

On input 1^N and given a seed $w \in \{0,1\}^{\ell_{nw}}$, the machine G outputs

$$G(1^N, w) = G_{cry}(1^N, G_{nw}(1^{n^{\delta_0}}, w))\ ;$$

that is, G outputs the composition of G_{nw} and G_{cry}.

[21] Recall that the definition of $\texttt{amort-}\mathcal{DTIME}[T]$ appears in Definition 1.5.

Analysis. Consider any $F \in \mathcal{DTIME}[O(N)]/A$ (to reflect the relationship with the notation above more clearly, we denote the input length to the time function by N rather than n). Assume towards a contradiction that there are infinitely many $N \in \mathbb{N}$ such that $G(1^N, \mathbf{u}_{(1+\epsilon)\cdot\log(A(N))})$ does not $(N^{-\delta_0/2})$-fool F on inputs of length N. [22]

Let D be an algorithm that gets an input $w \in \{0,1\}^m$, computes $n = m^{1/\delta_0}$ and $N = A^{-1}(n)$, and outputs $F(G_{cry}(1^N, w))$. Note that D can be computed on inputs of length n^{δ_0} in time $O(N \cdot n^{\delta_0})$ and with n bits of advice. Now, recall that G_{cry} is $(1/N)$-pseudorandom for F, and therefore

$$\left| \mathop{\mathbb{E}}_{x\in\{0,1\}^N}[F(x)] - \mathop{\mathbb{E}}_{w\in\{0,1\}^{n^{\delta_0}}}[D(w)] \right| \leq 1/N \,. \tag{1}$$

It follows that for any distribution $\mathbf{w}$ over $\{0,1\}^{n^{\delta_0}}$, if F on inputs of length N is an $(N^{-\delta_0/2})$-distinguisher for $G_{cry}(1^N, \mathbf{w})$, then D on inputs of length n^{δ_0} is a $(N^{-\delta_0/2} - 1/N)$-distinguisher for $\mathbf{w}$. (This is since $\mathbb{E}_{w\in\{0,1\}^{n^{\delta_0}}}[D(w)]$ is $1/N$-close to $\mathbb{E}_{x\in\{0,1\}^N}[F(x)]$, but the latter is $(N^{-\delta_0/2})$-far from $G_{cry}(1^N, \mathbf{w})$.) In particular, there are infinitely many $n = A(N)$ such that D on inputs of length n^{δ_0} is a $(N^{-\epsilon_{nw}})$-distinguisher for $G_{nw}^{L_\ell}(1^{n^{\delta_0}}, \mathbf{u}_{\ell_{nw}})$.

Fix an n as above, and recall the notation $\ell = \frac{\log(n)}{1-\delta_1}$. By Theorem 4.1, there is an oracle machine R that computes L_ℓ when given oracle access to $D_{n^{\delta_0}}$. Plugging the time and advice complexity of D_n, we obtain an algorithm that decides L_ℓ in time

$$O\left(N_{nw}^{c_{nw}\sqrt{\epsilon_{nw}}} \cdot n^{\delta_0} \cdot N\right) \leq 2^{(\epsilon_{nw}+c_{nw}\sqrt{\epsilon_{nw}})\cdot\ell} \cdot A^{-1}\left(2^{(1-\delta_1)\cdot\ell}\right)$$

when given an advice string of length

$$N_{nw}^{1-\sqrt{\epsilon_{nw}}/c_{nw}} + n = 2^{(1-\sqrt{\epsilon_{nw}}/c_{nw})\cdot\ell} + 2^{(1-\delta_1)\cdot\ell} \,.$$

Setting the parameters. The above shows that any distinguisher F can be converted to an algorithm that decides L infinitely-often; we now set the parameters to obtain a contradiction. We set $\delta_0 = (c_{nw}\cdot\delta_1)^2/(1-\delta_1)$, which implies that $\sqrt{\epsilon_{nw}}/c_{nw} = \delta_1$ (since we defined $\epsilon_{nw} = (1-\delta_1)\cdot\delta_0$). The number of advice bits is then bounded by $2^{(1-\delta_1)\cdot\ell+1}$, and the running time can be bounded by

$$2^{(\delta_1^2 c_{nw}^2+\delta_1 c_{nw}^2)\cdot\ell} \cdot A^{-1}\left(2^{(1-\delta_1)\cdot\ell}\right) \leq 2^{(2\delta_1 c_{nw}^2)\cdot\ell} \cdot A^{-1}\left(2^{(1-\delta_1)\cdot\ell}\right) \,,$$

which contradicts our assumption about L if we set the universal constant c in the statement to be $2c_{nw}^2$. Therefore, G is an $(N^{-\delta_0/2})$-PRG for $\mathcal{DTIME}[O(N)]/A$.

The seed length of G on input 1^N is $(1+c_{nw}\cdot\sqrt{\epsilon_{nw}})/(1-\delta_1)\cdot\log(A(N)) = (1+\delta_1\cdot c_{nw}^2)/(1-\delta_1)\cdot\log(A(N))$, which is smaller than $(1+\epsilon)\cdot\log(A(N))$ if we set $\delta_1 > 0$ to be sufficiently small. The batch-computation time of G on input 1^N is bounded by the time it takes to compute the truth-table of L_ℓ, the time it takes to batch-compute G_{nw},

[22]Recall that the seed length of G for output length N is $\frac{1+c_{nw}\cdot\sqrt{\epsilon_{nw}}}{1-\delta_1}\cdot\log(A(N))$. We will set the parameters below such that this equals $(1+\epsilon)\cdot\log(A(N))$.

and the time it takes to apply G_{cry} to each output of G_{nw}; this is at most

$$\begin{aligned} &O\Big(2^{(1+c\cdot\delta_1+\delta_2)\ell}\cdot A^{-1}\Big(2^{\ell(1-\delta_1)}\Big)+2^{(1+c_{nw}\cdot\sqrt{\epsilon_{nw}})\cdot\ell}\cdot N\cdot n^{\delta_0}\Big) \\ &\leq N\cdot O\left(n^{(1+c\cdot\delta_1+\delta_2)/(1-\delta_1)}+n^{(1+c_{nw}\cdot\sqrt{\epsilon_{nw}})/(1-\delta_1)+\delta_0}\right) \\ &= N\cdot O\left(n^{(1+c\cdot\delta_1+\delta_2)/(1-\delta_1)}+n^{(1+\delta_1\cdot c_{nw}^2+(c_{nw}\cdot\delta_1)^2)/(1-\delta_1)}\right) , \end{aligned}$$

which is at most $N\cdot n^{1+\epsilon}=N\cdot A(N)^{1+\epsilon}$ if δ_1 and δ_2 are sufficiently small. ■

4.3 Proof of Theorem 1.2 and of a converse direction

As explained in the introduction, our first observation towards proving Theorem 1.2 is that in order to derandomize $pr\mathcal{BPTIME}[T]$, we do not actually need to "fool" non-uniform circuits, but only to "fool" the class $\mathcal{DTIME}[O(n)]/T^{-1}$. [23] Let us formally prove this statement:

Proposition 4.7 (PRGs for "$\mathcal{DTIME}$ with bounded advice" suffice for derandomization)**.** *For any time-constructible and increasing $T\colon\mathbb{N}\to\mathbb{N}$, if there exists a $(1/8)$-PRG for $\mathcal{DTIME}[O(n)]/T^{-1}$ with seed length $\ell(n)$ that is batch-computable in time $W(n)$, then*

$$pr\mathcal{BPTIME}[T]\subseteq pr\mathcal{DTIME}\left[O\Big(W(T(n))+2^{\ell(T(n))}\cdot T(n)\Big)\right] .$$

Proof. Let $\Pi=(Y,N)\in pr\mathcal{BPTIME}[T]$ and let M be a randomized time-T algorithm that solves Π. Given input $x\in\{0,1\}^n$, we invoke the PRG in our hypothesis, denoted G, on input $1^{T(n)}$ and all possible seeds to generate $2^{\ell(T(n))}$ outputs $\{G(1^{T(n)},w)\}_{w\in\{0,1\}^{\ell(T(n))}}$, and accept x if and only if

$$\Pr_{w\in\{0,1\}^{\ell(T(n))}}\left[M_x(G(1^{T(n)},w))=1\right]>1/2 ,$$

where $M_x\colon\{0,1\}^{T(|x|)}\to\{0,1\}$ accepts its input $r\in\{0,1\}^{T(|x|)}$ if and only if M accepts x when using randomness r.

The foregoing algorithm runs in time $O\Big(W(T(n))+2^{\ell(T(n))}\cdot T(n)\Big)$. To show its correctness, assume towards a contradiction that it does not solve Π. Then, there are infinitely many $m\in\mathbb{N}$ (we call them bad m's) such that there exists $x_m\in\{0,1\}^m$ for which M_{x_m} is an $(1/8)$-distinguisher for $G(1^{T(n)},\mathbf{u}_{\ell(T(n))})$.

Consider the following algorithm: Given $x\in\{0,1\}^n$, if $n\notin\{T(m):m\in\mathbb{N}\}$, accept; otherwise, when $n=T(m)$ for $m\in\mathbb{N}$, we denote the m-bit advice string by $a_m\in\{0,1\}^m$, and output $M_{a_m}(x)$. Note that this algorithm yields a function $D\in\mathcal{DTIME}[O(n)]/T^{-1}$, and that when given advice $a_m=x_m$ for all bad $m\in\mathbb{N}$ it holds that D on inputs of length $T(m)$ computes M_{x_m}, and is therefore a $(1/8)$-distinguisher for $G(1^{T(m)},\mathbf{u}_{\ell(T(n))})$. This contradicts our hypothesis about G. ■

Note that in the foregoing proof we only relied on the fact that the PRG "fools" $\mathcal{DTIME}[O(n)]/T^{-1}$ on inputs of length $n=T(m)$ for some $m\in\mathbb{N}$, rather than on all input lengths. For simplicity we did not explicitly state this relaxed hypothesis.

[23] Recall that we ignore rounding issues for simplicity, and that $\mathcal{DTIME}[O(n)]/T^{-1}$ is the class of linear-time algorithms whose advice length is $\lceil T^{-1}(n)\rceil$.

We now prove Theorem 1.2. We actually prove the result in two parts, each of which refers to a different parameter setting and asserts a stronger result. The first part refers to probabilistic polynomial-time algorithms, and for this setting we show derandomization in time $n^{1+\epsilon} \cdot T(n)$, which is better than the time bound of $n \cdot T(n)^{1+\epsilon}$ stated in Theorem 1.2. [24] In more detail:

Theorem 4.8 (derandomization with almost no overhead for polynomial-time algorithms)**.** *There exists a universal constant $c > 1$ such that for every $\epsilon > 0$ there exist $\rho > 0$ and $\delta_2 > \delta_1 > 0$ for which the following holds. For any constant $k \geq 1$, assume that there exist one-way functions that are secure against polynomial-sized circuits, and that for $T_k(n) = 2^{(1-\delta_1)\cdot kn+(c\cdot\delta_1)\cdot n}$ there exists $L \in \texttt{amort-}\mathcal{DTIME}[2^{\delta_2 \cdot n} \cdot T_k(n)]$ such that $L \notin \texttt{i.o.}\mathcal{DTIME}[T_k]/2^{(1-\delta_1)\cdot n+1}$. Then, there exists an $(n^{-\rho})$-PRG for $\mathcal{DTIME}[O(n)]/n^{1/k}$ with seed length $(1+\epsilon)\cdot(1/k)\cdot\log(n)$ that is batch-computable in time $n^{1+1/k+\epsilon/k}$. Consequently, we have that $pr\mathcal{BPTIME}[n^k] \subseteq pr\mathcal{DTIME}[n^{k+1+\epsilon}]$.*

Proof. We will instantiate Proposition 4.6 with parameter value $\epsilon' = \epsilon/3$, and denote by $\delta'_0, \delta'_1, \delta'_2$ be the three constants from Proposition 4.6 corresponding to ϵ'. We set $\delta_0 = \delta'_0$, and $\delta_1 = \delta'_1$, and $\delta_2 = \delta'_2/2$. Relying on Proposition 4.3 and on our hypothesis, there exists a $(1/n)$-PRG that on input 1^n uses a seed of length $n^{\delta_0/k}$ and is computable in time $n^{1+\delta_0/k}$.

Let $A(N) = \lceil N^{1/k} \rceil$, and let $A^{-1}(n) = (n-1)^k + 1 = \min\{N \in \mathbb{N} : A(N) = n\}$. Let f be the hard function from our hypothesis, and note that for every $\ell \in \mathbb{N}$, the truth-table of L_ℓ can be printed in time

$$\begin{aligned} 2^{\ell} \cdot 2^{(1-\delta_1)\cdot k\ell+(c\cdot\delta_1+\delta_2)\cdot\ell} &< 2^{(1+c\delta_1+\delta_2)\cdot\ell} \cdot 2 \cdot A^{-1}(2^{(1-\delta_1)\cdot k\ell}) \\ &< 2^{(1+c\delta_1+2\delta_2)\cdot\ell} \cdot A^{-1}(2^{(1-\delta_1)\cdot k\ell}) \, , \end{aligned}$$

whereas L_ℓ cannot be computed in time $2^{(c\cdot\delta_1)\cdot\ell} \cdot 2^{(1-\delta_1)\cdot k\ell} > 2^{(c\cdot\delta_1)\cdot\ell} \cdot A^{-1}(2^{(1-\delta_1)\cdot k\ell})$ when given at most $2^{(1-\delta_1)\cdot\ell+1}$ bits of advice. Thus, relying on Proposition 4.6, we can invoke the proposition with the parameter value ϵ'.

