# Average-Case Hardness of NP and PH from Worst-Case Fine-Grained Assumptions

**Lijie Chen**
MIT, Boston, MA, USA

**Shuichi Hirahara**
National Institute of Informatics, Tokyo, Japan

**Neekon Vafa**
MIT, Boston, MA, USA

## Abstract

What is a minimal worst-case complexity assumption that implies non-trivial average-case hardness of NP or PH? This question is well motivated by the theory of fine-grained average-case complexity and fine-grained cryptography. In this paper, we show that several standard worst-case complexity assumptions are sufficient to imply non-trivial average-case hardness of NP or PH:

- $\mathsf{NTIME}[n]$ cannot be solved in quasi-linear time on average if $\mathsf{UP} \not\subseteq \mathsf{DTIME}\left[2^{\widetilde{O}(\sqrt{n})}\right]$.
- $\Sigma_2\mathsf{TIME}[n]$ cannot be solved in quasi-linear time on average if $\Sigma_k\mathsf{SAT}$ cannot be solved in time $2^{\widetilde{O}(\sqrt{n})}$ for some constant $k$. Previously, it was not known if even *average-case hardness* of $\Sigma_3\mathsf{SAT}$ implies the average-case hardness of $\Sigma_2\mathsf{TIME}[n]$.
- Under the Exponential-Time Hypothesis (ETH), there is no average-case $n^{1+\varepsilon}$-time algorithm for $\mathsf{NTIME}[n]$ whose running time can be estimated in time $n^{1+\varepsilon}$ for some constant $\varepsilon > 0$.

Our results are given by generalizing the non-black-box worst-case-to-average-case connections presented by Hirahara (STOC 2021) to the settings of fine-grained complexity. To do so, we construct quite efficient complexity-theoretic pseudorandom generators under the assumption that the nondeterministic linear time is easy on average, which may be of independent interest.

**2012 ACM Subject Classification** Theory of computation → Complexity classes

**Keywords and phrases** Average-case complexity, worst-case to average-case reduction

**Digital Object Identifier** 10.4230/LIPIcs.ITCS.2022.45

**Related Version** *Full Version*: https://eccc.weizmann.ac.il/report/2021/166/

**Funding** *Lijie Chen*: supported by NSF CCF-2127597 and an IBM Fellowship.
*Shuichi Hirahara*: supported by JST, PRESTO Grant Number JPMJPR2024, Japan.
*Neekon Vafa*: supported by NSF fellowship DGE-1745302.

**Acknowledgements** We are grateful to Rahul Ilango and Ryan Williams for helpful discussions. In particular, we want to thank Ryan Williams for the observation that an HSG with seed length $O(\log t)$ already suffices for our proof.

## 1 Introduction

One of the central questions in theoretical computer science is to base the existence of one-way functions on the worst-case hardness of NP. Equivalently, this question is well known as the question of whether Heuristica and Pessiland can be excluded from Impagliazzo's five possible worlds [31]. Heuristica is a hypothetical world in which NP is hard in the worst case but NP is easy on average; Pessiland is a hypothetical world in which NP is hard on average but one-way functions do not exist. The existence of a one-way function is indispensable for complexity-theory-based cryptography [34]; thus, excluding these hypothetical worlds (where one-way functions do not exist) from Impagliazzo's five possible worlds would make the security of cryptographic primitives more reliable.

13th Innovations in Theoretical Computer Science Conference (ITCS 2022).
Editor: Mark Braverman; Article No. 45; pp. 45:1–45:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

There are a number of reasons why excluding Heuristica and Pessiland is difficult: Standard proof techniques, such as black-box reductions, hardness amplification procedures, and relativizing proof techniques, are known to be incapable of excluding Heuristica [19, 12, 45, 44, 32]. Similar impossibility results are known for Pessiland [1, 10, 40, 46].

## 1.1 Fine-Grained Average-Case Complexity

To make progress on this challenging question, Ball, Rosen, Sabin, and Vasudevan [4] proposed to study a weak variant of the question: Can we construct a fine-grained one-way function under standard worst-case complexity assumptions? Informally, a *fine-grained one-way function* is a function $f\colon \{0,1\}^* \to \{0,1\}^*$ such that $f$ can be computed in time $t(n)$ on inputs of length $n$ but cannot be inverted in time $t(n)^{1+\varepsilon}$ on average for some time bound $t\colon \mathbb{N} \to \mathbb{N}$ and for some constant $\varepsilon > 0$. In contrast to the standard definition of a one-way function in which we require $\varepsilon$ to be a super constant, a fine-grained one-way function is *slightly* hard to invert; thus, we expect that it is much easier to construct a fine-grained one-way function than a standard one-way function. LaVigne, Lincoln, and Vassilevska Williams [39] showed that some fine-grained *average-case hardness* of Zero-$k$-Clique and $k$-SUM implies the existence of a fine-grained public-key cryptosystem and, in particular, a fine-grained one-way function. This result is relevant to the question of whether one can exclude a fine-grained version of Pessiland in which there is no fine-grained one-way function and the class $\mathsf{NTIME}[n]$ of problems solvable by nondeterministic linear-time algorithms (which contains Zero-$k$-Clique and $k$-SUM) is hard on average.[1]

The open question posed in [4] can be naturally decomposed into the following two open questions: (1) Can we exclude the fine-grained version of Pessiland? In other words, can we construct a fine-grained one-way function from super-linear-time average-case hardness of $\mathsf{NTIME}[n]$? (2) Can we prove super-linear-time average-case hardness of $\mathsf{NTIME}[n]$ under standard worst-case complexity assumptions? Resolving this second question is (almost) necessary to resolve the open question of [4] because the existence of a fine-grained one-way function implies that (the search version of[2]) $\mathsf{NTIME}[n]$ cannot be solved in time $n^{1+\varepsilon}$ on average with respect to some $O(n)$-time samplable distribution for some constant $\varepsilon > 0$. In this paper, we focus on the second question:

▶ **Question 1.** *What is a minimal worst-case complexity assumption that implies "non-trivial"*[3] *average-case hardness of linear-time versions of* NP *or* PH*? Can we exclude a fine-grained version of Heuristica?*[4]

Regarding this question, the original work of [4] implicitly showed that $\mathsf{MATIME}[n]$ (which is a class sandwiched between $\mathsf{NTIME}[n]$ and $\Sigma_2\mathsf{TIME}[n]$) cannot be solved in time $n^{2-o(1)}$ on average under the Strong Exponential-Time Hypothesis (SETH [35]). Specifically, Ball

[1] Note that [39] does not exclude the fine-grained version of Pessiland because the average-case hardness of Zero-$k$-Clique and $k$-SUM is not implied by the average-case hardness of $\mathsf{NTIME}[n]$, which only means *some* problem in $\mathsf{NTIME}[n]$ is average-case hard.

[2] A standard search-to-decision reduction [7] incurs a multiplicative overhead of $O(n)$, which is prohibitively large in the fine-grained setting; thus, it is unclear to us whether the existence of a fine-grained one-way function implies an average-case hard *decision* problem in $\mathsf{NTIME}[n]$.

[3] It is evident that $\mathsf{NTIME}[n]$ cannot be computed in sub-linear time. Here, we aim at proving average-case hardness of NP or PH that does not follow from such an unconditional result.

[4] One possible definition of a fine-grained version of Heuristica is a world in which $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTIME}[n^{1+\varepsilon}]$ for some constant $\varepsilon > 0$ but (the search version of) $\mathsf{NTIME}[n]$ can be solved in time $n^{1+\varepsilon}$ on average with respect to every linear-time samplable distribution for every constant $\varepsilon > 0$.

et al. [4] showed that worst-case hardness assumptions of popular fine-grained complexity problems, such as $\mathsf{OV}$, 3SUM, and Zero-Weight-Triangle, imply the existence of average-case hard problems. For example, they introduced a problem $\mathcal{F}\mathsf{OV}$, which "encodes" $\mathsf{OV}$ as a low degree polynomial over a finite field $\mathcal{F}$, and showed that $\mathcal{F}\mathsf{OV}$ cannot be solved in sub-quadratic time on average unless $\mathsf{OV}$ can be solved in sub-quadratic time. They also observed that $\mathcal{F}\mathsf{OV}$ is a problem in $\mathsf{MATIME}[\widetilde{O}(n)]$[5] using Williams' $\mathsf{MA}$ protocol that refuted an $\mathsf{MA}$ variant of $\mathsf{SETH}$ [47]. As a consequence, the average-case hardness of (a padded version of) $\mathcal{F}\mathsf{OV} \in \mathsf{MATIME}[\widetilde{O}(n)]$ follows from the worst-case hardness of $\mathsf{OV}$, which in particular follows from $\mathsf{SETH}$ [48]. Their subsequent work [5] demonstrated the usefulness of average-case hard problems by constructing a cryptographic system called Proofs of Work. A subsequent line of research [20, 13, 18, 30] showed that worst-case hardness assumptions imply average-case hardness of natural problems, such as counting the number of $k$-Cliques in a random graph, which is in the linear-time variant of $\#\mathsf{P}$ but is unlikely to be in $\mathsf{NTIME}[n]$. Brakerski, Stephens-Davidowitz, and Vaikuntanathan [14] showed that a nearly optimal average-case lower bound for the $k$-SUM problem follows from the worst-case assumption that the Short Independent Vector problem (SIVP) over an $n$-dimensional lattice cannot be approximated to within an $n^{1+\varepsilon}$ factor in time $2^{o(n)}$. We mention that the approximation of SIVP is unlikely to be $\mathsf{NP}$-complete because the problem is known to be in $\mathsf{NP} \cap \mathsf{coNP}$ [23].

