Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 3 (2018)

Proving that $prBPP = prP$ is as hard as "almost" proving that $P \neq NP$

Roei Tell *

January 15, 2018

Abstract

The main result in this paper is that any proof that *promise-BPP = promise-P* necessitates proving that polynomial-sized circuits cannot simulate non-deterministic machines that run in arbitrarily small super-polynomial time (i.e., $NTIME[n^{f(n)}] \not\subseteq P/\text{poly}$, for essentially any $f(n) = \omega(1)$). The super-polynomial time bound in the conclusion of the foregoing conditional statement cannot be improved (to conclude that $NP \not\subseteq P/\text{poly}$) without *unconditionally* proving that $P \neq NP$.

This paper is a direct follow-up to the very recent breakthrough of Murray and Williams (ECCC, 2017), in which they proved a new "easy witness lemma" for $NTIME[o(2^n)]$. Our main contribution is conceptual, in highlighting the *strong "barriers" for proving $prBPP = prP$* (and even for proving much weaker statements) that can be demonstrated using their techniques. Following their approach, we apply the new lemma within the celebrated proof strategy of Williams (SICOMP, 2013), and derive the main result by using a parameter setting that is different than the ones they considered. We also include an alternative proof of the main theorem, which does not rely on the work of Murray and Williams, but rather uses a modification of the well-known lower bound of Santhanam (SICOMP, 2009).

1 Introduction

The $BPP = P$ conjecture asserts that any decision problem that can be efficiently solved using randomness (while allowing for a small error) can also be efficiently solved deterministically. In other words, the conjecture asserts that randomness is not needed to efficiently solve decision problems. This conjecture is central to the complexity-theoretic study of the role of randomness in computation.

The $BPP = P$ conjecture is often interpreted as an *algorithmic* problem, namely the problem of explicitly constructing efficient deterministic algorithms that simulate randomized algorithms. In fact, a version of the conjecture is *equivalent* to the conjectured existence of an algorithm for a single, specific problem (i.e., the *circuit acceptance probability problem*; see Proposition 4). However, as will be discussed next, it has also been

*Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel. Email: roei.tell@weizmann.ac.il

ISSN 1433-8092

known for at least two decades that the conjecture is actually intimately related to *circuit lower bounds*; that is, the conjecture is related to lower bounds for non-uniform models of computation. The attempts to prove such lower bounds have long been considered a prominent path to make progress on the $\mathcal{P} \neq \mathcal{NP}$ conjecture.

Informally, following a very recent breakthrough by Murray and Williams [MW17], the main result in this paper considerably strengthens the knowns connection between the $\mathcal{BPP} = \mathcal{P}$ conjecture and circuit lower bounds. To present the new result, let us first spell out the latter connections. On the one hand, any proof of sufficiently strong circuit lower bounds would also prove the conjecture; specifically, if there is a function in $\mathcal{E}$ that requires exponential-sized circuits, then $\mathcal{BPP} = \mathcal{P}$ (and even $pr\mathcal{BPP} = pr\mathcal{P}$, i.e. the promise-problem versions of $\mathcal{BPP}$ and of $\mathcal{P}$ are equal; see [IW99], which relies on the hardness-randomness paradigm [Yao82, BM84, NW94]). On the other hand, any proof that $pr\mathcal{P} = pr\mathcal{BPP}$ (or even of much weaker instances of this conjecture) implies long-sought circuit lower bounds (see, e.g., [IW99, IKW02, KI04, Wil13]).

A specific celebrated example of such a connection is that any proof that $pr\mathcal{BPP} = pr\mathcal{P}$ implies that there exists a function in $\mathcal{NEXP}$ that cannot be computed by any polynomial-sized circuit family [IKW02]. Very recently, Murray and Williams [MW17] proved a breakthrough result that allows to significantly strengthen this statement: In particular, we observe that an immediate corollary of [MW17, Thm 1.2] is that *if* $pr\mathcal{BPP} = pr\mathcal{P}$*, then there exists a function in* $NTIME[n^{\text{poly}\log(n)}]$ (rather than $\mathcal{NEXP}$) *that cannot be computed by any polynomial-sized circuit family*. We believe that this corollary is a fundamental result that is worth spelling out and highlighting. Furthermore, this result can be strengthened, and doing so is the technical contribution in this paper.