We deduce that there exists an $(n^{-\delta_0/2})$-PRG for $\mathcal{DTIME}[O(n)]/A$ with seed length $\ell(n) = (1+\epsilon')\cdot\log(A(n)) < (1+2\epsilon')\cdot(1/k)\cdot\log(n)$ that is batch-computable in time $W(n) = A(n)^{1+\epsilon'} \cdot n < n^{1+1/k+2\epsilon'/k}$. Using Proposition 4.7 with $T(n) = n^k$, we deduce that $pr\mathcal{BPTIME}[n^k] \subseteq pr\mathcal{DTIME}[O(n^{k+1+2\epsilon'})] \subset pr\mathcal{DTIME}[n^{k+1+\epsilon}]$. ■

Next we turn to the setting of superpolynomial-time algorithms. Note that for such algorithms there is no meaningful difference between derandomization in time $T(n)^{1+\epsilon}$ and $n \cdot T(n)^{1+\epsilon}$ (as we take $\epsilon > 0$ to be arbitrarily small), and we will indeed show a derandomization with the former time bound. Moreover, for this setting the required gaps between the upper-bound and the lower-bound in the hardness hypothesis (represented by parameters δ_1, δ_2) will be universal, rather than having the required gaps depend on the target derandomization overhead (represented by the parameter $\epsilon > 0$). That is:

Theorem 4.9 (derandomization with almost no overhead for superpolynomial-time algorithms)**.** *There exist universal constants $c > 1$ and $\delta_2 > \delta_1 > 0$ such that for every*

[24]Indeed, the difference between the two can be bridged by taking a sufficiently small $\epsilon > 0$ that depends on T (i.e., if $T(n) = n^k$ we set $\epsilon = \epsilon'/k$), but in this case the hardness hypothesis becomes less natural and does not match Theorem 4.11 (i.e., the hypothesized hardness will be $2^{(1-\delta)\cdot kn}$ for $\delta \ll 1/k$).

$\epsilon > 0$ and every $k \geq c/\epsilon$ the following holds. Assume that there exist one-way functions that are secure against polynomial-sized circuits, and that for $T_k(n) = 2^{(1-\delta_1)\cdot kn + (c\cdot\delta_1)\cdot n}$ there exists $L \in \texttt{amort-}\mathcal{DTIME}[2^{\delta_2\cdot n} \cdot T_k]$ such that $L \notin \texttt{i.o.}\mathcal{DTIME}[T_k]/2^{(1-\delta_1)\cdot n+1}$. Then, for any time-constructible and increasing $T(n) = n^{\omega(1)}$ we have $pr\mathcal{BPTIME}[T(n)] \subseteq pr\mathcal{DTIME}[T(n)^{1+\epsilon}]$.

Proof. For a sufficiently large $k \geq 1$, we instantiate Theorem 4.8 with parameter values $\epsilon = 1$ and $k+1$, to deduce that our hypothesis implies that $pr\mathcal{BPTIME}[n^{k+1}] \subseteq pr\mathcal{DTIME}[n^{k+3}]$. Fixing any $T(n) = n^{\omega(1)}$ as in our hypothesis and fixing any promise-problem $\Pi \in pr\mathcal{BPTIME}[T(n)]$, we define a padded promise-problem Π' whose "yes" instances are $\left\{x0^{T(|x|)^{1/k}-|x|} : x \text{ is a yes instance for } \Pi\right\}$ and whose "no" instances are $\left\{x0^{T(|x|)^{1/k}-|x|} : x \text{ is a no instance for } \Pi\right\}$. Note that $\Pi' \in pr\mathcal{BPTIME}[O(n^k)] \subseteq pr\mathcal{BPTIME}[n^{k+1}]$, and hence $\Pi' \subseteq pr\mathcal{DTIME}[O(n^{k+3})]$. By a padding argument it follows that $\Pi \in \mathcal{DTIME}[O(T(n)^{1+\frac{2}{k+3}})]$. Assuming that k is sufficiently large such that $\frac{2}{k+3} < \epsilon$, we have that $\Pi \in pr\mathcal{DTIME}[T(n)^{1+\epsilon}]$. ■

We now combine Theorems into a proof of Theorem 1.2. We will actually state and prove a stronger version of Theorem 1.2, in which the assumed upper-bound on the hard function is amortized, rather than in worst-case. That is:

Theorem 4.10 (Theorem 1.2, stronger version)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that the following holds. Let $T\colon \mathbb{N} \to \mathbb{N}$ be any time-constructible non-decreasing function, and let $k = k_{\epsilon,T} \geq 1$ be a sufficiently large constant. Assume that there exist one-way functions that are secure against polynomial-sized circuits, and that there exists $L \in \texttt{amort-}\mathcal{DTIME}[2^{k\cdot n}]$ such that $L \notin \texttt{i.o.}\mathcal{DTIME}[2^{(k-\delta)\cdot n}]/2^{(1-\delta)\cdot n}$. Then, we have that $pr\mathcal{BPTIME}[T(n)] \subseteq pr\mathcal{DTIME}[n \cdot T(n)^{1+\epsilon}]$.*

Proof. Given $\epsilon > 0$, let $\delta > 0$ be sufficiently small. In particular, we assume that δ is smaller than the values of δ_1 and δ_2 from Theorem 4.8 and of δ_1 and δ_2 from Theorem 4.9, when the latter two theorems are instantiated with the parameter value $\epsilon/2$.

Now, if $T(n) = n^{k_0}$ for some $k_0 \geq 1$, we let $k = k_{T,\epsilon} = (1-\delta)\cdot k_0 + (c+1)\cdot\delta > 1$, where $c > 1$ is the universal constant from Theorem 4.8. Our hypothesis is thus strong enough to instantiate Theorem 4.8 with parameter k_0 and conclude that $pr\mathcal{BPTIME}[n^k] \subseteq pr\mathcal{DTIME}[n^{k+1+\epsilon/2}]$. If $k \leq k_0$ then we rely on the fact that $k_0 < k/(1-\delta)$ and on a padding argument to deduce that $pr\mathcal{BPTIME}[n^{k_0}] \subseteq pr\mathcal{BPTIME}[n^{k/(1-\delta)}] \subseteq pr\mathcal{DTIME}[n^{\frac{k+1+\epsilon/2}{1-\delta}}] \subseteq pr\mathcal{DTIME}[n^{\frac{k_0+1+\epsilon/2}{1-\delta}}] \subset pr\mathcal{DTIME}[n \cdot n^{(1+\epsilon)\cdot k_0}]$, where the last containment relied on δ being sufficiently small. Otherwise, if $k > k_0$, then we rely on the fact that $k < k_0 + (c+1)\cdot\delta_0$ to deduce that $pr\mathcal{BPTIME}[n^{k_0}] \subseteq pr\mathcal{BPTIME}[n^k] \subseteq pr\mathcal{DTIME}[n^{k+1+\epsilon/2}] \subset pr\mathcal{DTIME}[n^{k_0+(c+1)\cdot\delta+1+\epsilon/2}] \subset pr\mathcal{DTIME}[n^{k_0+1+\epsilon}]$, where again the last containment relied on δ being sufficiently small.

For a super-polynomial T, let $k_0 \geq c/\epsilon$ be a sufficiently large constant (where $c > 1$ is the universal constant from Theorem 4.9) such that $k = (1-\delta)\cdot k_0 + c\cdot\delta \geq c/\epsilon$. Our hypothesis suffices to instantiate Theorem 4.9 with parameter k_0 and deduce that $pr\mathcal{BPTIME}[T(n)] \subseteq pr\mathcal{DTIME}[T(n)^{1+\epsilon}]$. ■

Finally, we show a partial inverse to Theorem 4.8. This partial inverse is the last missing piece needed to prove Proposition 1.6 (i.e., the proposition follows from Theorem 4.8 and from the partial inverse that we now prove). The proof is an adaptation of the standard proof that a PRG yields a function that is hard for non-uniform circuits, for our setting of a batch-computable PRG and distinguishers in "$\mathcal{DTIME}$ with advice".

Theorem 4.11 (batch-computable HSG $\Rightarrow$ hard function)**.** *For every $\epsilon, \epsilon' > 0$ there exists $\delta_1 > 0$ and $\delta_2 \geq 0$ such that for any time-constructible $T\colon \mathbb{N} \to \mathbb{N}$ the following holds. Assume that there exists a $(1/2)$-HSG for $\mathcal{DTIME}[O(n)]/T^{-1}$ with seed-length $\ell(n) = (1+\epsilon) \cdot \log(T^{-1}(n))$ that is batch-computable in time $T^{-1}(n)^{1+\epsilon'} \cdot n$. Then, for $T'(\ell) = T(2^{(1-\delta_1)\cdot\ell})$ there exists $L \in \texttt{amort-}\mathcal{DTIME}[O(2^{\delta_2\cdot\ell} \cdot T'(\ell))]$ such that $L \notin \texttt{i.o.}\mathcal{DTIME}[T']/2^{(1-\delta_1)\cdot\ell}$.*

Proof. For $\epsilon > 0$, let $\delta_1 = 1 - 1/(1+\epsilon)$. Let H be the HSG from the hypothesis, and note that for every $n \in \mathbb{N}$ it holds that $2^{\ell(T(n))} = n^{1+\epsilon}$, which by our choice of parameters implies that $n = 2^{(1-\delta_1)\cdot\ell(T(n))}$.

For $\ell \in \mathbb{N}$, let $n \in \mathbb{N}$ be the largest integer such that $\ell(T(n)) = \ell$.[25] We define L on inputs of length $\ell + 1$ such that $x \notin L$ if and only if there exists a seed $w \in \{0,1\}^\ell$ such that $H(1^{T(n)}, w)$ has prefix $\ell + 1$. Since H is batch-computable in time $T_H(N) = T^{-1}(N)^{1+\epsilon'} \cdot N$, it follows that L can be decided in time

$$T_L(\ell) = O\Big(n^{1+\epsilon'} \cdot T(n)\Big) = O\Big(2^\ell \cdot 2^{\delta_2\cdot\ell} \cdot T(2^{(1-\delta)\cdot\ell})\Big) \ ,$$

where $\delta_2 = \frac{\epsilon'-\epsilon}{1+\epsilon}$.

Now, assume towards a contradiction that there exists an algorithm A_L that runs in time $T(2^{(1-\delta_1)\cdot\ell})$ and ues $2^{(1-\delta_1)\cdot\ell}$ bits of advice that for infinitely many ℓ's decides L correctly on inputs of length $\ell + 1$ (we assume that A_L always runs in time $T(2^{(1-\delta_1)\cdot\ell})$, but we can only assume that it correctly computes L on infinitely-many input lengths). We define a Boolean function D such that on inputs of length $N = T(n)$ it holds that $D(x) = A_L(x_{\leq \ell+1})$, where $\ell = (1+\epsilon)\cdot\log(n)$ (we define D trivially on input lengths that are not of the form $T(n)$). Our assumption about A_L implies that D can be computed in time $O(T(2^{(1-\delta_1)\cdot\ell})) = O(T(n)) = O(N)$ with at most $2^{(1-\delta_1)\cdot\ell} \leq n = T^{-1}(N)$ bits of advice; that is, $D \in \mathcal{DTIME}[O(N)]/T^{-1}$. Also, for infinitely many input lengths N it holds that $D(H(1^N, w)) = 0$ for all $w \in \{0,1\}^\ell$, whereas $\Pr_{x\in\{0,1\}^N}[D(x)] \geq 1/2$. This contradicts our hypothesis that H is a $(1/2)$-HSG for $\mathcal{DTIME}[O(N)]/T^{-1}$. ■

4.4 Extensions and optimizations of Theorem 1.2

In this section our goal is to optimize the time overhead of the derandomization in Theorem 1.2. First, we improve the original deterministic time bound $n \cdot T(n)^{1+\epsilon}$ to the time bound $n^{1+\epsilon} \cdot T(n)$. As mentioned in Section 4.3, for probabilistic polynomial-time algorithms Theorem 4.8 already asserts such a time bound, and therefore it suffices to show such an improvement for superpolynomial-time algorithms.

We will do so for probabilistic algorithms that run in *at most sub-exponential time*, under the stronger hypothesis that there exists a one-way function that is secure against sub-exponential sized circuits. Let us first set up one piece of notation: We say that a function $T\colon \mathbb{N} \to \mathbb{N}$ is a valid γ'-$\mathsf{sub\text{-}exponential}$ function if T is time-constructible, increasing, satisfies $T(n) = 2^{n^{\gamma'}}$ and $T(n+1) \leq 2 \cdot T(n)$, and if $\lceil T^{-1} \rceil$ is time-constructible and satisfies $T^{-1}(\mathbb{N}) = \mathbb{N}$. Then, the result is the following:

[25]Note that such ℓ always exists since $\ell(T(m)) \in \big[(1+\epsilon)\cdot\log(T^{-1}(T(m))), (1+\epsilon)\cdot\log(\lceil T^{-1}(T(m))\rceil)\big]$.