Currently, it is an open question to prove that $\mathsf{NTIME}[n]$ is super-linearly average-case hard under $\mathsf{SETH}$. In fact, the proof techniques developed in the above-mentioned literature on fine-grained average-case complexity are unlikely to resolve this question without refuting a slightly nonuniform $\mathsf{AM}$ variant of $\mathsf{SETH}$. The fundamental impossibility result of Feigenbaum and Fortnow [19] shows that if there exists a randomized $k$-time $k$-query nonadaptive "locally random"[6] reduction from a problem $L$ to some average-case problem in $\mathsf{NTIME}[n]$, then $L$ can be solved by an $\mathsf{AM} \cap \mathsf{coAM}$ protocol in time $n \cdot k^{O(1)}$ with $O(\log n)$ bits of advice on inputs of length $n$. For example, the reduction of [4] is a $k$-query nonadaptive reduction from $\mathsf{OV}$ to an average-case version of $\mathcal{F}\mathsf{OV}$ for $k = \log^{O(1)} n$. If $\mathcal{F}\mathsf{OV}$ were in $\mathsf{NTIME}[n]$, then by combining [19, 4] we would obtain that $\mathsf{OV}$ is in $\mathsf{coAMTIME}[\widetilde{O}(n)]/\log n$, which would refute an $\mathsf{AM}/\log n$ variant of $\mathsf{SETH}$. Since the $\mathsf{AM}/\log n$ variant of $\mathsf{SETH}$ is not yet refuted, this connection explains the difficulty of resolving the open question by using the current proof techniques.

## 1.2 Non-Black-Box Worst-Case-to-Average-Case Connections

Recently, Hirahara [25, 28] developed a new type of proof technique that uses non-black-box reductions and is not subject to the impossibility results of [19, 12]. We say that a worst-case-to-average-case reduction is *non-black-box* if the reduction exploits the efficiency of a hypothetical average-case solver. Using non-black-box reductions, the following connections from worst-case hardness to average-case hardness were established.

▶ **Theorem 2** ([28]). *The following hold:*

1. $\mathsf{UP} \not\subseteq \mathsf{DTIME}(2^{O(n/\log n)})$ *implies* $\mathsf{NP} \times \{\mathcal{U}, \mathcal{T}\} \not\subseteq \mathsf{AvgP}$.
2. $\mathsf{PH} \not\subseteq \mathsf{DTIME}(2^{O(n/\log n)})$ *implies* $\mathsf{PH} \times \{\mathcal{U}, \mathcal{T}\} \not\subseteq \mathsf{AvgP}$.
3. $\mathsf{NP} \not\subseteq \mathsf{DTIME}(2^{O(n/\log n)})$ *implies* $\mathsf{NP} \times \{\mathcal{U}, \mathcal{T}\} \not\subseteq \mathsf{Avg_P P}$.

[5] Throughout this paper, we always use $\widetilde{O}(f(n))$ to denote $f(n) \cdot \mathrm{polylog}(f(n))$.

[6] A worst-case-to-average-case nonadaptive reduction $R$ is said to be *locally random* [6] if the query distribution of $R$ on input $x$ depends only on the length $|x|$ of $x$. The reductions presented in [4] satisfy this property. Bogdanov and Trevisan [12] improved [19] and presented a similar impossibility result for reductions that are not locally random.

Here, $\mathcal{U}$ denotes the uniform distribution and $\mathcal{T}$ denotes the tally distribution, i.e., the family $\{\mathcal{T}_n\}_{n\in\mathbb{N}}$ of distributions such that $\mathcal{T}_n$ is the singleton distribution on $\{1^n\}$. $\mathsf{AvgP}$ denotes the class of distributional problems solvable by average-case polynomial-time algorithms or, equivalently, errorless heuristic schemes. $\mathsf{Avg_P P}$ denotes the class of distributional problems solvable by average-case polynomial-time algorithms whose running time can be estimated in polynomial time. We refer the reader to the survey of Bogdanov and Trevisan [11] for more background on average-case complexity.

It would be very interesting to establish average-case lower bounds from weaker worst-case lower bounds, especially due to the fact that the Exponential-Time hypothesis ($\mathsf{ETH}$; [36]) only implies that $\mathsf{NP} \not\subseteq \mathsf{DTIME}(2^{o(n/\log n)})$, which just falls short of satisfying the hypothesis of Item (3) of Theorem 2.[7] Moreover, building on the work of Impagliazzo [32], Hirahara and Nanashima [29] constructed an oracle $\mathcal{O}$ such that $\mathsf{PH}^{\mathcal{O}} \not\subseteq \mathsf{DTIME}(2^{o(n/\log n)})^{\mathcal{O}}$ and yet $\mathsf{DistPH}^{\mathcal{O}} \subseteq \mathsf{AvgP}^{\mathcal{O}}$, meaning that no relativizing proof techniques can show strong average-case lower bounds such as $\mathsf{DistPH} \not\subseteq \mathsf{AvgP}$ from worst-case hardness assumptions such as $\mathsf{PH} \not\subseteq \mathsf{DTIME}(2^{o(n/\log n)})$. We remark that the proof for Item (2) in Theorem 2 in [28] does relativize, while the proofs for Item (1) and (3) "almost relativize" in the sense that the only non-relativizing part of the proofs is the theorem of Buhrman, Fortnow and Pavan [15] showing the existence of a complexity-theoretic pseudorandom generator in Heuristica.

## 1.3 Our Results

In this paper, we generalize the proof techniques from [28] to show that worst-case lower bounds much weaker than $2^{O(n/\log n)}$ already imply super-linear-time average-case lower bounds. Formally, we have the following theorem.

▶ **Theorem 3.** *The following hold:*

1. $\mathsf{UP} \not\subseteq \mathsf{DTIME}\left[2^{O\left(\sqrt{n\log n}\right)}\right]$ *implies* $\mathsf{NTIME}[n] \times \{\mathcal{U}^{\mathsf{para}}\} \not\subseteq \mathsf{Avg}_{1/2}\mathsf{TIME}[\widetilde{O}(n)]$.
2. $\Sigma_k\mathsf{TIME}[n] \not\subseteq \mathsf{DTIME}\left[2^{O\left(\sqrt{n\log n}\right)}\right]$ *implies* $\Sigma_2\mathsf{TIME}[n] \times \{\mathcal{U}^{\mathsf{para}}\} \not\subseteq \mathsf{Avg}_{1/2}\mathsf{TIME}[\widetilde{O}(n)]$ *for every constant* $k$.
3. $\mathsf{ETH}$ *implies* $\mathsf{NTIME}[n] \times \{\mathcal{U}^{\mathsf{para}}\} \not\subseteq \mathsf{Avg}_{\mathsf{DTIME}[n^{1+\varepsilon}]}\mathsf{TIME}[n^{1+\varepsilon}]$ *for some constant* $\varepsilon$.

Here, $\mathcal{U}^{\mathsf{para}}$ denotes a "parameterized uniform distribution", which is a slight generalization of the uniform distribution (see the full version for a formal definition). $\mathsf{Avg}_{1/2}\mathsf{TIME}[\widetilde{O}(n)]$ denotes the class of distributional problems that can be solved by quasi-linear-time errorless heuristics with failure probability at most 1/2.[8]

We present the significance of the three results of Theorem 3 below. The second item of Theorem 3 shows that the second level $\Sigma_2\mathsf{TIME}[n]$ of the linear-time version of $\mathsf{PH}$ is super-linearly average-case hard under the worst-case assumption that $\Sigma_k\mathsf{SAT}$ cannot be solved in time $2^{\widetilde{O}(\sqrt{n})}$ on inputs of length $n$. Here, $\Sigma_k\mathsf{SAT}$ is the problem of deciding, given a Boolean circuit $C$ on $mk$ inputs, whether

$$\exists y_1 \in \{0,1\}^m, \forall y_2 \in \{0,1\}^m, \ldots, \mathcal{Q}_k y_k \in \{0,1\}^m,\ C(y_1,\ldots,y_k) = 1$$

[7] ETH implies that 3SAT cannot be solved in time $2^{o(m)}$ on 3CNF formulas with $m$ variables and $O(m)$ clauses, which implies that $\mathsf{3SAT} \notin \mathsf{DTIME}(2^{o(n/\log n)})$ because 3CNF formulas can be encoded in $n := O(m\log m)$ bits as a binary string.