Specifically, the main result in the current paper is that if $pr\mathcal{BPP} = pr\mathcal{P}$, then there exists a function in $NTIME[n^{\omega(1)}]$ that cannot be computed by any polynomial-sized circuit family; that is, $NTIME[n^{\omega(1)}] \not\subseteq \mathcal{P}/\text{poly}$. Note that this circuit lower bound is so strong that an improvement in the time bound of the non-deterministic class in this bound (i.e., in $NTIME[n^{\omega(1)}] \not\subseteq \mathcal{P}/\text{poly}$) would immediately imply that $\mathcal{P} \neq \mathcal{NP}$. For further discussion of the meaning of this lower bound, see Section 1.1.

Theorem 1 *(main theorem; informal). If* $pr\mathcal{BPP} = pr\mathcal{P}$*, then, for essentially any super-constant function* $f(n) = \omega(1)$*, there exists a set in* $NTIME[n^{f(n)}] \setminus \mathcal{P}/\text{poly}$.

One might a-priori hope to further improve the conclusion of Theorem 1, and prove a theorem of the form "if $pr\mathcal{BPP} = pr\mathcal{P}$, then $\mathcal{NP} \not\subseteq \mathcal{P}/\text{poly}$ (and $\mathcal{P} \neq \mathcal{NP}$)". However, we note that such a result cannot be proved without *unconditionally* proving that $\mathcal{P} \neq \mathcal{NP}$. That is, any proof of the conditional statement "$\mathcal{BPP} = \mathcal{P} \implies \mathcal{P} \neq \mathcal{NP}$" would *unconditionally* imply that $\mathcal{P} \neq \mathcal{NP}$ (see Proposition 9). Therefore, the conclusion of Theorem 1 is optimal in this sense.

The hypothesis in the conditional statement in Theorem 1 refers to classes of *promise problems*, and is thus stronger than the hypothesis that $\mathcal{BPP} = \mathcal{P}$ (which refers to decision problems with the trivial promise). However, the promise-problem version of the conjecture is natural and well-studied by itself, and moreover, the strongest evidence that currently suggests that $\mathcal{P} = \mathcal{BPP}$ also suggests that $pr\mathcal{P} = pr\mathcal{BPP}$ (since it is based on constructions of pseudorandom generators; see [IW99]).

We comment that Theorem 1 can be proved without relying on the results of Murray and Williams [MW17]. In particular, there exists a relatively simple proof for the theorem that is based on (a modification of) the well-known circuit lower bound of Santhanam [San09]. We present this alternative proof of Theorem 1 in Section 6.[1]

However, we will deduce Theorem 1 as a corollary of a stronger theorem, which is proved using the results of Murray and Williams [MW17]. Specifically, similarly to [Wil13, MW17], the hypothesis that we actually use is much weaker than $prBPP = prP$: We only assume that there exists a deterministic algorithm for the *circuit acceptance probability problem* (see Definition 3) that runs in time *noticeably smaller* than the naive deterministic algorithm for this problem. (To see that this follows from $prBPP = prP$, see Proposition 4.) Specifically, Theorem 1 is a corollary of the following theorem:

Theorem 2 *(main theorem, stronger version; informal). Assume that for some constant $\epsilon > 0$ there exists an algorithm that gets as input a circuit C of size m over v variables, runs in time $2^{(1-\epsilon)\cdot v} \cdot \mathrm{poly}(m)$, and distinguishes between the case that the acceptance probability of C is one and the case that the acceptance probability of C is at most $1/3$. Then, for essentially any super-constant function $f(n) = \omega(1)$ there exists a set in $NTIME[n^{f(n)}] \setminus \mathcal{P}/\mathrm{poly}$.*

Theorem 2 has a form that is very similar to [MW17, Thms. 1.1 & 1.2], yet its parameters are different. The latter theorems can be considered as significant strengthenings of the celebrated result of Williams [Wil13], and their proofs in fact follow Williams' original proof strategy. In the original result, the hypothesis is that there exists a "weak" circuit-analysis algorithm, and the conclusion is a "weak" (yet highly non-trivial) circuit lower bound. In contrast, in Theorem 2 (as well as in [MW17]) the hypothesis is somewhat stronger, but the conclusion is considerably stronger. The key new technical component that allows to improve Williams' original result is the new "easy witness lemma" of Murray and Williams [MW17] (see Section 3 for details).