Theorem 4.12 (optimizing the derandomization overhead for superpolynomial-time algorithms)**.** *There exists a universal constant $c > 1$ such that for every $\epsilon > 0$ there exist $\rho > 0$ and $\delta_1, \delta_2 > 0$ for which the following holds. Assume that for some $\gamma > 0$ there exist one-way functions that are secure against circuits of size 2^{n^γ}, and let $T\colon \mathbb{N} \to \mathbb{N}$ be a valid γ'-sub-exponential function, where γ' depends on γ. For $T'(n) = 2^{(c\cdot\delta_1)\cdot n} \cdot T(2^{(1-\delta_1)\cdot n})$, assume that there exists $L \in \texttt{amort-}\mathcal{DTIME}[2^{\delta_2\cdot n} \cdot T'(n)]$ such that $L \notin \texttt{i.o.}\mathcal{DTIME}[T']/2^{(1-\delta_1)\cdot n+1}$. Then, there exists a $(n^{-\rho})$-PRG for $\mathcal{DTIME}[O(n)]/T^{-1}$ with seed length $(1+\epsilon)\cdot\log(T^{-1}(n))$ that is batch-computable in time $T^{-1}(n)^{1+\epsilon}\cdot n$. Consequently, $pr\mathcal{BPTIME}[T] \subseteq pr\mathcal{DTIME}[n^{1+\epsilon}\cdot T(n)]$.*

Proof. The proof is analogous to the proof of Theorem 4.8, and we define the parameters $\epsilon' = \epsilon/3$ and δ_0, δ_1, and δ_2 in the exact same way. Relying on Proposition 4.5 and on our hypothesis, there exists a $(1/n)$-PRG that on input 1^n uses a seed of length $\log(n)^{c'}$ for some constant c' and is computable in time $\tilde{O}(n)$. We define $A(n) = \lceil T^{-1}(n) \rceil$ and $A^{-1}(n) = T(n-1)+1$, and note that $A^{-1}(n) = \min\{N : A(N) = n\}$, that $A^{-1}(n) \leq T(n)$, and that $A(n) \leq \log(n)^{c'}$ if the constant γ' is sufficiently small.

Let L be the hard problem from our hypothesis, and note that the truth-table of L_ℓ can be printed in time

$$\begin{aligned} 2^{(1+c\cdot\delta_1+\delta_2/2)\cdot\ell} \cdot T(2^{(1-\delta_1)\cdot\ell}) &= 2^{(1+c\cdot\delta_1+\delta_2/2)\cdot\ell} \cdot \Big(A^{-1}(2^{(1-\delta_1)\cdot\ell}+1)+1\Big) \\ &\leq 2^{(1+c\cdot\delta_1+\delta_2/2)\cdot\ell} \cdot \Big(2\cdot A^{-1}(2^{(1-\delta_1)\cdot\ell})+3\Big) \\ &< 2^{(1+c\cdot\delta_1+\delta_2)\cdot\ell} \cdot A^{-1}(2^{(1-\delta_1)\cdot\ell})\,, \end{aligned}$$

whereas no algorithm that uses at most $2^{(1-\delta_1)\cdot\ell+1}$ bits of non-uniform advice can decide L_ℓ in time $2^{(c\cdot\delta_1)\cdot n}\cdot T(2^{(1-\delta_1)\cdot n}) \geq 2^{(c\cdot\delta_1)\cdot n}\cdot A^{-1}(2^{(1-\delta_1)\cdot n})$.

Using Proposition 4.6 with the parameter value ϵ', there exists an $(n^{-\delta_0/2})$-PRG for $\mathcal{DTIME}[O(n)]/A$ with seed length $\ell(n) = (1+\epsilon')\cdot\log(A(n)) < (1+2\epsilon')\cdot\log(T^{-1}(n))$ that is batch-computable in time $W(n) = A(n)^{1+\epsilon'}\cdot n < T^{-1}(n)^{1+2\epsilon'}\cdot n$. Proposition 4.7 then implies that $pr\mathcal{BPTIME}[T] \subseteq pr\mathcal{DTIME}[O(n^{1+2\epsilon'})\cdot T(n)] \subset pr\mathcal{DTIME}[n^{1+\epsilon}\cdot T(n)]$. ∎

Finally, we can further reduce the derandomization overhead for derandomization that works in *average-case*. We will derandomize time-T algorithms in time $n^\epsilon\cdot T(n)$ with respect to all distributions that are samplable in time $T(n)$, under a hypothesis similar to that of Theorem 4.12. Moreover, for every $L \in \mathcal{BPTIME}[T]$ we construct a *single deterministic algorithm* that works for all distributions that are samplable in time T. (The result extends naturally to promise-problems, and we explain this after the proof.)

Theorem 4.13 (average-case derandomization with almost no overhead)**.** *There exists a universal constant $c > 1$ such that for every $\epsilon > 0$ there exist $\delta_1, \delta_2 > 0$ for which the following holds. Let $T(n) = n^k$ for a constant $k \geq 1$. Assume that there exist one-way functions secure against polynomial-sized circuits, and that for $T_k(n) = 2^{(c\cdot\delta_1)\cdot n}\cdot 2^{(1-\delta_1)\cdot(2k/\epsilon)\cdot n}$ there exists $L_0 \in \texttt{amort-}\mathcal{DTIME}[2^{\delta_2\cdot n}\cdot T_k(n)]$ such that $L_0 \notin \texttt{i.o.}\mathcal{DTIME}[T_k]/2^{(1-\delta_1)\cdot n+1}$. Then, for every $L \in \mathcal{BPTIME}[T]$ there exists an algorithm A that runs in time $n^\epsilon\cdot T(n)$ such that for every distribution that can be sampled in time $T(n)$ it holds that $\Pr_{x\sim\mathcal{D}}[A(x) = L(x)] = 1-\texttt{neg}(n)$, where $\texttt{neg}$ is a negligible function.*

Proof. Let $L \in \mathcal{BPTIME}[T]$, let M be a probabilistic time-T machine that decides L, and let S be a probabilistic algorithm that gets input 1^n, runs in time $T(n)$, and outputs an n-bit string.

By our hypothesis and the "moreover" part of Proposition 4.3, for some negligible function $\mathtt{neg}$, there exists a $\mathtt{neg}$-PRG $G^{\mathtt{cry}}$ for circuits of size $\mathrm{poly}(n)$ that on input $1^{T(n)}$ uses a seed of length at most $n^{\epsilon'}$ and is computable in time $T(n) \cdot n^{\epsilon'}$, where $\epsilon' = \epsilon/2$. Let $\tilde{S}$ be an algorithm that gets input 1^n and $n^{\epsilon'}$ bits of randomness, maps its randomness to a string of length $T(n)$ using $G^{\mathtt{cry}}$, and applies S to the latter string; that is, $\tilde{S}(1^n, w) = S(1^n, G^{\mathtt{cry}}(1^{T(n)}, w))$. Let $\tilde{M}$ be a machine that gets input $w \in \{0,1\}^m$, maps it to a string $x = \tilde{S}(1^n, w)$ of length $n = m^{1/\epsilon'}$, and outputs $M(x)$. Observe that on m-bit inputs $\tilde{M}$ runs in time $O(m \cdot T(m^{1/\epsilon'})) < T(m^{2/\epsilon'})$, and that on each input $\tilde{M}$ either outputs 0 with probability at least $2/3$ or outputs 1 with probability at least $1/3$.

Relying on Proposition 4.6 with the parameter ϵ' and the function $A(n) = \left\lceil n^{\epsilon'/2k} \right\rceil$, and assuming that $\delta_1, \delta_2 > 0$ are sufficiently small, there exists a $(n^{-\Omega(1)})$-PRG for $\mathcal{DTIME}[O(n)]/A$, denoted G, that on inputs of length n uses a seed of length less than $\ell(n) = (3\epsilon'/k) \cdot \log(n)$ and is batch-computable in time less than $n^{1+3\epsilon'/k}$. By a proof analogous to the proof of Proposition 4.7, for every sufficiently large $m \in \mathbb{N}$ and $w \in \{0,1\}^m$, the random coins of $\tilde{M}$ can be replaced by the distribution $G(1^N, \mathbf{u}_{\ell(N)})$, where $N = 1^{T(m^{1/\epsilon'})}$, while changing the acceptance probability by at most $1/8$.[26]

Now, observe that the random coins of $\tilde{M}$ on input $w \in \{0,1\}^m$ are only used as random coins for M on input $x = \tilde{S}(1^{m^{1/\epsilon'}}, w)$. Therefore, for every sufficiently large $n \in \mathbb{N}$ and $x \in \{0,1\}^n$ such that $x = \tilde{S}(1^n, w)$ for some $w \in \{0,1\}^{n^{\epsilon'}}$,[27] the random coins of M on input x can be replaced by the distribution $G(1^N, \mathbf{u}_{\ell(N)})$, where $N = T(n)$, while changing the acceptance probability by at most $1/8$.

While the latter claim was stated only for $x = \tilde{S}(1^n, w)$, we now show that this claim actually holds for almost all x's in the distribution $S(1^n, \mathbf{u}_{T(n)})$. That is:

Claim 4.13.1. *With probability at least $1 - \mathtt{neg}(T(n))$ over choice of $x \sim S(1^n, \mathbf{u}_{T(n)})$, the random coins of M can be replaced by the distribution $G(1^N, \mathbf{u}_{\ell(N)})$ while changing the acceptance probability by less than $1/6$.*

Proof. For every $x \in \{0,1\}^n$, denote $\nu(x) = \Pr[M(x, \mathbf{u}_N) = 1]$ and denote $\tilde{\nu}(x) = \Pr[M(x, G(1^N, \mathbf{u}_{\ell(N)})) = 1]$. Let $\mathtt{YES} \subseteq \{0,1\}^{T(n)}$ be the set of z's such that $x = S(1^n, z)$ satisfies $|\nu(x) - \tilde{\nu}(x)| \leq 1/8$, and let $\mathtt{NO} \subseteq \{0,1\}^{T(n)}$ be the set of z's such that $x = S(1^n, z)$ satisfies $|\nu(x) - \tilde{\nu}(x)| > 1/6$. Note that a probabilistic algorithm D can solve the promise-problem $(\mathtt{YES}, \mathtt{NO})$ in time $O(T(n))$,[28], and therefore there exists a (deterministic) circuit $D' \colon \{0,1\}^{T(n)} \to \{0,1\}$ of size $O(T(n)^2)$ that solves $(\mathtt{YES}, \mathtt{NO})$ (i.e., D' is obtained by hard-wiring fixed $O(T(n))$ random strings into D).

[26]To see this, assume that there exist infinitely many $m \in \mathbb{N}$ such that for some $w \in \{0,1\}^m$ it holds that $\tilde{M}(w)$ is a $(1/8)$-distinguisher for $G(1^N, \mathbf{u}_{\ell(N)})$. We define an algorithm D that gets input $x \in \{0,1\}^n$, if $n \notin \left\{T(m^{1/\epsilon'}) : m \in \mathbb{N}\right\}$ accepts, and otherwise given advice a_m computes $M(a_m, x)$ (i.e., x is used as randomness). The algorithm D runs in linear time, uses $T^{-1}(n)^{\epsilon'}$ bits of advice, and (by our assumption) $(1/8)$-distinguishes the output of G from uniform infinitely-often. This is a contradiction.

[27]We ignore rounding issues for simplicity. The more accurate statement here would be that x is the n-bit prefix of $\tilde{S}(1^n, w)$ for some $w \in \{0,1\}^{\lceil n^{\epsilon'} \rceil}$.

[28]Specifically, D that gets input $z \in \{0,1\}^{T(n)}$, maps it to $x = S(1^n, z)$, estimates both $\nu(x)$ and $\tilde{\nu}(x)$ up to accuracy 0.01 with confidence $2/3$, and accepts if and only if $|\nu(x) - \tilde{\nu}(x)| < 1/7$.

We already proved that when $z = G^{\mathtt{cry}}(1^{T(n)}, w)$ for some $w \in \{0,1\}^{n^{\epsilon'}}$, we have that $z \in \mathtt{YES}$; hence, in this case $D'(z) = 1$. Assume towards a contradiction that with probability more than $\mathtt{neg}(T(n))$ over choice of $x = S(1^n, \mathbf{u}_{T(n)})$ it holds that $|\nu(x) - \tilde{\nu}(x)| > 1/6$. Then, with probability more than $\mathtt{neg}(T(n))$ over a uniform choice of $z \in \{0,1\}^{T(n)}$ it holds that $z \in \mathtt{NO}$, in which case $D'(z) = 0$. Since $G^{\mathtt{cry}}$ is a $\mathtt{neg}$-PRG for circuits of arbitrary polynomial size, this is a contradiction. □

Our deterministic algorithm gets input $x \in \{0,1\}^n$, computes the output-set of G, denoted $R = \left\{G(1^N, w) : w \in \{0,1\}^{\ell(N)}\right\}$, computes $M(x, r)$ for each $r \in R$, and outputs the majority value. Recalling that $\ell(N) = (3\epsilon'/k) \cdot \log(N)$, this algorithm runs in time $O(n^{3\epsilon'} \cdot T(n))$, and by Claim 4.13.1, with probability at least $1 - \mathtt{neg}(N)$ over choice of $x \sim S(1^n, \mathbf{u}_{T(n)})$ this algorithm outputs $L(x)$. Finally, observe that this algorithm does not depend on the sampling algorithm S, and therefore we obtained a single deterministic algorithm that works for all distributions samplable in time $T(n)$. ■

Remark: Extension to promise-problems. The argument above extends to promise-problems (i.e., to $pr\mathcal{BPTIME}[T]$ rather than only $\mathcal{BPTIME}[T]$), under the additional natural hypothesis that the probability that $\mathcal{D}_n$ violates the promise is at most $\mathtt{neg}(n)$.

To see this, note that the only place in the proof above where we relied on the fact that M decides a language $L \subseteq \{0,1\}^*$ was in the last paragraph. Specifically, in the first part of the proof we showed that with probability at least $1 - \mathtt{neg}(n)$ over choice of $x \sim S(1^n, \mathbf{u}_{T(n)})$, we can replace the random coins of $M(x, \cdot)$ by pseudorandom coins while changing the acceptance probability by less than $1/6$. Now, relying on the hypothesis that for *every* input $x \in \{0,1\}^*$ the acceptance probability of $M(x, \cdot)$ is either at least $2/3$ or at most $1/3$, we deduced that the machine obtained by using pseudorandom coins still accepts every $x \in L$ and rejects every $x \notin L$.