[8] We mention that the constant 1/2 can be actually improved to any constant smaller than 1.

is true or not, where $\mathcal{Q}_k := \exists$ if $k$ is odd and $\mathcal{Q}_k := \forall$ if $k$ is even. This problem is a canonical complete problem for the $k$-th level $\Sigma_k\mathsf{P}$ of $\mathsf{PH}$ under quasi-linear time reductions (see, e.g., [37]). In particular, $\Sigma_1\mathsf{SAT}$ is equivalent to the Circuit Satisfiability problem and ETH implies that $\Sigma_1\mathsf{SAT} \notin \mathsf{DTIME}(2^{o(n/\log n)})$. Solving $\Sigma_k\mathsf{SAT}$ is known to be notoriously hard: The first algorithm faster than the trivial brute-force algorithm for $k \geq 2$ was given in [42] and runs in time $2^{n-n^{1/(k+1)}}$ on *CNF formulas* with $n$ variables. No non-trivial algorithm for general Boolean circuits is known.

The second item of Theorem 3 makes an important progress on a central and long-standing open question in the theory of structural average-case complexity. The influential paper of Impagliazzo [31] mentioned

> "a central problem in the structure of average-case complexity is: if all problems in NP are easy on average, can the same be said of all problems in the polynomial hierarchy?"

Impagliazzo [32] constructed an oracle under which $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ and for some constant $\alpha > 0$ $\mathsf{Dist}\Sigma_2\mathsf{P} \not\subseteq \mathsf{HeurSIZE}(2^{n^{\alpha}})$, thereby explaining the difficulty of resolving this open problem. Previously, it was unknown whether *average-case easiness* of $\Sigma_{k+1}\mathsf{TIME}[n]$ follows from average-case easiness of $\Sigma_k\mathsf{TIME}[n]$ for any constant $k$. The contrapositive of our second result shows that even *worst-case easiness* of $\Sigma_k\mathsf{TIME}[n]$ follows from average-case easiness of $\Sigma_2\mathsf{TIME}[n]$ for all constants $k$.

The third item of Theorem 3 shows that some super-linear-time average-case hardness of $\mathsf{NTIME}[n]$ follows from $\mathsf{ETH}$, which is one of the popular worst-case assumptions. $\mathsf{Avg}_{\mathsf{DTIME}[n^{1+\varepsilon}]}\mathsf{TIME}[n^{1+\varepsilon}]$ is the class of distributional problems $(L, \mathcal{D})$ such that there exists an errorless heuristic scheme running in time $n^{1+\varepsilon}/\delta$ that solves $L$ with failure probability $\delta$ for any given parameter $\delta$, and moreover whether the heuristic algorithm fails or not can be computed in time $n^{1+\varepsilon}$; see the full version for a precise definition. We mention that the Hamiltonian path problem can be solved on average with respect to the Erdős–Rényi random graph in this average-case sense [22]. We note that, as in our second result, $\Sigma_k\mathsf{TIME}[n] \not\subseteq \mathsf{DTIME}\left[2^{O\left(\sqrt{n\log n}\right)}\right]$ also implies $\mathsf{NTIME}[n] \times \{\mathcal{U}^{\mathsf{para}}\} \not\subseteq \mathsf{Avg}_{\mathsf{DTIME}[\widetilde{O}(n)]}\mathsf{TIME}[\widetilde{O}(n)]$ for every constant $k$; see the full version for details.

The first item of Theorem 3 shows that $\mathsf{NTIME}[n]$ is super-linearly hard on average under the worst-case assumption that $\mathsf{UP}$ cannot be solved in time $2^{\widetilde{O}(\sqrt{n})}$ on inputs of length $n$. $\mathsf{UP}$ is the complexity class of problems that can be solved by nondeterministic polynomial-time algorithms whose accepting path is always at most 1 and is known to characterize the complexity of worst-case injective one-way functions [38, 21]. The trivial deterministic upper bound on $\mathsf{UP}$ is $\mathsf{DTIME}(2^{n^{O(1)}})$ and thus the hypothesis that $\mathsf{UP} \not\subseteq \mathsf{DTIME}\left[2^{O\left(\sqrt{n\log n}\right)}\right]$ is quite plausible.

In addition to Theorem 3, we suggest a new approach to bypass the impossibility results of [19, 12]. We observe that a reduction that uses a limited amount of nondeterministic bits is not subject to these impossibility results. Let $\mathsf{NTIMEGUESS}[t(n), g(n)]$ denote the complexity class of problems solvable by $t(n)$-time nondeterministic algorithms that use at most $g(n)$ nondeterministic bits on inputs of length $n$. We show that the average-case hardness of $\mathsf{NP}$ follows from the worst-case assumption that certificates for $\mathsf{UP}$ cannot be "compressed" into $o(n)$ bits:

▶ **Theorem 4.** *For every constant $\varepsilon > 0$, if* $\mathsf{UP} \not\subseteq \mathsf{NTIMEGUESS}[\mathrm{poly}(n), \varepsilon n]$*, then* $\mathsf{NP} \times \{\mathcal{U}^{\mathsf{para}}\} \not\subseteq \mathsf{Avg}_{1/2}\mathsf{P}$.

Interestingly, for $\varepsilon = 1$, Theorem 4 can be proved by a black-box reduction that uses $n$ nondeterministic bits. Similarly, we also prove the following theorem.

▶ **Theorem 5.** *For every constant $\varepsilon > 0$, if $\mathsf{NP} \not\subseteq \mathsf{NTIMEGUESS}[\mathrm{poly}(n), \varepsilon n]$, then $\Sigma_2\mathsf{P} \times \{\mathcal{U}^{\mathsf{para}}\} \not\subseteq \mathsf{Avg}_{1/2}\mathsf{P}$.*

We remark that for Theorem 4 and Theorem 5, we indeed show a certain "easy-witness lemma" [33, 41]. Take Theorem 5 for example, we indeed prove that assuming $\Sigma_2\mathsf{P} \times \{\mathcal{U}^{\mathsf{para}}\} \subseteq \mathsf{Avg}_{1/2}\mathsf{P}$, for every $L \in \mathsf{NP}$ and every verifier $V$ for $L$[9], $x \in L$ implies that there exists a $y$ such that $\mathsf{K}^{\mathrm{poly}(n)}(y) \leq \varepsilon n$ and $V(x, y) = 1$. In other words, every yes instance $x$ of $L$ admits an "easy witness" $y$ that can be compressed into $\varepsilon n$ bits. This is similar to the easy-witness lemmas proved in [33, 41]. In particular, [41] proved that if NP admits fixed-polynomial size circuits, then for every verifier $V$ for some $L \in \mathsf{NP}$, $x \in L$ implies that there exists a $y$ such that $y$ is the truth-table of a small circuit and $V(x, y) = 1$.

Finally, we mention that the most technical ingredient of our result is a construction of extremely efficient hitting set generators (HSGs) and pseudorandom generators (PRGs) under the assumption that $\mathsf{NTIME}[n]$ is easy on average. More specifically, we prove the following.

▶ **Lemma 6.** *Assuming that $\mathsf{NTIME}[n] \times \{\mathcal{U}^{\mathsf{para}}\} \subseteq \mathsf{Avg}_{1/2}\mathsf{TIME}[\widetilde{O}(n)]$, for every large enough $t$ and $m$ such that $\log t \leq m \leq t$, there exist an HSG $H_{t,m}$ and a PRG $G_{t,m}$ satisfying the following:*

1. *$H_{t,m}$ 0.1-hits $t$-time deterministic algorithms with $m$ bits of advice on $m$-bit inputs.*
2. *$H_{t,m}$ has $O(\log(t))$ seed length and is computable in $\widetilde{O}(t) \cdot \mathrm{poly}(m)$ time.*
3. *$G_{t,m}$ 0.1-fools $t$-time deterministic algorithms with $m$ bits of advice on $m$-bit inputs.*
4. *$G_{t,m}$ has $O(\log(m))$ seed length and is computable in $\widetilde{O}(t) \cdot \mathrm{poly}(m)$ time with $O(\log t)$ bits of advice.*

We note that our construction of the PRG $G_{t,m}$ has a shorter seed length comparing to that of the HSG $H_{t,m}$, at the expense of requiring $O(\log t)$ bits of advice to compute. In [15], $O(\log t)$-seed length PRG fooling $t$-time computation that is computable in $\mathrm{poly}(t)$ time is constructed, under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. Lemma 6 is a fine-grained version of the construction in [15]: we start from a much stronger assumption to get a much more efficient PRG/HSG construction. See Section 2 for an overview of how Lemma 6 is proved and why it is needed, and the full version for formal proofs.[10]

## 2 Techniques Overview

Now we discuss the intuitions behind our results. For concreteness we will first focus on proving the following theorem, and then discuss how to adapt the techniques to prove our other results.

▶ **Theorem 7.** *$\Sigma_2\mathsf{TIME}[n] \times \{\mathcal{U}^{\mathsf{para}}\} \subseteq \mathsf{Avg}_{1/2}\mathsf{TIME}[\widetilde{O}(n)]$ implies $\mathsf{NP} \subseteq \mathsf{TIME}\left[2^{O\left(\sqrt{n \log n}\right)}\right]$.*

Note that the contrapositive of Theorem 7 says that $\mathsf{NP} \not\subseteq \mathsf{TIME}\left[2^{O\left(\sqrt{n \log n}\right)}\right]$ implies $\Sigma_2\mathsf{TIME}[n] \times \{\mathcal{U}^{\mathsf{para}}\} \not\subseteq \mathsf{Avg}_{1/2}\mathsf{TIME}[\widetilde{O}(n)]$.