The hypotheses of Theorem 2 can even be further relaxed (since the theorem's proof relies on Williams' proof strategy, which is well-known to support such further relaxations).[2] For a discussion of such relaxations see the end of Section 5.

1.1 Theorem 1 as a "barrier" for proving $prBPP = prP$

Our main interpretation of Theorem 1 is as posing a strong "barrier" for proving that $prBPP = prP$: The theorem implies that proving $prBPP = prP$ is *at least as hard* as proving that $NTIME[n^{\omega(1)}] \not\subseteq \mathcal{P}/\mathrm{poly}$. The foregoing statement significantly strengthens previously-known "barriers" for proving $prBPP = prP$ (cf., e.g., [IKW02]). Let us now try to clarify the meaning of this "barrier"; that is, we ask what are the conceptual implications of proving that $NTIME[n^{\omega(1)}] \not\subseteq \mathcal{P}/\mathrm{poly}$. Note that the latter statement asserts that polynomial-sized circuits cannot simulate *both super-polynomial running time and non-determinism*.

[1] The idea for the alternative proof was suggested to us by Igor Oliveira after a preliminary version of this paper appeared online.

[2] For example, the conclusion of Theorem 2 also follows from "non-deterministic derandomization", and in particular follows from the hypothesis that $pr\text{-}co\mathcal{RP} \subseteq pr\text{-}\mathcal{NP}$.

On the one hand, one can consider "$NTIME[n^{\omega(1)}] \not\subseteq \mathcal{P}/\text{poly}$" to be a weaker form of "$\mathcal{NP} \not\subseteq \mathcal{P}/\text{poly}$" (i.e., of the assertion that polynomial-sized circuits cannot simulate polynomial non-determinism). From this view, Theorem 1 implies that proving that $pr\mathcal{BPP} = pr\mathcal{P}$ is as hard as "almost" proving that $\mathcal{P} \neq \mathcal{NP}$ (as suggested by the title of the paper). On the other hand, as pointed out by Ryan Williams, one could also view the statement "$NTIME[n^{\omega(1)}] \not\subseteq \mathcal{P}/\text{poly}$" as a weaker form of "$DTIME[n^{\omega(1)}] \not\subseteq \mathcal{P}/\text{poly}$". The latter statement asserts that polynomial-sized circuits cannot simulate algorithms with *super-polynomial running time*; this is a "strengthened time-hierarchy theorem" (in which we compare uniform algorithms to non-uniform circuits). From this perspective, Theorem 1 implies that proving $pr\mathcal{BPP} = pr\mathcal{P}$ is as hard as proving (a weak form of) a "strengthened time-hierarchy theorem". Needless to say, both views imply a "barrier" for proving $pr\mathcal{BPP} = pr\mathcal{P}$ that seems challenging at the current time.

1.2 Organization

We begin the paper by providing a very brief digest of the proof, which is intended primarily for experts; this digest appears in Section 2, and can be safely skipped. Readers who wish to skip this part are encouraged to begin by reading a more detailed overview, which also describes known approaches that are needed for the proof, and appears in Section 3. Then, in Section 4 we present preliminary formal definitions, and in Section 5 we formally prove Theorem 2. Finally, in Section 6 we present an alternative proof of Theorem 1.

2 A brief digest (intended primarily for experts)

The proof of Theorem 2 uses the celebrated proof strategy of Williams [Wil13], who showed that any non-trivial derandomization of a circuit class implies lower bounds for that class against $\mathcal{NEXP}$. The key technical ingredient in Williams' proof strategy is an "easy witness lemma" (as in [IKW02]): A lemma that asserts that if some unexpected complexity collapse occurs (e.g., $\mathcal{NEXP} \subseteq \mathcal{P}/\text{poly}$), then there exist *small* circuits that encode *large* witnesses in proof systems (e.g., for $\mathcal{NE}$).

Very recently, Murray and Williams [MW17] proved a new easy witness lemma, with significantly improved parameters. Indeed, this new lemma allows to improve the results obtained via Williams' proof strategy. Specifically, Murray and Williams showed that any non-trivial derandomization of a circuit class implies lower bounds for this class against $NTIME[n^{\text{poly}\log(n)}]$. As mentioned in the introduction, a weak version of Theorem 1 follows immediately as a corollary of this result (i.e., the result implies that if $pr\mathcal{BPP} = pr\mathcal{P}$, then $NTIME[n^{\text{poly}\log(n)}] \not\subseteq \mathcal{P}/\text{poly}$).