Now, if we replace L by a promise-problem $\Pi = (\mathtt{YES}, \mathtt{NO})$ and assume that with probability at least $1 - \mathtt{neg}(n)$ over $x \sim S(1^n, \mathbf{u}_{T(n)})$ it holds that x does not violate the promise, then with probability at least $1 - \mathtt{neg}(n)$ it holds that replacing the random coins by pseudorandom ones changes the acceptance probability by less than $1/6$ *and* that the acceptance probability of $M(x, \cdot)$ is either at least $2/3$ or at most $1/3$. In this case, the deterministic machine outputs the correct decision at x.

5 Fast derandomization via a simple paradigm

In this section we present our alternative proof of Theorem 1.1, as well as prove Theorem 1.7 and our conditional near-optimal quantified derandomization. In Section 5.1 we first present our construction of a reconstructive PRG for quantified derandomization, which works when given access to a function that is hard for SVN circuits. In Section 5.2 we describe how our proofs follow from this construction, in high-level. Then, in Section 5.3 prove our results regarding quantified derandomization, and in Section 5.4 we prove Theorems 1.1 and 1.7. Throughout this section we ignore rounding issues for simplicity (this does not meaningfully affect our proofs).

5.1 A reconstructive PRG for quantified derandomization

Recall that standard PRGs are pseudorandom for *distinguishers*, which in our setting are modeled as non-uniform circuits. We now wish to construct a PRG for *quantified derandomization*, or in other words for the special case of distinguishers that are extremely biased. To do so we first define such distinguishers, as follows:

Definition 5.1 (quantified distinguisher). *Let* $\mathbf{w}$ *be a distribution over* $\{0,1\}^N$ *and let* $B = B(N)$*. We say that a circuit* $D\colon \{0,1\}^N \to \{0,1\}$ *is a* B-quantified ρ-distinguisher for $\mathbf{w}$ *if for some* $\sigma \in \{0,1\}$ *it holds that* $\Big| \{x : D(x) = \sigma\} \Big| \leq B$ *and* $\Pr[D(\mathbf{w}) = \sigma] \geq \rho$.

Our PRG will be reconstructive. Its reconstruction procedure will be modeled as a non-deterministic oracle machine R that gets oracle access to a quantified distinguisher D as well as a bounded number of non-uniform advice bits, and is able to compute the function non-deterministically and unambiguously (see Definition 3.3). In more detail:

Proposition 5.2 (a reconstructive PRG for quantified derandomization). *For every* $\alpha > 0$ *there exists* $\mu > 0$ *such that for every two constants* $\beta, \gamma > 0$ *the following holds. There exists an oracle machine* G *that, when given input* 1^N *and a random seed of length* $\ell = (\alpha + \beta) \cdot n$ *where* $n = \log(N)$ *and oracle access to* $f\colon \{0,1\}^{(1+\beta)\cdot n} \to \{0,1\}$*, satisfies the following:*

1. *The machine* G *runs in time* $\tilde{O}(N^{1+\alpha+\beta})$ *and outputs* N *bits.*

2. *There exists an oracle machine* R *that, when given oracle access to a* $2^{N^{1-\gamma}}$*-biased* $(N^{-\mu})$*-distinguisher* D *for* $G^f(1^N, \mathbf{u}_\ell)$*, computes* f *non-deterministically and unambiguously in time* $N^{1+\alpha}$*, using* N^α *oracle queries to* D *and* $O(N + N^{1+\beta+(2\alpha-\gamma)})$ *bits of advice.*

Proof. We identify f with its truth-table $f \in \{0,1\}^{N^{1+\beta}}$.

The generator G. For sufficiently small constants $\eta > \mu > 0$, let $\bar{f} = Enc(f)$ where Enc is the code from Corollary 3.20, instantiated with $m = |f|$ and $\rho = N^{-\mu}$ and the constant $\eta > 0$; note that $\bar{f} \in \{0,1\}^{\bar{N}\cdot\log(|\Sigma|)}$, where $\bar{N} = O\Big(|f| \cdot N^{2\mu/\eta'}\Big) < N^{1+\beta+\alpha}$ (relying on a sufficiently small choice of μ) and $|\Sigma| = O\Big(|f|^{\eta'} \cdot N^{2\mu}\Big) < N^{(1+\beta)\cdot\eta}$.

For every $s \in [|\bar{f}|/N]$, let $\bar{f}_s \in \{0,1\}^N$ be the s^{th} consecutive substring of length N of $\bar{f}$ (i.e., for $i \in [n]$ it holds that the i^{th} bit of $\bar{f}_s$ equals $\bar{f}_{(s-1)\cdot N+i}$). The machine G gets $s \in [|\bar{f}|/N]$ as a seed and outputs $\bar{f}_s$. Note that $|\bar{f}|/N < N^{\beta+\alpha}$, and therefore $|s| < (\beta+\alpha)\cdot\log(N)$. Also note that the running-time of G is dominated by the time it takes to compute $\bar{f}$, which is $\tilde{O}(N^{1+\beta+\alpha})$.

Reconstructing a corrupt version of $\bar{f}$. For $\rho = N^{-\mu}$, let $D\colon \{0,1\}^N \to \{0,1\}$ be a $2^{N^{1-\gamma}}$-quantified ρ-distinguisher for G^f, let $\sigma \in \{0,1\}$ be the rare output of D, and denote $S_\sigma = D^{-1}(\sigma)$. We first describe an algorithm $\tilde{R}^D$ that computes a function $\tilde{f} \in \Sigma^{\bar{N}}$ that agrees with $\bar{f} \in \Sigma^{\bar{N}}$ on at least a ρ-fraction of their inputs. To do so, let $\mathcal{H} \subseteq \Big\{\{0,1\}^N \to \{0,1\}^{N^{1-\gamma+\alpha}}\Big\}$ be the family of quasilinear-time computable functions from Theorem 3.12 (instantiated with parameters $m = N$ and $m' = m^{\gamma-\alpha}$). For every fixed $s \in [N^{2\beta}]$ we have that

$$\Pr_{h\in\mathcal{H}}\left[\exists g \in S_\sigma : h(\bar{f}_s) = h(g)\right] \leq |S_\sigma| \cdot 2^{-N^{1-\gamma+\alpha}} \leq 2^{-N^{1-\gamma+\alpha}+N^{1-\gamma}},$$

where the last inequality relied on the fact that $|S_\sigma| \leq 2^{N^{1-\gamma}}$. By a union-bound, there exists some $h \in \mathcal{H}$ such that for every $s \in [N^{\beta+\alpha}]$ there does not exist $g \in S_\sigma \setminus \{\bar{f}_s\}$ for which $h(\bar{f}_s) = h(g)$. Let us now fix such an h.

The algorithm $\tilde{R}$ gets as advice the bit σ, the description of h, and all the $N^{\beta+\alpha}$ strings $\{h(\bar{f}_s) : s \in [N^{\beta+\alpha}]\}$, which constitute $O(N + N^{1+\beta+2\alpha-\gamma})$ bits of advice. Given input $x \in [\bar{N}]$, the algorithm:

1. Non-deterministically guesses $g \in \{0,1\}^N$.

2. Queries D on input g, and outputs $\perp$ unless $D(g) = \sigma$.

3. For the appropriate s (such that x indexes a location in $\bar{f}_s$), the algorithm verifies that $h(g) = h(\bar{f}_s)$, and otherwise outputs $\perp$.

4. Outputs the symbol of g that appears in the location indexed by x.

Note that the running-time of $\tilde{R}$ is dominated by the computation of h, and is thus bounded by $\tilde{O}(N)$. We now claim that there exists a set $T \subseteq [\bar{N}]$ of density at least ρ such that for every $x \in T$ it holds that $\tilde{R}(x)$ non-deterministically computes $\bar{f}(x)$.[29]

To see this, let $S = \{s \in [N^{\beta+\alpha}] : D(\bar{f}_s) \neq \sigma\}$, and recall that (by the properties of D) we have that $|S|/N^{\beta+\alpha} \geq \rho$. By the properties of h, for every $s \in S$ there does not exist $g \in D^{-1}(\neg\sigma) \setminus \{\bar{f}_s\}$ such that $h(g) = h(\bar{f}_s)$. Hence, by the construction of $\tilde{R}$ above, for every x that indexes a location in $\bar{f}_s$ for some $s \in S$ we have that $\tilde{R}(x)$ non-deterministically computes $\bar{f}(x)$.[30] We define the set $T \subseteq [\bar{N}]$ to consist of all x that index locations in $\bar{f}_s$ for some $s \in S$, and indeed we have that $|T|/\bar{N} \geq \rho$.

Reconstructing f. We now run the local list-decoding algorithm for f from Corollary 3.20, denoted *Dec*, while giving it oracle access to $\tilde{R}$. The underlying oracle machine runs in time $N^\eta \cdot (1/\rho)^{1/\eta'} < N^{\alpha/2}$ (relying on a sufficiently small choice of $\eta, \mu > 0$) and uses $\log(O(1/\rho)) < \log(N)$ bits of non-uniform advice.

Answering the queries to $\tilde{R}$ in the latter oracle machine with the actual oracle machine for $\tilde{R}$, we obtain a randomized procedure that non-deterministically computes f in time $N^{\alpha/2} \cdot \tilde{O}(N) = \tilde{O}(N^{1+\alpha/2})$ that makes at most N^α oracle queries to D and uses

$$O(N + N^{1+\beta+2\alpha-\gamma} + \log(N)) = O(N + N^{1+\beta+2\alpha-\gamma})$$

bits of non-uniform advice (that depends on D and on f). Using naive error-reduction and fixing an appropriate random string, we obtain an algorithm R that computes f non-deterministically, unambiguously and without randomness. The running-time of R is $\tilde{O}(N^{1+\alpha/2}) < N^{1+\alpha}$ and the number of advice bits is still $O(N + N^{1+\beta+2\alpha-\gamma})$. ■

Remark: Comparison to Kolmogorov-based derandomization. The proof of Proposition 5.2 can be viewed as a "constructive" (i.e., efficient) variant of the classical technique of derandomizing probabilistic algorithms using *strings with high Kolmogorov complexity* (see, e.g., [LV08, Thm 7.3.5]). To see this, recall that for any $\bar{D}\colon \{0,1\}^{\bar{N}} \to$ of size $\bar{N}^{.99}$

[29]By "non-deterministically computes $f(x)$" we mean that for some non-deterministic choices $\tilde{R}(x) = f(x)$, whereas for all non-deterministic choices $\tilde{R}(x) \in \{\bar{f}(x), \perp\}$.

[30]For simplicity, we ignore rounding issues at this point, and assume that blocks of size N in $\bar{f}$ do not truncate blocks of size $\log(|\Sigma|) < (1+\beta) \cdot \eta \cdot \log(N)$.

that accepts all but $2^{\bar{N}^{.99}}$ of its inputs, and any string $f \in \{0,1\}^{\bar{N}}$ with maximal Kolmogorov complexity $\bar{N}$, we have that $\bar{D}(f) = 1$. This is the case since otherwise the string f would have a description of length $O(\bar{N}^{.99})$, consisting of the circuit $\bar{D}$ along with the index $i \in \{0,1\}^{.99 \cdot \bar{N}}$ of f inside the set $\bar{D}^{-1}(0)$. In fact, the argument works even when f only has high *time-bounded Kolmogorov complexity*.[31]

In the reduction in our argument, instead of only showing that f has small time-bounded Kolmogorov complexity, we show that f has small circuit complexity; in other words, we show that f can be computed in a "strongly-explicit" manner by an SVN circuit that gets as input an index $x \in [\log(|f|)]$ and outputs f_x. We note that while the analysis of this argument relies only on classical notions such as hash functions and error-reduction, the instantiation above uses technical constructions that are more recent than in Sipser's time, such as very efficient samplers and hash functions.

5.2 High-level description of the proofs

Let us now describe the proofs of our results in the current section, in an informal and high-level way. Our goal is to derandomize algorithms that run in time $N = T(n)$, and to do so we will use a PRG that "fools" circuits $D\colon \{0,1\}^N \to \{0,1\}$ of size N. Similarly to Section 2, we denote by $\epsilon > 0$ a very small constant, and construct ϵ-PRGs (the extension to $n^{-.01}$-PRGs is straightforward).

The proof of our near-optimal quantified derandomization follows easily from Proposition 5.2. Specifically, for this proof we only need to consider distinguishers that evaluate to some $\sigma \in \{0,1\}$ on all but $2^{N^{1-\epsilon}}$ of their inputs, and we instantiate the PRG from Proposition 5.2 with a function f of truth-table size $|f| = N^{1+\epsilon}$ that is hard for SVN circuits of size $O(N^{1+\epsilon/2})$. Our algorithm for quantified derandomization first computes the truth-table of f, which can be done in time $N^{1+O(\epsilon)}$ by our hypothesis, and then encodes it using a locally-list-decodable code that is computable in near-linear time $\tilde{O}(N^{1+\epsilon})$; then it enumerates over the N-bit consecutive parts of the encoding of f (which constitute the output-set of the PRG), and outputs the majority of the evaluations of D on these parts. The running-time of this algorithm is at most $N^{1+O(\epsilon)}$.