[9] $V$ is a verifier for $L$ if (1) $V(x, y)$ runs in $\mathrm{poly}(x)$ time and (2) $x \in L$ if and only if there exists $y$ such that $V(x, y) = 1$.

[10] Indeed, in the full version we obtain a general trade-off between the average-case easiness of $\mathsf{NTIME}[n]$ and the efficiency of the constructed PRGs/HSGs.

## 2.1 Review of the Framework in [28]

Since our results crucially build on the framework introduced by Hirahara [28], it would be instructive to first review his approach for worst-case to average-case reduction based on *meta-complexity*, and examine why it requires a $2^{O(n/\log n)}$ worst-case lower bound.

### 2.1.1 Key insight: Computational shallowness implies efficient algorithms

**Computational depth and worst-case to average-case reduction.** One crucial concept introduced in [28] is the $(s,t)$*-time-bounded computational depth*, defined as $\mathsf{cd}^{s,t}(x) := \mathsf{K}^s(x) - \mathsf{K}^t(x)$, where $\mathsf{K}^t(x)$ is the time-bounded Kolmogorov complexity (see the full version for the formal definition). This is a generalization of the computation depth, $\mathsf{cd}^t(x) := \mathsf{K}^t(x) - \mathsf{K}(x)$, defined by [2].

Computational depth provides a fundamental link between worst-case complexity and average-case complexity of $\mathsf{NP}$. In particular, [3] showed that, if $\mathsf{NP}$ is easy on average ($\mathsf{DistNP} \subseteq \mathsf{AvgP}$), then an input $x$ to a language $L \in \mathsf{NP}$ can be solved in $2^{O(\mathsf{cd}^{\mathrm{poly}(n)}(x) + \log|x|)}$ time; that is, one can solve all *shallow* inputs (inputs with small $\mathsf{cd}^{\mathrm{poly}(n)}$) very efficiently.

**Time-bounded computational depth suffices.** The key insight of [28] is that under certain assumptions, one can generalize the above results to hold for time-bounded computational depth as well. For instance, Hirahara [28] proved:

▶ **Lemma 8** (Informal). *If $\Sigma_2\mathsf{P} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{AvgP}$, then for every $L \in \mathsf{NP}$ there is a polynomial $p$ such that for every input $x$ and $t \geq \mathrm{poly}(n)$, one can solve $L$ on input $x$ in $2^{\mathsf{cd}^{t,p(t)}(x)} \cdot \mathrm{poly}(t)$ time.*

Lemma 8 is a significant conceptual improvement: now we can consider multiple values of $t$, and $L$ is easy to solve on input $x$ if $\mathsf{cd}^{t,p(t)}(x)$ is small for *any* of the considered $t$. In more details, let $\tau \in \mathbb{N}$ be a parameter, $t_0 = \mathrm{poly}(n)$, and $t_i = p(t_{i-1})$ for $i \in [\tau]$. We have the following telescoping sum

$$\sum_{i\in[\tau]} \mathsf{cd}^{t_{i-1},t_i}(x) = \sum_{i\in[\tau]} \left[\mathsf{K}^{t_{i-1}}(x) - \mathsf{K}^{t_i}(x)\right] = \mathsf{K}^{t_0}(x) - \mathsf{K}^{t_\tau}(x) \leq \mathsf{K}^{t_0}(x) \leq n + O(1).$$

It then follows that there exists an $i \in [\tau]$ such that $\mathsf{cd}^{t_{i-1},t_i}(x) \leq n/\tau + O(1)$, meaning that we can solve $L(x)$ in $2^{n/\tau} \cdot \mathrm{poly}(t_i) \leq 2^{n/\tau} \cdot \mathrm{poly}(t_\tau)$ time. So our goal now is to carefully choose $\tau$ to minimize the running time. Since $t_\tau = 2^{\log n \cdot O(1)^\tau}$, setting $\tau = \varepsilon \log n$ for a small enough constant $\varepsilon > 0$ leads to $t_\tau \leq 2^{n^{0.99}}$ and the final running time $2^{O(n/\tau)} = 2^{O(n/\log n)}$ for $L \in \mathsf{NP}$. To summarize, from Lemma 8 we can show that $\Sigma_2\mathsf{P} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{AvgP}$ implies $\mathsf{NP} \subseteq \mathsf{DTIME}[2^{O(n/\log n)}]$.

**Bottleneck for improvement: the blow-up function $p(t)$.** The argument above can only achieve running time $2^{O(n/\log n)}$ since $t_\tau = 2^{\log n \cdot O(1)^\tau}$ grows *too fast*: to make $t_\tau$ smaller than $2^n$, we have to take $\tau \leq O(\log n)$, meaning that the running time is at least $2^{\Omega(n/\log n)}$.

Assume for now that $p(t)$ is *linear* in $t$. It follows that $t_\tau = 2^{\log n + O(\tau)}$, and then setting $\tau = \sqrt{n}$ leads to a much faster running time of $2^{O(\sqrt{n})}$. Similarly, by a slightly more involved calculation, if we have $p(t) = \widetilde{O}(t) \cdot \mathrm{poly}(n)$, we can also obtain a running time of $2^{O(\sqrt{n \log n})}$. In this case, since $p(t)$ also depends on $n$, we will write it as $p_n(t)$ to avoid confusion.

### 2.1.2 Why $p(t)$ has to be a large polynomial in [28]

We now know that the key to improving the result in [28] is the blow-up function $p(t)$. Therefore, it is crucial to understand why the blow-up function $p(t)$ has to be a polynomial in Lemma 8.

The proof of Lemma 8 in [28] consists of many components, and $p(t)$ is indeed a composition of several polynomial blow-up functions, each for one component.[11]

We will focus on one important component in the proof of Lemma 8 and understand why the blow-up function of the component, which we denote by $\tau(t)$, has to be a big polynomial. Since the overall blow-up $p(t)$ has to be at least $\tau(t)$, improving this $\tau(t)$ to be quasi-linear in $t$ is *necessary* for improving $p(t)$. Furthermore, it turns out that the ideas behind improving this $\tau(t)$ are already enough to *simultaneously* improve the blow-up functions for all other components.

**Fast algorithm for $\mathsf{Gap}(\mathsf{K}^A$ vs $\mathsf{K})$.** One important component used by [28] is the following algorithm for the $\mathsf{Gap}(\mathsf{K}^A$ vs $\mathsf{K})$ problem. Roughly speaking, if $\Sigma_2\mathsf{P}$ is easy on average, then one can distinguish between strings with small $\mathsf{K}^{t,\mathsf{SAT}}$ complexity and large $\mathsf{K}^{\tau(t)}$ complexity. (See the full version for a formal definition of $\mathsf{K}^{t,A}(x)$.)

▶ **Lemma 9** ([25, 27, 26]). *If $\Sigma_2\mathsf{P} \times \{\mathcal{U}, \mathcal{T}\} \subseteq \mathsf{AvgP}$, then there is an algorithm $A^{\mathsf{Gap}\text{-}\mathsf{K}^{\mathsf{t}}}$ and a polynomial $\tau$ such that*

1. *$A^{\mathsf{Gap}\text{-}\mathsf{K}^{\mathsf{t}}}$ takes $x \in \{0,1\}^n$ and $s, t \in \mathbb{N}$ with $t \geq \max(n, s)$ as inputs, and runs in $\mathrm{poly}(t)$ time.*
2. *If $\mathsf{K}^{t,\mathsf{SAT}}(x) \leq s$, then $A^{\mathsf{Gap}\text{-}\mathsf{K}^{\mathsf{t}}}(x, s, t) = 1$.*
3. *If $\mathsf{K}^{\tau(t)}(x) \geq s + c\log t$, then $A^{\mathsf{Gap}\text{-}\mathsf{K}^{\mathsf{t}}}(x, s, t) = 0$, where $c \geq 1$ is a constant.*

Most importantly, the blow-up function $\tau$ is one crucial component in the overall blow-up function $p$, meaning that $p(t)$ must be at least $\tau(t)$.

### 2.1.3 Analyzing the blow-up function $\tau$ in $A^{\mathsf{Gap}\text{-}\mathsf{K}^{\mathsf{t}}}$: Derandomization is the bottleneck

Now we wish to analyze why the blow-up function $\tau(t)$ in $A^{\mathsf{Gap}\text{-}\mathsf{K}^{\mathsf{t}}}$ has to be a polynomial and see if we can improve it to a quasi-linear function in $t$. We need the following two crucial technical components to discuss the proof of Lemma 9.