To prove Theorem 1.7 we need to consider *all* distinguishers $D\colon \{0,1\}^N \to \{0,1\}$ of size N, rather than only very biased distinguishers. Let $\mathsf{Samp}\colon \{0,1\}^{\bar{N}} \times \{0,1\}^{(1+O(\epsilon))\cdot\log(\bar{N})} \to \{0,1\}^N$ be a linear-time-computable averaging sampler with accuracy $1/10$ and confidence $2^{\bar{N}^{1-\epsilon}-\bar{N}}$, where $\bar{N} = N^{1+O(\epsilon)}$.[32] For any potential distinguisher D, denote the (actual) acceptance probability of D by $\mu = \Pr_r[D(r) = 1]$, and define $\bar{D}\colon \{0,1\}^{\bar{N}} \to \{0,1\}$ such that $\bar{D}(z) = 1$ if and only if $\Pr_r\left[\mathsf{Samp}(z,r) \in D^{-1}(1)\right] \in \mu \pm (1/10)$ (where the $1/10$ term corresponds to the error of the sampler). We stress that *our algorithm does not actually construct* $\bar{D}$ (i.e., $\bar{D}$ is a mental experiment for the analysis), and therefore there is no problem to "hard-wire" μ into $\bar{D}$. Note that $\bar{D}$ is of size $N^{2+O(\epsilon)}$ and accepts all but $2^{\bar{N}^{1-\epsilon}}$ of its $\bar{N}$-bit inputs. By a standard analysis, any .01-PRG G_0 for $\bar{D}$ yields a $(1/9)$-PRG $G(s,r) = \mathsf{Samp}(G_0(s),r)$ for D. [33] Therefore, we just need a .01-PRG for $\bar{D}$.

[31]Recall that the time-bounded Kolmogorov complexity of $f \in \{0,1\}^*$, defined by Levin, is the minimum over $\langle M\rangle + \log(t)$ such that $\langle M\rangle$ is the description of a machine that prints f in time t.

[32]As noted in [DMO+20], such a construction follows by re-analyzing a well-known construction of [TSZS06, Thm 5] for the min-entropy value $\bar{N}^{1-\epsilon}$; see Theorem 3.17 for a formal statement.

[33]To see this, call a string z good if $\Pr_r\left[\mathsf{Samp}(z,r) \in D^{-1}(1)\right] \in \mu \pm (1/10)$. Then, for any $\sigma \in \{0,1\}$ it holds that $\Pr_{s,r}[\mathsf{Samp}(G(s),r) \in D^{-1}(\sigma)] \le \Pr[G(s) \text{ is not good}] + \mu + .01 < \mu + 1/9$.

We then use the PRG from Proposition 5.2 with parameters as follows. We fix a function f of truth-table size $N^{2+O(\epsilon)}$, and assume that it is hard for SVN circuits of size $|f|^{1-\epsilon}$. The derandomization algorithm computes the truth-table of f, encodes it with the error-correcting code to obtain a codeword $\bar{f}$, and outputs the majority of the evaluations of D on the set $\left\{\mathsf{Samp}(\bar{f}_s, r)\right\}_{s,r}$. This set is of size $N^{2+O(\epsilon)}$ (since each of the sets of values for s and for r is of size $N^{1+O(\epsilon)}$), evaluating D on the set takes time $N^{3+O(\epsilon)}$. Thus, we obtain derandomization in either cubic time or quadratic time, depending on the time that it takes to compute the truth-table of f (this corresponds to the two different result statements in Theorem 1.7).

Finally, to prove Theorem 1.1, we want to reduce the running time in the proof of Theorem 1.7 from cubic (or quartic) to quadratic. The main bottleneck is that the circuit $\bar{D}$ is of size more than N^2. [34] To decrease the size of $\bar{D}$, recall that in the context of Theorem 1.1 we assume hardness for *randomized* non-deterministic circuits, and therefore we are allowed to use randomness in the definition of $\bar{D}$ (i.e., the reconstruction procedure from Proposition 5.2 will transform a randomized circuit $\bar{D}$ into a randomized SVN circuit that computes the hard function).

Thus, we define $\bar{D}$ in a way that utilizes this randomness in order to perform error-reduction more efficiently. Specifically, instead of defining $\bar{D}$ such that it computes $\nu(z) = \Pr_r[D(\mathsf{Samp}(z,r)) = 1]$ exactly, the circuit $\bar{D}$ estimates $\nu(z)$ up to error .01 using random sampling of r's, and accepts if and only if its estimate $\tilde{\nu}(z)$ is in the interval $\mu \pm .01$. This handles the main bottleneck in the proof, since the resulting circuit $\bar{D}$ is now of size $N^{1+O(\epsilon)}$, and therefore we can "fool" it using a hard function f whose truth-table is only of size $N^{1+O(\epsilon)}$. Overall, since f is computable in exponential time (in its input length), we can compute its truth-table in time $N^{2+O(\epsilon)}$, encode it using the code (in time $N^{1+O(\epsilon)}$), and output the majority of evaluations of D on the set $\left\{\mathsf{Samp}(\bar{f}_s, r)\right\}_{r,s}$, which is now of size $N^{1+O(\epsilon)}$ (since $\bar{f}$ is of size $N^{1+O(\epsilon)}$ and hence there are at most $N^{O(\epsilon)}$ values for s). This yields derandomization with quadratic overhead.

5.3 Near-optimal quantified derandomization

Relying on Proposition 5.2, we now present two very efficient solutions for the quantified derandomization problem: The first is an unconditional construction of a *non-uniform* circuit family (i.e., a non-explicit PRG), and the second is a conditional construction of an algorithm that solves the problem, where the hypothesis refers to the existence of a sufficiently hard "batch-computable" function (see below). Specifically, we first show that, unconditionally, there exist a non-uniform PRG for quantified derandomization of circuits with at most $2^{N^{1-\gamma}}$ exceptional inputs that has seed length $\gamma \cdot \log(N)$.

Theorem 5.3 (non-explicit PRG with short seed)**.** *For every $\gamma > 0$ there exist $\mu > 0$ such that the following holds. For every $N \in \mathbb{N}$ there exist N^γ strings $w_1, ..., w_{N^\gamma} \in \{0,1\}^N$ such that for every circuit $D\colon \{0,1\}^N \to \{0,1\}$ of size N that evaluates to some $\sigma \in \{0,1\}$ on all but $2^{N^{1-\gamma}}$ it holds that $\Pr_{i\in[N^\gamma]}\left[D(w_i) = \sigma\right] \geq 1 - N^{-\mu}$.*

[34]To see why this is a problem, observe that "fooling" $\bar{D}$ using our approach requires a function whose truth-table is of size more than $|\bar{D}| \geq N^2$. Thus, the PRG G_0 will have seed length at least $\log(N)$, and the final PRG (i.e., G_0 composed with Samp) will have seed length at least $2 \cdot \log(N)$. Evaluating D at each of the N^2 outputs will take time at least N^3.

Proof. Let $\alpha = \gamma/4$ and $\beta = 3\gamma/4$. By a standard counting argument, there exists a function f whose truth-table is of size $N^{1+\beta}$ that cannot be computed by SVN circuits of size $N^{1+\beta-\gamma/2}$. The strings $w_1, ..., w_{N^\gamma}$ will be the output-set of the machine G from Proposition 5.2, when the latter is instantiated with f as the hard function and with parameters $\alpha = \gamma/4$ and β as above. Note that the output-set of G is indeed of size $N^{(\alpha+\beta)} = N^\gamma$. The claim follows by noting that there does not exist a $2^{N^{1-\gamma}}$-biased $(N^{-\mu})$-distinguisher for G, otherwise f could be computed by an SVN circuit of size $O\left(N^{1+\beta+(2\alpha-\gamma)} + N^{1+\alpha}\right) = O(N^{1+\beta-\gamma/2})$. ■

Our second result asserts that if there exists a function whose entire truth-table on n-bit inputs can be printed in time $2^{(1.01)\cdot n}$, but that cannot be computed (on an input-by-input basis) by SVN circuits of size $2^{.99\cdot n}$, then we can solve the quantified derandomization problem with near-linear time overhead.

Theorem 5.4 (near-optimal quantified derandomization)**.** *For every $\gamma > 0$ there exists $\mu > 0$ such that for every $\delta > 0$ the following holds. Assume that there exists $L \in \mathcal{DTIME}[2^n]$ such that for every $n \in \mathbb{N}$ it holds that the truth-table of L can be printed in time $2^{(1+\delta)\cdot n}$, but L cannot be computed by SVN circuits of size $2^{(1-\gamma/4)\cdot n}$. Then, there exists an $(N^{-\mu})$-PRG with seed length $\gamma \cdot \log(N)$ for the class of circuits of linear size that evaluate to some $\sigma \in \{0,1\}$ on all but $2^{N^{1-\gamma}}$ of their inputs such that the entire output-set of the PRG can be printed in time $N^{(1+\gamma)\cdot(1+\delta)}$.*

The conclusion of Theorem 5.4 implies that randomized time-T algorithms that err on at most $2^{T(n)^{1-\gamma}}$ of their random choices can be deterministically simulated in time $O\left(T(n)^{(1+\gamma)\cdot(1+\delta)} + T(n)^{1+\gamma}\right) < T(n)^{1+2\gamma+\delta}$. (This follows by the standard reduction of $pr\mathcal{BPP}$ to the circuit acceptance probability problem; see, e.g., [Vad12, Corollary 2.31] or [Gol08, Exercise 6.14].)

Proof of Theorem 5.4. Given $D\colon \{0,1\}^N \to \{0,1\}$, let $\alpha = \gamma/4$, let $\beta = 3\gamma/4$, and let f be a function as in our hypothesis whose truth-table is of size $N^{1+\beta}$. We instantiate the PRG from Proposition 5.2 with these parameters and with f as the hard function, and claim that D evaluates to σ over $1 - N^{-\mu}$ of the outputs of the PRG (where μ is as in Proposition 5.2). This holds since otherwise D is a $2^{N^{1-\gamma}}$-biased $(N^{-\mu})$-distinguisher for the PRG, and hence f can be computed by an SVN circuit of size

$$O\left(N^{1+\alpha} + N^{1+\beta+(2\alpha-\gamma)}\right) = O\left(N^{1+\beta-\gamma/2}\right) < N^{(1+\beta)\cdot(1-\gamma/4)} ,$$

which is a contradiction.

To print the output-set of the PRG, we first construct the truth-table of f, which can be done in time $N^{(1+\beta)\cdot(1+\delta)} < N^{(1+\gamma)\cdot(1+\delta)}$. Then, for each of the N^γ seeds we can compute the corresponding N-bit output in time $\tilde{O}(N^{1+\alpha+\beta}) = \tilde{O}(N^{1+\gamma})$. ■

5.4 Standard derandomization: Proofs of Theorems 1.1 and 1.7

We now prove Theorems 1.1 and 1.7. Both results will first reduce the standard derandomization problem to a quantified derandomization problem, and then use the reconstructive PRG from Proposition 5.2 to solve the latter. The main difference between the proofs is a different reduction to quantified derandomization.

In the following result statements, the conclusions will be that there exists a PRG with seed length $c \cdot \log(N)$ whose entire output-set is computable in time $c' \cdot \log(N)$ for some small $c, c' \in \mathbb{N}$. To see how these imply the result statements in Section 1, recall that this allows us to solve CAPP in time $N^{c'} + N^{c+1}$ (by evaluating a given N-bit circuit of size N over the output-set of the PRG), which in turn implies that randomized algorithms that run in time $N = T(|x|)$ can be deterministically simulated in time $O\Big(T(|x|)^{\max\{c',c+1\}}\Big)$; that is, $pr\mathcal{BPTIME}[T] \subseteq pr\mathcal{DTIME}[T^{\max\{c',c+1\}}]$.

Theorem 5.5 (Theorem 1.7, restated)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that there exists $L \in \mathcal{DTIME}[2^n]$ such that for every $n \in \mathbb{N}$ it holds that L cannot be computed by SVN circuits of size $2^{(1-\delta)\cdot n}$, even infinitely-often. Then, there exists an $(N^{-\delta})$-PRG with seed length $(2+\epsilon)\cdot\log(N)$ whose entire output-set can be printed in time $N^{4+\epsilon}$. Moreover, if for every $n \in \mathbb{N}$ the entire truth-table of L on n-bit inputs can be printed in time $2^{(3/2)\cdot n}$, then the output-set of the PRG can be printed in time $N^{3+\epsilon}$.*

Proof. We first specify the parameters that we will use in the proof, and instantiate the hard function, the reconstructive PRG from Proposition 5.2 and the extractor from Theorem 3.17 with these parameters.

- For sufficiently small constants $\gamma, \delta > 0$ that depends on ϵ, let $\alpha = \gamma/4$ and let $\beta = 1 + 3c \cdot \gamma$.
- Let $\mathtt{Ext}\colon \{0,1\}^{\bar{N}} \times \{0,1\}^{(1+2c\cdot\gamma)\cdot\log(\bar{N})} \to \{0,1\}^N$ be the extractor from Theorem 3.17, instantiated with error $\frac{1}{2}\cdot N^{-\delta}$ for a sufficiently small $\delta > 0$, where $\bar{N} = N^{1+3\gamma} > (c \cdot N)^{1/(1-2\gamma)}$.
- Let $f \in \{0,1\}^{\bar{N}^{1+\beta}}$ be a function that can be computed in time $\bar{N}^{1+\beta}$ but cannot be computed by SVN circuits of size $\bar{N}^{1+\beta-\gamma/4} < \bar{N}^{1-\delta}$.
- Let G_0 be the reconstructive PRG from Proposition 5.2, instantiated for output length $\bar{N}$ with f as the hard function and with the parameters α, β, γ.

Our PRG is defined by $G(1^N, (s_0, s_1)) = \mathtt{Ext}(G_0(1^{\bar{N}}, s_0), s_1)$. Note that the seed length of G is $(1 + 2c\cdot\gamma + \alpha + \beta)\cdot\log(\bar{N}) < (2+\epsilon)\cdot\log(N)$, and that G can be computed (on a single seed) in time $\tilde{O}(N^{1+\alpha+\beta}) < N^{1+\epsilon}$. Moreover, we can print the output-set of G by first computing the truth-table of f in time $\bar{N}^{2\cdot(1+\beta)} = \bar{N}^{4+6c\cdot\gamma} < N^{4+\epsilon}$, and then for each of the $N^{2+\epsilon}$ seeds computing the corresponding output of G in time less than $N^{1+\epsilon}$. Thus, we can print the entire output-set of G in time $N^{4+\epsilon}$.