**Direct product generators and derandomization from DistNP $\subseteq$ AvgP.** The first ingredient is the direct product generator $\mathsf{DP}_{n,k}\colon \{0,1\}^n \times \{0,1\}^{nk} \to \{0,1\}^{nk+k}$, defined as

$$\mathsf{DP}_{n,k}(y; z_1, z_2, \ldots, z_k) = (z_1, z_2, \ldots, z_k, \langle y, z_1\rangle, \langle y, z_2\rangle, \ldots, \langle y, z_k\rangle),$$

where $\langle y, z_i\rangle$ denotes the inner product between the two vectors $y$ and $z_i$ over $\mathbb{F}_2$. We use $d = nk$ to denote the seed length of the $\mathsf{DP}_{n,k}$ and write $\mathsf{DP}_{n,k}$ as $\mathsf{DP}_k$ when $n$ is clear from the context.

[11] We will not review all the components in this technical overview; the interested reader can refer to [28, Section 2] for an exposition of the ideas behind Lemma 8.

▶ **Theorem 10.** *(Reconstruction property of $\mathsf{DP}_k$; Informal) There exists a randomized polynomial-time oracle algorithm $R^{(\text{-})}$ satisfying the following. Let $D\colon \{0,1\}^{d+k} \to \{0,1\}$ be an oracle such that $D$ distinguishes the output distribution $\mathsf{DP}_{n,k}(y;U_d)$ from $U_{d+k}$; that is*

$$\left| \Pr_{z \sim U_d}[D(\mathsf{DP}_k(y;z)) = 1] - \Pr_{w \sim U_{d+k}}[D(w) = 1] \right| \geq 1/4.$$

*Then with high probability over an internal coin flip of $R^{(\text{-})}$, there exists an advice string $\alpha \in \{0,1\}^{k+O(\log n)}$ such that $R^D(\alpha)$ outputs $y$.*

Note that the reconstruction algorithm $R^{(\text{-})}$ of Theorem 10 is randomized. One can derandomize that reconstruction algorithm using a PRG (pseudorandom generators), whose existence is implied by $\mathsf{DistNP} \subseteq \mathsf{AvgP}$.

▶ **Theorem 11** ([15])**.** $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ *implies that there is an $O(\log n)$-seed* $\operatorname{poly}(n)$*-time computable PRG fooling circuits of size $n$.*

**Proof sketch of Lemma 9.** We will solve the following task instead:

- Given $x \in \{0,1\}^n$ and $s,t \in \mathbb{N}$ as input with the promise that $\mathsf{K}^{t,\mathsf{SAT}}(x) \leq s$, find a witness to $\mathsf{K}^{\tau(t)}(x) \leq s + O(\log t)$.[12]

Note that an algorithm $B$ for the task above immediately gives an algorithm for $A^{\mathsf{Gap}\text{-}\mathsf{K}^t}$: run $B(x,s,t)$ to obtain a program $\Pi$, and accept iff $\Pi$ outputs $x$ in $\tau(t)$ time and $|\Pi| \leq s + c\log t$.

Let $k$ be a parameter to be chosen later. We consider the output distribution $\mathsf{DP}_k(x;U_d)$ (recall that here $d = nk$). For all $z \in \{0,1\}^d$ and a large enough universal constant $c_1 \geq 1$, we have

$$\mathsf{K}^{2t,\mathsf{SAT}}(\mathsf{DP}_k(x;z)) \leq \mathsf{K}^{t,\mathsf{SAT}}(x) + d + c_1 \leq s + d + c_1, \tag{1}$$

since one can first compute $x$ and then compute $\mathsf{DP}_k(x;z)$, and our promise ensures $\mathsf{K}^{t,\mathsf{SAT}}(x) \leq s$.

We then set $k = s + 2c_1$ so that $s + d + c_1 = d + k - c_1$. Now, consider the following function $\bar{D}\colon \{0,1\}^{d+k} \to \{0,1\}$, defined as $\bar{D}(w) := [\mathsf{K}^{2t,\mathsf{SAT}}(w) \leq s + d + c_1]$. Note that $\bar{D}(w)$ can be computed in $O(t)$ $\Sigma_2$-time. Hence, from our assumption that $\mathsf{Dist}\Sigma_2\mathsf{P} \subseteq \mathsf{AvgP}$, for some $T_D(t) = \operatorname{poly}(t)$, we have a $T_D(t)$-time heuristic $D(w)$ such that $D(w) \in \{\bar{D}(w), \bot\}$ for every $w \in \{0,1\}^{d+k}$, and $D(w) = \bar{D}(w)$ for at least half of $w \in \{0,1\}^{d+k}$.

In particular, from the above conditions on $D$, together with (1) and the fact that $s + d + c_1 = |w| - c_1$ for a large enough constant $c_1$, we have:

1. $D(\mathsf{DP}(x;z)) \neq 0$ for all $z \in \{0,1\}^d$. (*i.e.*, $\Pr_{z\sim U_d}[D(\mathsf{DP}_k(x;z)) = 0] = 0$.)
2. $\Pr_{w \sim U_{d+k}}[D(w) = 0] \geq 1/4$.

That is, $D$ is a distinguisher between $\mathsf{DP}(x;U_d)$ and $U_{d+k}$ with a constant advantage. By Theorem 10, there is a $\operatorname{poly}(n)$-time randomized oracle algorithm $R^{(\text{-})}$ that takes $k + O(\log n)$ bits of advice and $D$ as oracle and outputs $x$ with high probability. The composed algorithm $R^D$ runs in $\operatorname{poly}(n) \cdot T_D(t)$ randomized time, and takes $k + O(\log n)$ bits of advice.

[12] *i.e.*, an $(s + c\log t)$-bit program outputting $x$ in $\tau(t)$ time.

**Derandomize $R^D$.** Still, to find a witness to $\mathsf{K}^{\tau(t)}(x) \leq k+O(\log t)$, we have to *derandomize* $R^D$ into a deterministic algorithm. We can achieve this by replacing the randomness of $R^D$ with the outputs of the PRG from Theorem 11. Now $R^D$ is derandomized with a polynomial-overhead into a $p^{\mathsf{dr}}(T_D(t) \cdot \mathrm{poly}(n))$-time deterministic algorithm with $k + O(\log t)$ bits of advice that outputs $x$. Here, $p^{\mathsf{dr}}(\text{-})$ ($\mathsf{dr}$ stands for derandomization) is the derandomization overhead incurred by applying Theorem 11.

To summarize, we have $\mathsf{K}^{p^{\mathsf{dr}}(T_D(t)\cdot\mathrm{poly}(n))}(x) \leq k + O(\log t)$, meaning that we need to set

$$\tau(t) = p^{\mathsf{dr}}(T_D(t) \cdot \mathrm{poly}(n)). \qquad (2)$$

**Improving $\tau(t)$.** As discussed above, we hope to get $\tau(t) = \tau_n(t) = \widetilde{O}(t) \cdot \mathrm{poly}(n)$. Examining (2), we have to ensure two conditions:

$$(1)\ T_D(t) = \widetilde{O}(t) \qquad \text{and} \qquad (2)\ p^{\mathsf{dr}}(t) = p^{\mathsf{dr}}_n(t) \leq \widetilde{O}(t) \cdot \mathrm{poly}(n).$$

Note that $T_D(t) = \widetilde{O}(t)$ can be achieved if we are willing to assume $\Sigma_2\mathsf{TIME}[n] \times \{\mathcal{U}^{\mathsf{para}}\} \subseteq \mathsf{Avg}_{1/2}\mathsf{TIME}[\widetilde{O}(n)]$, as we already did in Theorem 7. Achieving $p^{\mathsf{dr}}(t) \leq \widetilde{O}(t) \cdot \mathrm{poly}(n)$ is much more involved, as we have to get a near-optimal derandomization of the randomized reconstruction algorithm $R^D$ from the assumption that $\Sigma_2\mathsf{TIME}[n] \times \{\mathcal{U}^{\mathsf{para}}\} \subseteq \mathsf{Avg}_{1/2}\mathsf{TIME}[\widetilde{O}(n)]$.

## 2.2 Extremely Efficient HSGs and PRGs from Average-case Easiness of NP

One of our main technical contributions is the construction of *extremely* efficient HSGs (hitting set generators) and PRGs that suffice to derandomize $R^D$ with minor overhead, from the assumption that $\mathsf{NTIME}[n] \times \{\mathcal{U}^{\mathsf{para}}\} \subseteq \mathsf{AvgTIME}_{1/2}[\widetilde{O}(n)]$.

**Requirements on the extremely efficient HSGs for the derandomization of $R^D$.** Recall that we are given a randomized algorithm $R^D$ that runs in $t_1 = \widetilde{O}(t) \cdot \mathrm{poly}(n)$ time and takes $n_{\mathsf{adv}} = k + O(\log n) \leq O(n)$ bits as advice. We further observe that $R^D$ only takes $n_{\mathsf{r}} = \mathrm{poly}(n)$ random bits from Theorem 10, since the oracle $D$ is a uniform deterministic algorithm. Let $R^D(r)$ be the output of $R^D$ given randomness $r \in \{0,1\}^{n_{\mathsf{r}}}$; then, we have

$$\Pr_{r\sim U_{n_{\mathsf{r}}}}[R^D(r) = x] \geq 2/3.$$

We wish to replace the randomness $r$ of $R^D(r)$ by an output of an HSG. To achieve this, we want an HSG $H\colon \{0,1\}^{O(\log t)} \to \{0,1\}^{n_{\mathsf{r}}}$ that 0.1-hits $t_1$-time randomized algorithms that take $n_{\mathsf{adv}}$-bits as advice on $n_{\mathsf{r}}$-bit inputs. Given such an HSG $H$, it follows that there is $u \in \{0,1\}^{O(\log t)}$ such that

$$R^D(H(u)) = x.$$

In other words, given an additional advice $u \in \{0,1\}^{O(\log t)}$, we have a deterministic algorithm $R^D(H(u))$ that outputs $x$. Here, $R^D(H(u))$ uses $k + O(\log t)$ bits of advice in total, and runs in $t_1 + T_G$ time, where $T_G$ is the running time of computing $H(u)$ given $u$. Therefore, to get $\mathsf{K}^{\widetilde{O}(t)\cdot\mathrm{poly}(n)}(x) \leq k + O(\log t)$ from the algorithm $R^D(H(u))$, we need $T_G = \widetilde{O}(t) \cdot \mathrm{poly}(n)$.

**Our HSG construction.** As the technical centerpiece of this paper, we construct the required HSG from the assumption that $\mathsf{NTIME}[n] \times \{\mathcal{U}^{\mathsf{para}}\} \subseteq \mathsf{AvgTIME}[\widetilde{O}(n)]$.

▶ **Lemma 12.** *Assuming that* $\mathsf{NTIME}[n] \times \{\mathcal{U}^{\mathsf{para}}\} \subseteq \mathsf{Avg}_{1/2}\mathsf{TIME}[\widetilde{O}(n)]$*, for every large enough* $t$ *and* $m$ *such that* $\log t \le m \le t$*, there exists an HSG* $H_{t,m}$ *satisfies the following:*

1. $H_{t,m}$ $0.1$*-hits* $t$*-time deterministic algorithms with* $m$ *bits of advice on* $m$*-bit inputs.*
2. $H_{t,m}$ *has* $O(\log(t))$ *seed length and is computable in* $\widetilde{O}(t) \cdot \mathrm{poly}(m)$ *time.*

We mention that combing Lemma 12 with a careful "bootstrapping" argument, we are able to construct a $\widetilde{O}(t) \cdot \mathrm{poly}(m)$-time-computable PRG $G_{t,m}$ fooling similar algorithms, with a near-optimal seed length $O(\log m)$. However, the cost is that now our PRG $G_{t,m}$ requires $O(\log t)$ bits of advice to compute. We also remark that we have a trade-off between the $\mathsf{AvgTIME}_{1/2}$ upper bound and the running time of the PRG (HSG). See the full version for the details.

### 2.2.1 HSGs for "weakly non-uniform" algorithms using the HSG for "low-end" derandomization and strings with high time-bounded Kolmogorov complexity

Note that the most interesting setting for Lemma 12 is when $t \gg m$, since if $m \ge t^{\Omega(1)}$, the running time requirement becomes $\mathrm{poly}(t)$, and we can resort to Theorem 11. Hence, unlike the usual goal in derandomization that one wishes to hit (or fool for PRGs) maximumly non-uniform algorithms (a.k.a. circuits), our goal here is only to hit "weakly non-uniform" algorithms, whose non-uniformity is much less than its running time. But on the other hand, we wish the running time of the HSG $H$ is quasi-linear in the running time $t$ and polynomial in the non-uniformity $m$.[13]

Perhaps surprisingly, we managed to prove Lemma 12 using the HSG construction from [43] intended for "low-end" derandomization (*i.e.*, it was intended for the trade-off between weaker lower bounds and slower derandomization). In particular, [43] gives the following construction of HSG.

▶ **Theorem 13** ([43]). *For every* $t, m \in \mathbb{N}$ *such that* $t \ge m \ge \log t$*, there is a* $t \cdot \mathrm{poly}(m)$*-time algorithm* $\mathsf{SU}_{t,m} \colon \{0,1\}^t \times \{0,1\}^{O(\log t)} \to \{0,1\}^m$ *and a* $\mathrm{poly}(m)$*-time deterministic oracle algorithm* $\mathsf{Rcon}^{(\text{-})}$ *that takes* $\mathrm{poly}(m)$ *bits of advice, such that for every* $x \in \{0,1\}^t$ *and for every* $D$ *that* $0.1$*-avoids* $\mathsf{SU}_{t,m}(x, \text{-})$*, there exists an advice* $\alpha$ *so that for every* $i \in [t]$*,* $\mathsf{Rcon}^D(i, \alpha) = x_i$*.*

Our crucial observation here is that Theorem 13 enables us to construct HSGs fooling weakly non-uniform algorithms with assumptions *much weaker than circuit lower bounds*. This observation is crucial for our proof of Lemma 12.

Formally, we recall a "local" version of $t$-time bounded Kolmogorov complexity, denoted by $\mathsf{K}^t_{\mathsf{loc}}(x)$, such that $\mathsf{K}^t_{\mathsf{loc}}(x)$ is the minimum size of a program $\Pi$ such that $\Pi(i)$ outputs $x_i$ in $t$ time for every $i \in [|x|]$. It is not hard to see that $\mathsf{K}^t_{\mathsf{loc}}(x) \ge t$ is essentially equivalent to saying that the circuit complexity of $x$ is at least $t$ (up to a $\mathrm{polylog}(t)$ factor).

We claim that for some large enough constant $c \ge 1$, $t_1 = t \cdot m^c$ and any $x \in \{0,1\}^*$, if $\mathsf{K}^{t_1}_{\mathsf{loc}}(x) \ge m^c$, then $\mathsf{SU}_{|x|,m}(x, \text{-})$ 0.1-hits all $t$-time algorithms on $m$-bit inputs and with $m$-bit advice. Since if $\mathsf{SU}_{|x|,m}(x, \text{-})$ fails to 0.1-hit some $t$-time algorithm $D$ on $m$-bit inputs with

[13] Our task is somewhat similar to the question of optimal derandomization for $\mathsf{BPTIME}[n^k]$ studied in [17], which aims to derandomize $n^k$-time randomized algorithms with $n$-bit non-uniform advice. The difference is that [17] also requires the seed length of the PRG to be $(1 + o(1)) \log n$ to keep the derandomization overhead at most $n^{1+o(1)}$. But here, we are fine with an $O(\log t)$-length seed.

$m$-bit advice, by Theorem 13, there is an advice $\alpha \in \{0,1\}^{\mathrm{poly}(m)}$ such that for every $i \in [|x|]$, $\mathsf{Rcon}^D(i, \alpha) = x_i$. This procedure $\mathsf{Rcon}^D(\text{-}, \alpha)$ implies that $\mathsf{K}^{t_1}_{\mathsf{loc}}(x) < m^c$, a contradiction to our assumption.

Therefore, to prove Lemma 12, it suffices to resolve the following construction problem.

▶ **Problem 14.** *Assuming that* $\mathsf{NTIME}[n] \times \{\mathcal{U}^{\mathsf{para}}\} \subseteq \mathsf{Avg}_{1/2}\mathsf{TIME}[\widetilde{O}(n)]$*. Given* $t, m \in \mathbb{N}$ *with* $m \ll t$*, in* $t \cdot \mathrm{poly}(m)$ *time construct a string* $x \in \{0,1\}^*$ *satisfying* $\mathsf{K}^t_{\mathsf{loc}}(x) \geq m$*.*

Note that $|x|$ is clearly bounded by the construction time $t \cdot \mathrm{poly}(m)$, so $\mathsf{SU}_{|x|,m}$ is computable in $|x| \cdot \mathrm{poly}(m) = t \cdot \mathrm{poly}(m)$ time, as desired.

### 2.2.2 A refined version of [15]

Our solution to Problem 14 is inspired by the proof of Theorem 11 and follows a similar outline.[14] However, since here we need a $t \cdot \mathrm{poly}(m)$ running time, our algorithm has to be very refined.

Our starting point is the time hierarchy theorem [24]. In particular, there is a language $L \in \mathsf{TIME}[2^n] \setminus \mathsf{TIME}[2^n/n^2]$. Moreover, this language is in fact equipped with a uniform "refuter" $\mathcal{R}$, such that given a code of an algorithm $B$ with running time at most $2^n/n^2$, for every large enough input length $n$, $\mathcal{R}(B, n)$ outputs an $n$-bit input $x_n$ satisfying $L(x_n) \neq B(x_n)$. (See the full version for details.)

Next, we apply an efficient $\mathsf{PCP}$ to $L$ (*e.g.*, [8, 9]), with proof length $\ell = \ell(n) = n + O(\log n)$ and verifier $V$ running in $\mathrm{poly}(n)$ time with $\ell$ bits randomness.

1. For an input $x \in \{0,1\}^n$, $V_x$ takes an oracle $O\colon \{0,1\}^\ell \to \{0,1\}$ and also $\ell$ random bits.
2. If $x \in L$, there exists an oracle $O_x\colon \{0,1\}^\ell \to \{0,1\}$ such that $\Pr_{r \sim U_\ell}\left[V_x^{O_x}(r) = 1\right] = 1$. And there is a uniform algorithm computing the truth table of $O_x$ from $x$ in $2^n \cdot \mathrm{poly}(n)$ time.
3. If $x \notin L$, for all oracle $O\colon \{0,1\}^\ell \to \{0,1\}$, we have $\Pr_{r \sim U_\ell}\left[V_x^{O}(r) = 1\right] \leq 1/2$.