Fixing any circuit $D\colon \{0,1\}^N \to \{0,1\}$ of size N, we now want to show that G "fools" D with error $N^{-\delta}$. To do so we define $\bar{D}\colon \{0,1\}^{\bar{N}} \to \{0,1\}$ such that $\bar{D}(z) = 1$ if and only if $\Big|\Pr_{s_1}[D(\mathtt{Ext}(z,s_1) = 1)] - \Pr_r[D(r) = 1]\Big| \leq N^{-\delta}$. Note that $\bar{D}$ can be computed by a circuit of size $O\Big(2^{(1+2c\cdot\gamma)\cdot\log(\bar{N})}\cdot\bar{N}\Big) = O(\bar{N}^{2+2c\cdot\gamma})$, and that $\Pr_z[\bar{D}(z) = 0] \leq 2^{\bar{N}^{1-\gamma}-\bar{N}}$. It follows that $\Pr_{s_0}[G_0(1^{\bar{N}}, s_0) \in \bar{D}^{-1}(0)] \leq \frac{1}{2}\cdot N^{-\delta}$ (assuming $\delta > 0$ is sufficiently small), otherwise the combination of the oracle machine R and of the distinguisher $\bar{D}$ yields an SVN circuit that computes f of size

$$O\Big(\bar{N}^{1+\alpha} + \bar{N}^{\alpha}\cdot\bar{N}^{2+2c\cdot\gamma} + \bar{N}^{1+\beta+(2\alpha-\gamma)}\Big) < \bar{N}^{1+\beta-\gamma/4}\,. \tag{5.1}$$

Therefore, for every $\sigma \in \{0,1\}$ we have that

$$\begin{aligned}
&\Pr_{s_0,s_1}\left[D(G(1^N,(s_0,s_1))) = \sigma\right] \\
&\quad \le \Pr_{s_0}\left[G_0(1^{\bar{N}},s_0) \notin \bar{D}^{-1}(1)\right] + \Pr_{s_0,s_1}\left[\mathtt{Ext}(G_0(1^{\bar{N}},s_0),s_1) \in D^{-1}(\sigma) | G_0(1^{\bar{N}},s_0) \in \bar{D}^{-1}(1)\right] \\
&\quad \le \Pr_r[D(r) = \sigma] + N^{-\delta} ,
\end{aligned}$$

which means that G "fools" D with error $N^{-\delta}$.

The "moreover" part by noting that under the stronger hypothesis, the first step of computing the truth-table of f takes time $\bar{N}^{(3/2)\cdot(1+\beta)} < N^{3+\epsilon}$, and it still dominates the running-time of the entire algorithm that prints the output-set of the PRG. ■

Next, we show our alternative proof of Theorem 1.1. The high-level outline of this proof is similar to the proof of Theorem 5.5, but we will carry out the error-reduction (manifested in the circuit $\bar{D}$) in a more efficient way. In more detail, since we assume that the "hard" function cannot be computed by *randomized* SVN circuits, the existence of a randomized distinguisher would still contradict our hypothesis (i.e., we can use the reconstruction procedure with a randomized distinguisher); accordingly, we will use randomness to reduce the error of the distinguisher more efficiently (i.e., by a smaller circuit). Details follow.

Theorem 5.6 (Theorem 1.1, restated)**.** *For every $\epsilon > 0$ there exists $\delta > 0$ such that the following holds. Assume that there exists $L \in \mathcal{DTIME}[2^n]$ such that L cannot be computed by randomized SVN circuits of size $2^{(1-\delta)\cdot n}$, even infinitely-often. Then, there exists an $(N^{-\delta})$-PRG with seed length $(1+\epsilon)\cdot\log(N)$ whose entire output-set can be printed in time $N^{2+\epsilon}$.*

Proof. As in the proof of Theorem 5.5, we first specify the parameters and instantiate the hard function, the extractor, and the reconstructive PRG:

- For sufficiently small $\gamma, \delta > 0$, let $\alpha = \gamma/5$, let $\beta = 3\gamma$, and let $\gamma' = \gamma/2$.
- Let $\mathtt{Ext}\colon \{0,1\}^{\bar{N}} \times \{0,1\}^{(1+2c\cdot\gamma)\cdot\log(\bar{N})} \to \{0,1\}^N$ be the extractor from Theorem 3.17, instantiated with error $\frac{1}{6}\cdot N^{-\delta}$ and with $\bar{N} = N^{1+3\gamma}$.
- Let $f \in \{0,1\}^{\bar{N}^{1+\beta}}$ be a function that can be computed in time $\bar{N}^{1+\beta}$ but cannot be computed by randomized SVN circuits of size $\bar{N}^{1+\beta-\gamma/4}$.
- Let G_0 be the PRG from Proposition 5.2, instantiated for output length $\bar{N}$ with the function f and with parameters α, β, γ'.

Our PRG is defined by $G(s_0,s_1) = \mathtt{Ext}(G(s_0),s_1)$. This PRG has seed length $(1+2c\cdot\gamma+(\alpha+\beta))\cdot\log(\bar{N}) < (1+\epsilon)\cdot\log(N)$ and it can be computed in time less than $N^{1+\epsilon}$ given access to f. Printing the entire output-set of G can be done in time less than $N^{2+\epsilon}$, by first computing the truth-table of f in time $\bar{N}^{2\cdot(1+\beta)} < N^{2+\epsilon}$ and then evaluating G on each of the $N^{1+\epsilon}$ seeds.

Now fix a circuit $D\colon \{0,1\}^N \to \{0,1\}$ of size N. For any $z \in \{0,1\}^{\bar{N}}$, we denote by $\mu(z) = \Pr_{s_1}[D(\mathtt{Ext}(z,s_1)) = 1]$ the average value of D over the sample $\mathtt{Ext}(z,\cdot)$, and we also denote by $\nu = \Pr_r[D(r) = 1]$ the true average value of D. We will now define a circuit $\bar{D}\colon \{0,1\}^{\bar{N}} \to \{0,1\}$ of size $\tilde{O}(N^{1+2\delta})$ such that:

1. The circuit $\bar{D}$ accepts all but at most $2^{\bar{N}^{1-\gamma}}$ of its inputs.

2. For every $z \in \bar{D}^{-1}(1)$ it holds that $\left|\mu(z) - \nu\right| \leq \frac{1}{2} \cdot N^{-\delta}$.

3. The top gate of $\bar{D}$ is a CAPP gate: That is, a gate that takes as input a description of a circuit C', outputs 1 if $\Pr[C'(x) = 1] \geq 2/3$, outputs 0 if $\Pr[C'(x) = 1] \leq 1/3$, and otherwise outputs *some* bit (i.e., we do not care how it behaves on inputs that violate the promise).

To define $\bar{D}$, for $z \in \{0,1\}^{\bar{N}}$ and $t = (1 + 2c \cdot \gamma) \cdot \log(\bar{N})$, let T_z be a circuit that gets as input $O\left(t \cdot N^{2\delta}\right)$ bits, uses these bits to obtain an estimate of $\mu(z)$, and outputs 1 if and only if its estimate is in the interval $\nu \pm \frac{1}{3} \cdot N^{-\delta}$. Note that T_z is of size $\tilde{O}(N^{1+2\delta})$, and that for at least $2/3$ of the inputs of T_z it holds that the estimate is correct up to accuracy $\frac{1}{6} \cdot N^{-\delta}$. In particular, if $\Pr_r[T_z(r) = 0] < 2/3$ then $\left|\mu(z) - \nu\right| \leq \frac{1}{2} \cdot N^{-\delta}$. The circuit $\bar{D}$ gets input z, constructs the circuit T_z, and feeds T_z into the top CAPP gate. Note that the size of $\bar{D}$ is $\tilde{O}(N^{1+2\delta})$, for every $z \in \bar{D}^{-1}(1)$ it holds that $\left|\mu(z) - \nu\right| \leq \frac{1}{2} \cdot N^{-\delta}$, and for all but at most $2^{\bar{N}^{1-\gamma}}$ of the inputs z it holds that $\bar{D}(z) = 1$. [35]

We claim that $\Pr_{s_0}[G_0(1^{\bar{N}}, s_0) \in \bar{D}^{-1}(0)] \leq \frac{1}{2} \cdot N^{-\delta}$ (assuming $\delta > 0$ is sufficiently small). To see this, note that otherwise, the combination of the oracle machine R and of the distinguisher $\bar{D}$ yields an SVN circuit with CAPP gates that computes f in size

$$\tilde{O}\left(\bar{N}^{1+\alpha} + \bar{N}^{\alpha} \cdot \bar{N}^{1+2\delta} + \bar{N}^{1+\beta+(2\alpha-\gamma)}\right) < \bar{N}^{1+\beta-\gamma/4} \ .$$

We now want to transform the foregoing SVN circuit with CAPP gates, denoted C_f, into a randomized SVN circuit. To do so, for each CAPP gate g, the randomized SVN circuit samples $O(\log(\bar{N}))$ inputs for the circuit C' that feeds into g, estimates the acceptance probability of C' up to accuracy .01 and with error $1/N^2$, and sets g's output to 1 iff the estimate is above $1/2$. Note that for every input i to C_f, every non-deterministic choices w by C_f, and every gate g that gets circuit $C' = C'(i, w)$,

- If C' satisfies the promise of CAPP (i.e., it has at most $1/3$ exceptional inputs) then with probability at least $1 - 1/N^2$ we compute g correctly.

- Otherwise, the circuit C_f correctly computes f *regardless of the output* of g. (This is because in the definition of a CAPP gate for $\bar{D}$ we did not enforce its behavior on inputs that violate the promise, and thus the proof holds regardless of the outputs of the original CAPP gate on circuits that violate the promise.)

Recalling that there are at most $\bar{N}^{\alpha} < \frac{1}{3} \cdot N^2$ such CAPP gates in C_f (since the reconstruction procedure makes at most $\bar{N}^{\alpha}$ oracle calls), by a union-bound, for every input and non-deterministic choices, with high probability the randomized SVN circuit has the same output as C_f. This yields a contradiction to our hardness hypothesis for f.

Finally, since for every $z \in \bar{D}^{-1}(1)$ it holds that $\left|\mu(z) - \nu\right| \leq \frac{1}{2} \cdot N^{-\delta}$ and since all but at most $\frac{1}{2} \cdot N^{-\delta}$ outputs of G_0 are in $\bar{D}^{-1}(1)$, using the same calculation as in the proof of Theorem 5.5 we deduce that G "fools" D with error $N^{-\delta}$. ∎

[35]This holds because for all but at most $2^{\bar{N}^{1-\gamma}}$ inputs z it holds that $\mu(z) \in \nu \pm \frac{1}{6} \cdot N^{-\delta}$, and for every such input the circuit T_z accepts with probability at least $2/3$.

6 The $O(n)$ Overhead is Optimal Under #NSETH

In this section we show that under a counting version of the Non-Deterministic Strong Exponential Time Hypothesis (NSETH), which is denoted #NSETH and is weaker than NSETH, for any polynomial function $T(n) = n^k$ (where $k \geq 1$) and $\epsilon > 0$ it holds that $\mathcal{BPTIME}[T(n)] \not\subseteq \mathcal{DTIME}[T(n) \cdot n^{1-\epsilon}]$. Hence, the derandomization in Theorem 1.2 is essentially optimal under #NSETH. (Needless to say, the derandomization is optimal under the stronger assumption NSETH.)

The same result has been noted in [Wil16, Section 3.2] for the special case of $T(n) = n$, and here we adapt the proofs in [Wil16] for the case of larger $T(n)$. Roughly speaking, we consider a problem called k-OV from fine-grained complexity. Under #NSETH, there is no nondeterministic machine that counts the number of satisfying assignments for a k-OV instance in time $n^{k-\epsilon}$, for any $\epsilon > 0$. [36] On the other hand, following [Wil16], we show a Merlin-Arthuer (MA) algorithm that counts the number of satisfying assignments to a k-OV instance in time n^{k-1}. If $\mathcal{BPTIME}[n^{k-1}] \subseteq \mathcal{DTIME}[n^{k-\epsilon}]$ for some $\epsilon > 0$ then the above Merlin-Arther algorithm can be derandomized to a nondeterministic machine that counts satisfying assignments for k-OV in time $n^{k-\epsilon}$, which contradicts #NSETH. [37]

6.1 #NSETH and k-OV

The Non-deterministic Strong Exponential Time Hypothesis (NSETH) was first introduced by Carmosino *et al.* [CGI+16]. We now define a natural counting version of NSETH, denoted #NSETH. Denote by k-TAUT the language of all k-DNFs that are tautologies; then, #NSETH is the following:

Assumption 6.1 (#NSETH). *For every $\epsilon > 0$ there exists a k such that there does not exist a non-deterministic machine M that gets as input a k-*SAT *formula Φ over n variables, runs in time $2^{(1-\epsilon)\cdot n}$ and satisfies the following:*

1. *There exists non-deterministic choices such that M outputs the number of satisfying assignments for Φ.*

2. *For all non-deterministic choices, M either outputs the number of satisfying assignments for Φ or outputs $\perp$.*

Note that #NSETH is weaker than NSETH (since a non-deterministic machine that refutes #NSETH also refutes NSETH). We now introduce the problem k-Orthogonal Vectors (k-OV), and recall the well-known fact that under NSETH it holds that tk-OV $\notin co\mathcal{NTIME}[n^{k-\epsilon}]$, for every $k \in \mathbb{N}$ and $\epsilon > 0$ (i.e., there does not exist a non-deterministic machine that rejects every no instance for some non-deterministic guess, and accepts every yes instance for all non-deterministic guesses).