**Our algorithm solving Problem 14.** Our algorithm works as follows[15]:

1. Given $t, m \in \mathbb{N}$, we set $n = \log t + c_1 \log m$ for a large enough constant $c_1$.
2. We construct the code for a "cheating" algorithm $A_{\mathsf{cheat}}$ with running time $2^n/n^2$ from our assumption that $\mathsf{NTIME}[n] \times \{\mathcal{U}^{\mathsf{para}}\} \subseteq \mathsf{Avg}_{1/2}\mathsf{TIME}[\widetilde{O}(n)]$. We then apply the refuter $\mathcal{R}$ to find an input $x_n \in \{0,1\}^n$ such that $A_{\mathsf{cheat}}(x_n) \neq L(x_n)$.
3. We output the truth table of $O_{x_n}$ (our proof will guarantee that $L(x_n) = 1$), which has length $2^n \cdot \mathrm{poly}(n) \leq t \cdot \mathrm{poly}(m)$.

It remains to specify the algorithm $A_{\mathsf{cheat}}$, which is constructed in the following three steps:

1. We first design a Merlin–Arthur algorithm $A_1$ that attempts to solve $L$. On an input $x \in \{0,1\}^n$, it guesses an $m$-bit program $\Pi$ with a running time bound $t$, draws $r \sim U_\ell$, and accepts iff $V_x^{\Pi}(r) = 1$. To summarize, $A_1$ is a Merlin–Arthur algorithm with running time $t \cdot \mathrm{poly}(m)$, proof complexity $m$, and randomness complexity $\ell$.
2. Under the assumption that $\mathsf{NTIME}[n] \times \{\mathcal{U}^{\mathsf{para}}\} \subseteq \mathsf{Avg}_{1/2}\mathsf{TIME}[\widetilde{O}(n)]$, $A_1$ can be simulated by a nondeterministic algorithm $A_2$ with running time $t \cdot \mathrm{poly}(m)$.
3. Again, under the assumption that $\mathsf{NTIME}[n] \times \{\mathcal{U}^{\mathsf{para}}\} \subseteq \mathsf{Avg}_{1/2}\mathsf{TIME}[\widetilde{O}(n)]$, $A_2$ can be simulated by a deterministic algorithm $A_3$ with running time $t \cdot \mathrm{poly}(m)$.
4. We set $A_{\mathsf{cheat}} = A_3$. Since $c_1$ is a large enough constant, it follows that $A_{\mathsf{cheat}}$ runs in $2^n/n^2$ deterministic time.

[14] See [28, Lemma 3.4] for a quick proof sketch of Theorem 11.

[15] In fact, the proof strategy below is also inspired by the refuter-based proof of [16] for proving almost-everywhere circuit lower bounds via the algorithmic method.

From time-hierarchy theorem and the refuter $\mathcal{R}$, it follows that $A_{\mathsf{cheat}}(x_n) \neq L(x_n)$. We claim that this means $L(x_n) = 1$ and $A_{\mathsf{cheat}}(x_n) = A_1(x_n) = 0$. This is because if $L(x_n) = 0$, then on any guessed program $\Pi$, $A_1$ rejects with probability at least $1/2$, and hence $A_{\mathsf{cheat}}(x_n) = A_1(x_n) = 0 = L(x_n)$.

Finally, we claim that when $A_{\mathsf{cheat}}(x_n) = 0 \neq 1 = L(x_n)$, it follows that $\mathsf{K}^t(O_{x_n}) > m$. This is because, if $\mathsf{K}^t(O_{x_n}) \leq m$, then on some guessed $m$-bit program $\Pi$, we would have $\Pi$ computes $O_{x_n}$ in time $t$, and therefore $A_1$ would accept that proof. Consequently $A_{\mathsf{cheat}}(x_n) = A_1(x_n) = 1$, a contradiction to our assumption. To summarize, our algorithm solves Problem 14 in $t \cdot \mathrm{poly}(m)$ time, as desired.

### 2.2.3 Proof for Theorem 7

Finally, combining the algorithm above for Problem 14 with Theorem 13 proves Lemma 12, which gives us the desired derandomization to prove the following variant of Lemma 8:

▶ **Lemma 15** (Informal). *If $\Sigma_2\mathsf{TIME}[n] \times \{\mathcal{U}^{\mathsf{para}}\} \subseteq \mathsf{Avg}_{1/2}\mathsf{TIME}[\widetilde{O}(n)]$, then for every $L \in \mathsf{NP}$ there is $p_n(t) = \widetilde{O}(t) \cdot \mathrm{poly}(n)$ such that for every input $x$ and $t \geq \mathrm{poly}(n)$, one can solve $L$ on input $x$ in $2^{\mathsf{cd}^{t,p(t)}(x)} \cdot \mathrm{poly}(t)$ time.*

As already discussed in Section 2.1.1, Lemma 15 implies a $2^{O\left(\sqrt{n \log n}\right)}$-time algorithm for $\mathsf{NP}$ from the assumption that $\Sigma_2\mathsf{TIME}[n] \times \{\mathcal{U}^{\mathsf{para}}\} \subseteq \mathsf{Avg}_{1/2}\mathsf{TIME}[\widetilde{O}(n)]$, thereby proving Theorem 7.

## 2.3 Average-case Hardness of $\Sigma_2\mathsf{TIME}[n]$ from Worst-case Hardness of $\Sigma_k\mathsf{TIME}[n]$

Finally, we discuss how to prove Item (2) of Theorem 3. We state its contrapositive below.

▶ **Theorem 16.** $\Sigma_2\mathsf{TIME}[n] \times \{\mathcal{U}^{\mathsf{para}}\} \subseteq \mathsf{Avg}_{1/2}\mathsf{TIME}[\widetilde{O}(n)]$ *implies for any constant $k \in \mathbb{N}$* $\Sigma_k\mathsf{TIME}[n] \subseteq \mathsf{DTIME}\left[2^{O\left(\sqrt{n \log n}\right)}\right]$.

To prove Theorem 16, we utilize the advice-efficient HSG construction of [26] (see also the full version for details) to prove the following fine-grained version of Lemma 15, which gives algorithms for any $\mathsf{NTIME}[T(n)]$ language.

▶ **Lemma 17.** *If $\Sigma_2\mathsf{TIME}[n] \times \{\mathcal{U}^{\mathsf{para}}\} \subseteq \mathsf{Avg}_{1/2}\mathsf{TIME}[\widetilde{O}(n)]$, then for every $L \in \mathsf{NTIME}[T(n)]$ there is $p_n(t) = \widetilde{O}(t) \cdot \mathrm{poly}(n)$ such that for every input $x$ and $t \geq \mathrm{poly}(T(n))$, one can solve $L(x)$ in $2^{\mathsf{cd}^{t,p(t)}(x)} \cdot \mathrm{poly}(t)$ time.*

Lemma 17 is very powerful. In particular, set $T(n) = 2^{O\left(\sqrt{n \log n}\right)}$, $t_0 = \mathrm{poly}(T(n))$, and $t_i = p_n(t_{i-1})$ for $i \in [\tau]$, where $\tau$ is a parameter. One can calculate that for $\tau \leq n$, it holds that $t_\tau \leq 2^{O\left(\sqrt{n \log n} + \tau \log n\right)}$, meaning that we can set $\tau = \sqrt{n/\log n}$ to get a $2^{O\left(\sqrt{n \log n}\right)}$ time algorithm for $\mathsf{NTIME}[T(n)]$. That is:

▶ **Corollary 18.** *If $\Sigma_2\mathsf{TIME}[n] \times \mathcal{U}^{\mathsf{para}} \subseteq \mathsf{Avg}_{1/2}\mathsf{TIME}[\widetilde{O}(n)]$, then*

$$\mathsf{NTIME}\left[2^{O\left(\sqrt{n \log n}\right)}\right] \subseteq \mathsf{DTIME}\left[2^{O\left(\sqrt{n \log n}\right)}\right]. \tag{3}$$

We claim that (3) implies that $\Sigma_k\mathsf{TIME}[n] \subseteq \mathsf{DTIME}\left[2^{O\left(\sqrt{n \log n}\right)}\right]$ for every $k \in \mathbb{N}$. This can be shown by a simple induction. First note that the base case for $k = 1$ follows directly from (3).

Now, for $k \geq 2$, assume that $\Sigma_{k-1}\mathsf{TIME}[n] \subseteq \mathsf{DTIME}\left[2^{O\left(\sqrt{n \log n}\right)}\right]$, we will establish the same containment for $\Sigma_k\mathsf{TIME}[n]$. For a language $L \in \Sigma_k\mathsf{TIME}[n]$, by definition, there is a verifier $V(x, y)$ computable in $\Pi_{k-1}\mathsf{TIME}[n]$, such that $L(x) = 1$ iff there exists $y \in \{0, 1\}^{O(n)}$ with $V(x, y) = 1$. From our induction hypothesis, we also have that $\Pi_{k-1}\mathsf{TIME}[n] \subseteq \mathsf{DTIME}\left[2^{O\left(\sqrt{n \log n}\right)}\right]$, meaning that $V(x, y)$ can be computed in $2^{O\left(\sqrt{n \log n}\right)}$ time (note that $|x| + |y| \leq O(n)$).