Definition 6.2 (k-OV). *We define k-OV$_{n,d}$ to be the following promise problem. An input to the problem consists of k sets $A_1, A_2, \ldots, A_k$, where each A_i is a set of n vectors from $\{0,1\}^d$. We*

[36] By "non-deterministically counting" we mean that for all non-deterministic guesses the algorithm either output $\perp$ or the correct number of satisfying assignments, and it must output the correct number for some non-deterministic guess.

[37] This derandomization would be immediate under a hypothesis that refers to promise-problems (i.e., under the hypothesis $pr\mathcal{BPTIME}[n^{k-1}] \subseteq pr\mathcal{DTIME}[n^{k-\epsilon}]$), and we show that it can also be done under a weaker hypothesis that refers only to languages.

say that a tuple $(a_1, a_2, \dots, a_k) \in A_1 \times A_2 \times \dots \times A_k$ *is a* satisfying assignment *if*

$$\sum_{j=1}^{d} \prod_{i=1}^{k} (a_i)_j = 0 \,. \tag{2}$$

An input Φ *to* k*-OV*$_{n,d}$ *is a yes instance if there exists a satisfying assignment for* Φ.

Lemma 6.3 (fine-grained hardness of k-OV under NSETH). *For any integer* $k \geq 2$*, assuming* NSETH*, there is no non-deterministic machine that counts the number of satisfying assignment for a given* k*-OV*$_{n,d}$ *instance in* $n^{k-\epsilon}$ *time, for any* $\epsilon > 0$ *and* $d = \log^2 n$.

For proof of Lemma 6.3, see the standard reduction from k-SAT to k-OV [Wil05]. (The standard reduction actually yields dimension $d = O_k(\log(n))$, but for simplicity we bounded the dimension by $d \leq \log^2(n)$.)

6.2 Proof of Theorem 6.4

Now we ready to prove the following main theorem of this section.

Theorem 6.4 (derandomization has a multiplicative overhead of n, under #NSETH). *Assume that* #NSETH *holds. Then, for every* $k \geq 1$ *and* $\epsilon > 0$ *it holds that* $\mathcal{BPTIME}[n^k] \not\subseteq \mathcal{DTIME}[n^{k+1-\epsilon}]$.

Proof. For $k \geq 1$ and $\epsilon > 0$, assume towards a contradiction that $\mathcal{BPTIME}[n^k] \subseteq \mathcal{DTIME}[n^{k+1-\epsilon}]$. We show that in this case, for some $\epsilon' > 0$ and for $d(n) = \log^2(n)$, there exists a machine that non-deterministically computes the number of satisfying assignments of a given $(k+1)$-OV$_{n,d}$ instance in time $n^{k+1-\epsilon'}$. Relying on Lemma 6.3, this contradicts #NSETH.

Given a $(k+1)$-OV$_{n,d}$ instance $A_1, A_2, \dots, A_{k+1} \in (\{0,1\}^d)^n$, the machine guesses and verifies a prime p in $(n^{2(k+1)}, 2 \cdot n^{2(k+1)}] \cap \mathbb{N}$. Then, the machine constructs $(k+1)$ degree-n polynomial mappings $P^i : \mathbb{F}_p \to \mathbb{F}_p^d$ (i.e., for each $\ell \in [d]$, the ℓ-th coordinate of P^i is a degree-n polynomial) such that for every $i, \ell \in [k+1] \times [d]$ we have that $P^i(j)_\ell = (A_{i,j})_\ell$ for all $j \in [n]$ (where $A_{i,j}$ is the j^{th} vector in A_i). Each of the $(k+1) \cdot d$ corresponding polynomials can be constructed in time $\tilde{O}(n)$, using interpolation via FFT, and therefore the $(k+1)$ polynomial mappings can be constructed in time $\tilde{O}(n)$.

Now, let $F : \{0,1\}^{(k+1)\cdot d} \to \{0,1\}$ be the function that treats its input as $(k+1)$ vectors from $\{0,1\}^d$, and outputs 1 if and noly if the vectors are a satisfying assignment for our given k-OV instance. We define a polynomial $X : \mathbb{F}_p^{(k+1)\cdot d} \to \mathbb{F}_p$ that is a low-degree extension of F (i.e., F and X agree on all inputs in $\{0,1\}^{(k+1)\cdot d} \to \{0,1\}$). To do so, fix a degree-d polynomial $\Phi : \mathbb{F}_p \to \mathbb{F}_p$ such that $\Phi(0) = 1$ and $\Phi(i) = 0$ for all $i \in \{1, 2 \dots, d\}$; note that a suitable Φ can be found in time $\text{poly}(d, \log(p))$ by straightforward interpolation. Then, for $x_1, x_2, \dots, x_{k+1} \in \mathbb{F}_p^d$, we define X as

$$X(x_1, x_2, \dots, x_{k+1}) := \Phi\left(\sum_{\ell=1}^{d} \prod_{i=1}^{k+1} (x_i)_\ell \right).$$

Note that X can be computed in polynomial time (in its input length), since we can construct Φ in time $\text{poly}(d, \log(p))$. Also, note that X has degree $d \cdot (k+1)$ and is an extension of F. Finally, we construct a single-variate polynomial $Q : \mathbb{F}_p \to \mathbb{F}_p$ as follows:

$$Q(j_{k+1}) = \sum_{(j_1, j_2, \ldots, j_k) \in [n]^k} X\left(P^1(j_1), P^2(j_2), \ldots, P^k(j_k), P^{k+1}(j_{k+1})\right).$$

Note that Q is an $O(n \log^2 n)$-degree polynomial that can be described by $O(n \cdot \text{polylog}(n))$ bits and evaluated in time $n^k \cdot \text{poly}(d, \log p)$.

Consider the following problem:

- Given a prime p in $(n^{2(k+1)}, 2 \cdot n^{2(k+1)}] \cap \mathbb{N}$, and an $O(n \log^2 n)$-degree single-variate polynomial H over $\mathbb{F}_p$, together with an k-$\mathsf{OV}_{n,\log^2 n}$ instance Φ, determine whether H is the same as the polynomial Q above constructed from Φ.

Clearly, the problem above is in $\mathcal{BPTIME}[n^k \cdot \text{polylog}(n)]$[38], as one can sample a random element x from $\mathbb{F}_p$ and check whether $Q(x) = H(x)$ (we rely on the fact that the field size is larger than $> n^{2k}$, and that the degree of the polynomial $O(n \log^2 n)$). By our assumption (and a padding argument), it is also in $\mathcal{DTIME}[n^{k+1-\epsilon/2}]$.

From that one can get an algorithm computing the number of satisfying assignments to $(k+1)$-$OV_{n,\log^2 n}$ non-deterministically as follows:

1. Guess and verify a prime p in $(n^{2(k+1)}, 2 \cdot n^{2(k+1)}] \cap \mathbb{N}$.
2. One guesses a polynomial H of $O(n \log^2 n)$ degree over $\mathbb{F}_p$.
3. In $n^{k+1-\epsilon/2}$ time, one reject H if H is not the same polynomial as Q.
4. Otherwise, output $\sum_{i \in [n]} H(i)$.

Note that the number of satisfying assignment is at most n^k, and $p > n^{2(k+1)}$. Hence, assuming the polynomial H passed the test, we have $\sum_{i \in [n]} H(i) = \sum_{i \in [n]} Q(i)$ is the number of satisfying assignments to the given $(k+1) - \mathsf{OV}_{n,d}$ instance. Moreover, the above algorithm runs in non-deterministic $n^{k+1-\epsilon'}$ time for some $\epsilon' > 0$, so we obtain a contradiction to #NSETH. ■

Acknowledgements

We are very grateful to Oded Goldreich and to Ryan Williams for helpful feedback as well as detailed remarks on an early draft of the paper. We also thank Dean Doron, Guy Rothblum, Amnon Ta-Shma, Avishay Tal, and Avi Wigderson for helpful conversations and feedback. The first author wants to thank Guy Rothblum for inviting him to the Weizmann Institute of Science, where part of the work was done.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819702). Lijie Chen is supported by NSF CCF-1741615, a Google Faculty Research Award, and an IBM Fellowship.

[38]The problem comes with a promise that p is a prime. But such promise can be easily checked in polylog(n) time deterministically [AKS04]. Therefore, we can make it as a total function by always outputting 0 when the given p is not a prime.

References

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

[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.

[Bjö14] Andreas Björklund. "Determinant Sums for Undirected Hamiltonicity". In: *SIAM J. Comput.* 43.1 (2014), pp. 280–299. DOI: 10.1137/110839229. URL: https://doi.org/10.1137/110839229.

[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.

[BSW03] Boaz Barak, Ronen Shaltiel, and Avi Wigderson. "Computational analogues of entropy". In: *Proc. 7th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*. 2003, pp. 200–215.

[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.

[CJW20] Lijie Chen, Ce Jin, and Richard Ryan Williams. "Sharp threshold results for computational complexity". In: *Proc. 52nd Annual ACM Symposium on Theory of Computing (STOC)*. 2020.

[CKN18] Marek Cygan, Stefan Kratsch, and Jesper Nederlof. "Fast Hamiltonicity Checking Via Bases of Perfect Matchings". In: *J. ACM* 65.3 (2018), 12:1–12:46. DOI: 10.1145/3148227. URL: https://doi.org/10.1145/3148227.

[CLR+09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. Third. MIT Press, Cambridge, MA, 2009.

[CSS16] Ruiwen Chen, Rahul Santhanam, and Srikanth Srinivasan. "Average-case lower bounds and satisfiability algorithms for small threshold circuits". In: *Proc. 31st Annual IEEE Conference on Computational Complexity (CCC)*. 2016, 1:1–1:35.

[CT19] Lijie Chen and Roei Tell. "Bootstrapping results for threshold circuits "just beyond" known lower bounds". In: *Proc. 51st Annual ACM Symposium on Theory of Computing (STOC)*. 2019, pp. 34–41.

[DMO+19] Dean Doron, Dana Moshkovitz, Justin Oh, and David Zuckerman. "Nearly Optimal Pseudorandomness From Hardness". In: *Electronic Colloquium on Computational Complexity: ECCC* 26 (2019), p. 99.

[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.

[FSU+13] Bill Fefferman, Ronen Shaltiel, Christopher Umans, and Emanuele Viola. "On beating the hybrid argument". In: *Theory of Computing* 9 (2013), pp. 809–843.

[GL89] Oded Goldreich and Leonid A. Levin. "A Hard-core Predicate for All One-way Functions". In: *Proc. 21st Annual ACM Symposium on Theory of Computing (STOC)*. 1989, pp. 25–32.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001. ISBN: 0-521-79172-3. DOI: 10.1017/CBO9780511546891. URL: http://www.wisdom.weizmann.ac.il/\%7Eoded/foc-vol1.html.

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

[GW14] Oded Goldreich and Avi Widgerson. "On derandomizing algorithms that err extremely rarely". In: *Proc. 46th Annual ACM Symposium on Theory of Computing (STOC)*. Full version available online at *Electronic Colloquium on Computational Complexity: ECCC*, 20:152 (Rev. 2), 2013. 2014, pp. 109–118.

[HIL+99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. "A Pseudorandom Generator from any One-way Function". In: *SIAM Journal of Computing* 28.4 (1999), pp. 1364–1396.

[IW99] 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)*. 1999, pp. 220–229.

[KL18] Valentine Kabanets and Zhenjian Lu. "Satisfiability and derandomization for small polynomial threshold circuits". In: *Proc. 22th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*. Vol. 116. LIPIcs. Leibniz Int. Proc. Inform. 2018, Art. No. 46, 19.

[LPT+17] Daniel Lokshtanov, Ramamohan Paturi, Suguru Tamaki, Ryan Williams, and Huacheng Yu. "Beating brute force for systems of polynomial equations over finite fields". In: *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2017, pp. 2190–2202.

[LV08] Ming Li and Paul Vitányi. *An introduction to Kolmogorov complexity and its applications*. Third. Texts in Computer Science. Springer, New York, 2008, pp. xxiv+790.

[MNT93] Yishay Mansour, Noam Nisan, and Prasoon Tiwari. "The computational complexity of universal hashing". In: vol. 107. 1. 1993, pp. 121–133.

[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.

[PPS+05] Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. "An improved exponential-time algorithm for k-SAT". In: *J. ACM* 52.3 (2005), pp. 337–364. DOI: 10.1145/1066100.1066101. URL: https://doi.org/10.1145/1066100.1066101.

[RRV02] Ran Raz, Omer Reingold, and Salil Vadhan. “Extracting all the randomness and reducing the error in Trevisan’s extractors”. In: *Journal of Computer and System Sciences* 65.1 (2002), pp. 97–128.

[Sip88] Michael Sipser. “Expanders, randomness, or time versus space”. In: vol. 36. 3. 1988, pp. 379–383.

[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.

[Tel17] Roei Tell. “Improved Bounds for Quantified Derandomization of Constant-Depth Circuits and Polynomials”. In: *Proc. 32nd Annual IEEE Conference on Computational Complexity (CCC)*. 2017, 18:1 –18:49.

[Tel18] Roei Tell. “Quantified Derandomization of Linear Threshold Circuits”. In: *Proc. 50th Annual ACM Symposium on Theory of Computing (STOC)*. 2018, pp. 855–865.

[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.

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

[Vad12] Salil P. Vadhan. *Pseudorandomness*. Foundations and Trends in Theoretical Computer Science. Now Publishers, 2012.

[Wil05] Ryan Williams. “A new algorithm for optimal 2-constraint satisfaction and its implications”. In: *Theor. Comput. Sci.* 348.2-3 (2005), pp. 357–365. DOI: 10.1016/j.tcs.2005.09.023. URL: https://doi.org/10.1016/j.tcs.2005.09.023.

[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, Art. No. 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.

A The Nisan-Wigderson PRG with small output length

In this section we prove Theorem 4.1; that is, we instantiate the NW PRG for a small output length (i.e., when the hard function has truth-table of size N and the output length is N^{ϵ}), using appropriate modifications a-la [RRV02] to reduce the seed length.