Hence, it follows that $L \in \mathsf{NTIME}\left[2^{O\left(\sqrt{n \log n}\right)}\right]$, and consequently $L \in \mathsf{TIME}\left[2^{O\left(\sqrt{n \log n}\right)}\right]$ as well, which completes the proof.

## References

1 Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on NP-hardness. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 701–710, 2006. doi:10.1145/1132516.1132614.

2 Luis Antunes, Lance Fortnow, Dieter van Melkebeek, and N. V. Vinodchandran. Computational depth: Concept and applications. *Theor. Comput. Sci.*, 354(3):391–404, 2006. doi:10.1016/j.tcs.2005.11.033.

3 Luis Filipe Coelho Antunes and Lance Fortnow. Worst-case running times for average-case algorithms. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, pages 298–303. IEEE Computer Society, 2009. doi:10.1109/CCC.2009.12.

4 Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Average-case fine-grained hardness. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 483–496, 2017. doi:10.1145/3055399.3055466.

5 Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Proofs of Work From Worst-Case Assumptions. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, pages 789–819, 2018. doi:10.1007/978-3-319-96884-1_26.

6 Donald Beaver, Joan Feigenbaum, Joe Kilian, and Phillip Rogaway. Locally Random Reductions: Improvements and Applications. *J. Cryptol.*, 10(1):17–36, 1997. doi:10.1007/s001459900017.

7 Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby. On the Theory of Average Case Complexity. *J. Comput. Syst. Sci.*, 44(2):193–219, 1992. doi:10.1016/0022-0000(92)90019-F.

8 Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Short pcps verifiable in polylogarithmic time. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*, pages 120–134. IEEE Computer Society, 2005. doi:10.1109/CCC.2005.27.

9 Eli Ben-Sasson and Emanuele Viola. Short PCPs with projection queries. In *Proc. 41st Internat. Colloq. on Automata, Languages and Programming (ICALP'14)*, pages 163–173. Springer, 2014. doi:10.1007/978-3-662-43948-7_14.

10 Andrej Bogdanov and Christina Brzuska. On Basing Size-Verifiable One-Way Functions on NP-Hardness. In *Proceedings of the Theory of Cryptography Conference (TCC)*, pages 1–6, 2015. doi:10.1007/978-3-662-46494-6_1.

11 Andrej Bogdanov and Luca Trevisan. Average-Case Complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1), 2006. doi:10.1561/0400000004.

12 Andrej Bogdanov and Luca Trevisan. On Worst-Case to Average-Case Reductions for NP Problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006. doi:10.1137/S0097539705446974.

13 Enric Boix-Adserà, Matthew S. Brennan, and Guy Bresler. The Average-Case Complexity of Counting Cliques in Erdős-Rényi Hypergraphs. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 1256–1280, 2019. doi:10.1109/FOCS.2019.00078.

14 Zvika Brakerski, Noah Stephens-Davidowitz, and Vinod Vaikuntanathan. On the hardness of average-case k-sum. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPIcs*, pages 29:1–29:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.APPROX/RANDOM.2021.29.

15 Harry Buhrman, Lance Fortnow, and Aduri Pavan. Some results on derandomization. *Theory Comput. Syst.*, 38(2):211–227, 2005. doi:10.1007/s00224-004-1194-y.

16 Lijie Chen, Xin Lyu, and R. Ryan Williams. Almost-everywhere circuit lower bounds from non-trivial derandomization. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1–12. IEEE, 2020. doi:10.1109/FOCS46700.2020.00009.

17 Lijie Chen and Roei Tell. Simple and fast derandomization from very hard functions: eliminating randomness at almost no cost. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 283–291. ACM, 2021. doi:10.1145/3406325.3451059.

18 Mina Dalirrooyfard, Andrea Lincoln, and Virginia Vassilevska Williams. New Techniques for Proving Fine-Grained Average-Case Hardness. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 774–785, 2020. doi:10.1109/FOCS46700.2020.00077.

19 Joan Feigenbaum and Lance Fortnow. Random-Self-Reducibility of Complete Sets. *SIAM J. Comput.*, 22(5):994–1005, 1993. doi:10.1137/0222061.

20 Oded Goldreich and Guy N. Rothblum. Counting t-Cliques: Worst-Case to Average-Case Reductions and Direct Interactive Proof Systems. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 77–88, 2018. doi:10.1109/FOCS.2018.00017.

21 Joachim Grollmann and Alan L. Selman. Complexity Measures for Public-Key Cryptosystems. *SIAM J. Comput.*, 17(2):309–335, 1988. doi:10.1137/0217018.

22 Yuri Gurevich and Saharon Shelah. Expected Computation Time for Hamiltonian Path Problem. *SIAM J. Comput.*, 16(3):486–502, 1987. doi:10.1137/0216034.

23 Venkatesan Guruswami, Daniele Micciancio, and Oded Regev. The complexity of the covering radius problem. *Comput. Complex.*, 14(2):90–121, 2005. doi:10.1007/s00037-005-0193-y.

24 Juris Hartmanis and Richard E Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

25 Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 247–258. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00032.

26 Shuichi Hirahara. Characterizing average-case complexity of PH by worst-case meta-complexity. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 50–60. IEEE, 2020. doi:10.1109/FOCS46700.2020.00014.

27 Shuichi Hirahara. Non-disjoint promise problems from meta-computational view of pseudorandom generator constructions. In *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 20:1–20:47. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CCC.2020.20.

28 Shuichi Hirahara. Average-case hardness of NP from exponential worst-case hardness assumptions. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 292–302. ACM, 2021. doi:10.1145/3406325.3451065.

29 Shuichi Hirahara and Mikito Nanashima. On worst-case learning in relativized heuristica. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, 2021.

30 Shuichi Hirahara and Nobutaka Shimizu. Nearly Optimal Average-Case Complexity of Counting Bicliques Under SETH. In *Proceedings of the 2021 ACM-SIAM Symposium on*

*Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2346–2365, 2021. doi:10.1137/1.9781611976465.140.

31 Russell Impagliazzo. A Personal View of Average-Case Complexity. In *Proceedings of the Structure in Complexity Theory Conference*, pages 134–147, 1995. doi:10.1109/SCT.1995.514853.

32 Russell Impagliazzo. Relativized Separations of Worst-Case and Average-Case Complexities for NP. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 104–114, 2011. doi:10.1109/CCC.2011.34.

33 Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. Comput. Syst. Sci.*, 65(4):672–694, 2002. doi:10.1016/S0022-0000(02)00024-7.

34 Russell Impagliazzo and Michael Luby. One-way Functions are Essential for Complexity Based Cryptography (Extended Abstract). In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 230–235, 1989. doi:10.1109/SFCS.1989.63483.

35 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

36 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

37 Hamid Jahanjou, Eric Miles, and Emanuele Viola. Local Reductions. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 749–760, 2015. doi:10.1007/978-3-662-47672-7_61.

38 Ker-I Ko. On Some Natural Complete Operators. *Theor. Comput. Sci.*, 37:1–30, 1985. doi:10.1016/0304-3975(85)90085-4.

39 Rio LaVigne, Andrea Lincoln, and Virginia Vassilevska Williams. Public-Key Cryptography in the Fine-Grained Setting. *IACR Cryptology ePrint Archive*, 2019:625, 2019. URL: https://eprint.iacr.org/2019/625.

40 Noam Livne. On the Construction of One-Way Functions from Average Case Hardness. In *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 301–309, 2010. URL: http://conference.iiis.tsinghua.edu.cn/ICS2010/content/papers/24.html.

41 Cody D. Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime from a new easy witness lemma. *SIAM J. Comput.*, 49(5), 2020. doi:10.1137/18M1195887.

42 Rahul Santhanam and Richard Ryan Williams. Beating Exhaustive Search for Quantified Boolean Formulas and Connections to Circuit Complexity. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 231–241, 2015. doi:10.1137/1.9781611973730.18.

43 Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *J. ACM*, 52(2):172–216, 2005. doi:10.1145/1059513.1059516.

44 Emanuele Viola. On Constructing Parallel Pseudorandom Generators from One-Way Functions. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 183–197, 2005. doi:10.1109/CCC.2005.16.

45 Emanuele Viola. The complexity of constructing pseudorandom generators from hard functions. *Computational Complexity*, 13(3-4):147–188, 2005. doi:10.1007/s00037-004-0187-1.

46 Hoeteck Wee. Finding Pessiland. In *Proceedings of the Theory of Cryptography Conference (TCC)*, pages 429–442, 2006. doi:10.1007/11681878_22.

47 R. Ryan Williams. Natural proofs versus derandomization. *SIAM J. Comput.*, 45(2):497–529, 2016. doi:10.1137/130938219.

48 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. doi:10.1016/j.tcs.2005.09.023.