A.1 Preliminaries

First we need the notion of *weak combinatorial designs*, which was introduced by Raz, Reingold, and Vadhan [RRV02].

Definition A.1 (weak designs). *For positive integers $m, \ell, t \in \mathbb{N}$ and an integer $\rho > 1$, an (m, ℓ, t, ρ) weak design is a collection of sets $S_1, \ldots, S_m \subseteq [t]$ such that for every $i \in [m]$ it holds that $|S_i| = \ell$ and $\sum_{j<i} 2^{|S_i \cap S_j|} \leq (m-1) \cdot \rho$.*

We also need the following efficient algorithm for constructing weak designs with parameters that are suitable for us (i.e., large intersections between sets and small universe size t), which is from [Tel18].

Lemma A.2 (constructing weak designs; see [Tel18, Lemma 6.2]). *There exists an algorithm that gets as input $m \in \mathbb{N}$ and $\ell, \rho \in \mathbb{N}$ such that $\log(\rho) = (1-\alpha) \cdot \ell$, where $\alpha \in (0, 1/4)$, and satisfies the following. The algorithm runs in time $\mathrm{poly}(m) \cdot 2^{\ell}$ and outputs an (m, ℓ, t, ρ) weak design, where $t = \lceil (1+4\alpha) \cdot \ell \rceil$.*[39]

We also need the following standard construction of error correcting codes, which is a concatenation of the Reed-Muller codes from Corollary 3.20 and Hadamard codes.

Lemma A.3 (concatenating the RM code with the Hadamard code). *There exists a universal constant $c \geq 1$ such that for every constant $\epsilon \in (0,1)$ there exists a code $Enc : \{0,1\}^m \to \{0,1\}^{m^{1+c\sqrt{\epsilon}}}$ satisfying:*

- *The code is computable in time $O(m^{1+c\sqrt{\epsilon}})$.*
- *The code is locally list-decodable from agreement $1/2 + m^{-\epsilon}$ with decoding time $m^{c \cdot \sqrt{\epsilon}}$ and with list size $2^{m^{o(1)}}$.*

Proof. Let $\tau = \Theta(\epsilon)$ and $\eta = \eta(\tau)$ be two constants to be specified later, and let $\rho = m^{-\tau}$. We use the t-variate Reed-Muller code of degree $d = m^{\eta}$ over $\mathbb{F}_q$, where $t = 1/\eta$ and $q = (8/\rho^2) \cdot d$. Note that the input length to the code (measured in bits) is $\binom{t+d}{d} \cdot \log(q) \geq (d/t)^t \cdot \log(q) \geq m$, and the output length (measured in elements in $\mathbb{F}_q$) is $\bar{m} = q^t = O_{\eta}\left(m/\rho^{2/\eta}\right)$. Also, by Theorem 3.19, this code is locally list-decodable from agreement ρ with decoding circuit size $\mathrm{poly}(t, \log(q), d, 1/\rho) = m^{O(\eta+\tau)}$ and output list size $O(1/\rho)$.

Now, let $Had : \mathbb{F}_q \to \{0,1\}^q$ be the Hadamard code. Recall that for every $\mu > 0$, Hadamard code is $(\frac{1}{2} + \mu, \frac{4}{\mu^2})$-list-docodable in $\mathrm{poly}(\mu^{-1}, \log q)$ time [GL89]. Concatenating the aforementioned Reed-Muller code with the Hadamard code, we obtain a Boolean code Enc with output length $O_{\eta}(m/\rho^{2/\eta} \cdot q) = O_{\eta}(m^{1+2\tau/\eta+\eta+2\tau})$.

Our goal now is to prove that Enc is locally list-decodable as in the statement; the analysis is standard, and we include it for completeness. We set μ so that $\rho = \frac{\mu^3}{4}$. The local decoding algorithm for Enc takes two indexes $i_1 \in [O(1/\rho)], i_2 \in [4/\mu^2]$, and simulates the RM local decoding algorithm with index i_1; whenever the latter algorithm needs to query one coordinate, it decodes that coordinate using the local decoding algorithm for the Hadamard code with index i_2. Note that the decoding time is $\mathrm{poly}(\mu^{-1}, \log q) \cdot m^{O(\eta)}$ and the output list size is $O(\rho^{-1} \cdot \mu^{-2})$.

[39]The running time is stated as $\mathrm{poly}(m, 2^{\ell})$ in [Tel18], but it is easy to see that the algorithm runs in $\mathrm{poly}(m) \cdot 2^{\ell}$ time.

Now we show that given a string y that is $1/2 + 2\mu$ correlated with $Enc(x)$, there exist some indexes $i_1 \in [O(1/\rho)], i_2 \in [4/\mu^2]$ such that the above algorithm successfully decodes x. Note that there are at least μ fraction of blocks such that within each of these blocks it holds that y is at least $1/2 + \mu$ correlated with $Enc(x)$ (a block is a consecutive segment in the output of Enc which corresponds to a single coordinate of the codeword of the RM code); this holds because otherwise we have that y is at most $\mu \cdot 1 + (1-\mu) \cdot (1/2 + \mu) < 1/2 + 2\mu$ correlated with $Enc(x)$, violating our assumption. Since for each block there exists one element from $[4/\mu^2]$ that causes the Hadamard decoder to correctly decode the block, by an averaging argument there exists an $i_2 \in [4/\mu^2]$ such that the Hadamard decoder correctly decodes at least a $\mu^3/4 = \rho$ fraction of the blocks. Hence, by the property of the RM decoder, there exists some index i_1 such that the overall decoder for Enc is successful.

Finally, let us analyze the parameters of Enc. We want to have $\mu = \frac{1}{2} \cdot m^{-\epsilon}$, and so we set τ such that $\rho = \Theta(m^{-3\epsilon})$ (recall that $\rho = \mu^3/4$). The decoding circuit size is thus $m^{O(\eta+\epsilon)}$, the output list size is $m^{O(\epsilon)}$, and the output length of the code is $O_\eta(m^{1+6\epsilon/\eta+\eta+6\epsilon})$. Then, the statement follows by setting $\eta = \sqrt{\epsilon}$. ■

A.2 Proof of Theorem 4.1

Now we are ready to prove Theorem 4.1 (restated below for convenience).

Reminder of Theorem 4.1 *There exists a universal constant c such that for all sufficiently small ϵ, there exists an oracle machine G satisfies the following:*

- *When given input 1^{N^ϵ} and oracle access to a function $f\colon \{0,1\}^{\log(N)} \to \{0,1\}$, the machine G runs in time $N^{1+c\cdot\sqrt{\epsilon}}$ and outputs $2^{\ell(N)}$ strings in $\{0,1\}^{N^\epsilon}$, where $\ell_G(N) = (1 + c\sqrt{\epsilon}) \cdot \log N$.*
- *There exists an oracle machine R that, when given input $x \in \{0,1\}^{\log(N)}$ and oracle access to an $(N^{-\epsilon})$-distinguisher for $G(1^N, \mathbf{u}_{\ell(N)})^f$ and $N^{1-\sqrt{\epsilon}/c}$ bits of advice, runs in time $N^{c\cdot\sqrt{\epsilon}}$ and outputs $f(x)$.*

Proof. Let c_1 be the constant c from Lemma A.3.

The oracle machine G. We first describe the construction of the oracle machine G. We instantiate Lemma A.3 with parameter $\epsilon_{RM} \in (2\epsilon, 3\epsilon)$ such that $N^{-\epsilon_{RM}} = N^{-2\epsilon}/10$. Let $Enc : \{0,1\}^N \to \{0,1\}^{N^{1+\beta}}$ be the corresponding encoder, where $\beta = c_1 \cdot \sqrt{\epsilon_{RM}}$. We identify $f\colon \{0,1\}^{\log(N)} \to \{0,1\}$ with its truth-table of length N in the natural way, and let $\bar{f}$ be the function whose truth-table is $Enc(f)$. Note that $|\bar{f}| = N^{1+\beta}$.

We also apply Lemma A.2 with $m = N^\epsilon$ and $\ell = \log|\bar{f}| = (1+\beta) \cdot \log N$ and ρ such that $\log(\rho) = (1-3\beta) \cdot \ell$ (the hypothesis in Lemma A.2 is that $3\beta < 1/4$, which holds since ϵ is sufficiently small). This yields an (m, ℓ, t, ρ) weak design $\{S_i\}_{i\in[m]}$, where $t = (1+12\beta) \cdot \ell$, that can be computed in time $\mathrm{poly}(m) \cdot 2^\ell = N^{1+O(\beta)}$.

Now, let $\ell_G = \ell_G(N) = t = (1+12\beta) \cdot (1+\beta) \cdot \log N = (1 + O(\beta)) \cdot \log N$. The machine G computes the truth-table of f, computes the encoding $\bar{f} = Enc(f)$, computes the design, and for every $w \in \{0,1\}^{\ell_G}$ outputs the N^ϵ-bit string such that $G(1^N, w)^f_i =$

$\bar{f}(w_{|S_i})$ for $i \in [N^\epsilon]$.[40] Note that given oracle access to f, the time that it takes to compute G for all w is bounded by the time that it takes to compute $\bar{f}$, plus the time it takes to compute the design, plus the time that it takes to print each output (given the design and access to f); this is at most $O\left(N^{1+O(\beta)} + 2^\ell \cdot N^\epsilon\right) = N^{1+O(\beta)}$.

Construction of the oracle machine R. We introduce some useful notation. For two strings $\alpha, \beta \in \{0,1\}^*$, we denote by $\alpha \circ \beta$ the concatenation of α and β. For an integer $\ell \leq |\alpha$, we denote by $\alpha_{\leq \ell}$ the length-ℓ prefix of α. For $x \in \{0,1\}^\ell$ and $w \in \{0,1\}^{t-\ell}$ and $S \subseteq [t]$ of size $|S| = \ell$, we denote by $x \circ_S w$ the string that is obtained by fixing the bits in the set S_i to x, and the bits in the set $[t] \setminus S_i$ to w.

A standard hybrid argument (see, e.g., [RRV02], following [NW94]) shows that there exists an oracle machine P that computes $\bar{f}$ with advantage $N^{-2\epsilon}/10$ over a random guess when given access to an $(N^{-\epsilon})$-distinguisher D for $G(1^N, \mathbf{u}_{\ell_G})^f$. In more detail, for any $N^{-\epsilon}$-distinguisher D there exists an index $i \in [m]$, a suffix $z \in \{0,1\}^{m-i+1}$ and a string $w \in \{0,1\}^{t-\ell}$ such that

$$P(x) \stackrel{\text{def}}{=} D(G(1^N, x \circ_{S_i} w)_{\leq i-1} \circ z) \oplus z_1$$

correctly computes $\bar{f}$ on at least a $1/2 + N^{-2\epsilon}/10$ fraction of the ℓ-bit inputs.

Claim A.3.1. *The function P can be computed in time $O(N^{2\epsilon})$ with a single oracle to D and with $O(N^{1-\Omega(\beta)})$ bits of non-uniform advice.*

Proof. We give (i, z, w) as advice to P. For each $j < i$, note that $G(1^N, x \circ_{S_i} w)_j$ only depends on $|S_i \cap S_j|$ bits of x, hence the its whole truth-table can be stored with $2^{|S_i \cap S_j|}$ bits. By the definition of weak-designs, we have $\sum_{j<i} 2^{|S_i \cap S_j|} < N^\epsilon \cdot \rho = O(N^{1-\Omega(\beta)})$, so we can store the truth-tables of $G(1^N, x \circ_{S_i} w)_j$ for all $j \in [i-1]$. We also give the sets $\{S_i\}_{i \in [m]}$ as advice to P, which takes $O(m \cdot \ell) = O(N^\epsilon \cdot \log N)$ bits. So the overall number of advice bits is bounded by $O(N^{1-\Omega(\beta)})$.

Given $x \in \{0,1\}^\ell$, using the sets $\{S_i\}$ and the truth-tables for $\{G(1^N, x \circ_{S_i} w)_j\}_{j<i}$ that are all given as advice, we can compute $\alpha = G(1^N, x \circ_{S_i} w)_{\leq i-1}$ in time $O(N^{2\epsilon})$. Then we can output $P(x) = D(\alpha \circ z) \oplus z_1$ by a single oracle call to D and linear-time processing involving the advice z. □

Note that P computes a "corrupt" version of $\bar{f}$ (i.e., computes $\bar{f}$ correctly on $1/2 + N^{-2\epsilon}/10$ of inputs). To compute f we run the local decoder for $\bar{f}$ with oracle access to P. By Lemma A.3, the running time is bounded by $N^{O(\beta)}$ and the number of advice bits is bounded by $N^{1-\Omega(\beta)}$.

Finally, note that the local decoding circuits of Lemma A.3 is randomized (see Definition 3.18) and succeeds with probability at least $2/3$ for each input to f. We first repeat the decoder $O(\log N)$ times to amplify the success probability to at least $1 - 1/N^2$. It then follows by a union bound that there is a fixed choice of randomness that makes the amplified decoder successfully compute f on all inputs. Adding such a fixed choice of randomness to the advice, we obtain an $N^{O(\beta)}$ time algorithm computing f exactly on

[40] For a string $w \in \{0,1\}^t$ and a set $S \subseteq [t]$, we use $w_{|S} \in \{0,1\}^{|S|}$ to denote the projection of w onto the coordinates in S. That is, for all $i \in [|S|]$, $(w_{|S})_i = w_{S_i}$, where S_i is the i-th smallest element in S.

all inputs, with $N^{1-\Omega(\beta)} + O(\log N \cdot N^{O(\beta)}) = N^{1-\Omega(\beta)}$ bits of advice and access to the distinguisher D, which completes the proof. ■