The main observation that paves the path to Theorem 2 is that *the proof strategy of Williams can yield a stronger conclusion when starting from a stronger hypothesis*. (This observation also underlies [MW17, Thm 1.1].) Accordingly, we state and prove a parametrized "derandomization implies lower bounds" theorem (see Theorem 10), which uses the new easy witness lemma within the proof strategy of Williams with general parame-

ters.[3] This theorem has the following form: Assuming a sufficiently strong derandomization hypothesis, and taking any two functions $t > s$ that have a sufficient "gap" between them, it holds that $NTIME[t] \not\subseteq SIZE[s]$. The required "gap" between s and t is determined (in part) by the strength of the derandomization hypothesis. In particular, under the hypothesis in Theorem 2, we can take both s and t to be arbitrarily-small super-polynomial functions (i.e., $t(n) = n^{\omega(1)}$ and $s(n) = n^{\omega(1)} \ll t(n)$), which allows us to deduce that $NTIME[t]$ is not contained in $SIZE[s] \supseteq \mathcal{P}/\text{poly}$.

3 Proof overview: Easy witnesses for $NTIME[n^{\omega(1)}]$

For simplicity, we overview the proof of Theorem 1 (the proof of Theorem 2 is very similar, just more careful with the parameters). To do so we need to define the circuit acceptance probability problem (or CAPP, in short): Given as input the description of a circuit C, the problem is to distinguish between the case that the acceptance probability of C is at least $2/3$ and the case that the acceptance probability of C is at most $1/3$. The problem can be easily solved using *randomness* (by sampling inputs for C), but it is not a-priori clear how to solve it *deterministically*. In fact, a deterministic polynomial-time algorithm for CAPP exists if and only if $pr\mathcal{BPP} = pr\mathcal{P}$ (see Proposition 4). Therefore (since we assume that $pr\mathcal{BPP} = pr\mathcal{P}$), the starting point of our argument is the hypothesis that CAPP can be solved in deterministic polynomial time.

The proof closely follows the celebrated proof strategy of Williams [Wil13], but with a different setting of parameters than is typically used in this proof strategy. Our goal is to prove that $NTIME[n^{O(f(n))}]$ is not contained in $\mathcal{P}/\text{poly}$, where f is some very small function (e.g., $f(n) = \log^*(n)$). Assuming towards a contradiction that $NTIME[n^{O(f(n))}] \subseteq \mathcal{P}/\text{poly}$, we will construct a non-deterministic algorithm for any $L \in NTIME[n^{f(n)}]$ that runs in time noticeably smaller than $n^{f(n)}$. This will contradict the non-deterministic time hierarchy [Coo72].

The key technical tool that allows us construct such an algorithm is called an "easy witness lemma" (see, e.g., [IKW02]). Loosely speaking, such a lemma asserts that under seemingly-unlikely hypotheses (such as $NTIME[n^{O(f(n))}] \subseteq \mathcal{P}/\text{poly}$), there exist *small* circuits that encode *large* witnesses in proof systems (e.g., for $NTIME[n^{f(n)}]$). Specifically, the proof of Theorem 1 crucially relies on a new easy witness lemma, which was very recently proved by Murray and Williams [MW17], and has significantly better parameters than previous lemmas. Given our hypothesis, the new lemma implies that for every $L \in NTIME[n^{O(f(n))}]$, every verifier V for L and every $x \in L$, there exists a circuit $P_x \in \mathcal{P}/\text{poly}$ that encodes a witness π_x such that $V(x, \pi_x)$ accepts (see Lemma 8 for a precise statement).[4] The point is that witnesses for the verifier V are a-priori of size

[3] For simplicity, in this paper we do not consider restricted classes of circuits (such as classes with constant depth or limited fan-out). However, the results in the paper extend to many restricted circuit classes in a straightforward way, using the PCP of Ben-Sasson and Viola (see discussion in [BV14]).

[4] A circuit $P_x : \{0,1\}^{\log(|\pi_x|)} \to \{0,1\}$ encodes a string π_x if for every $i \in [|\pi_x|]$ it holds that $P_x(i)$ is the i^{th} bit of π_x (equivalently, π_x is the truth-table of P_x). We mention that in the proof we will actually assume that $NTIME[n^{f(n)}]$ is not contained in $SIZE[n^{g(n)}]$, for some $g(n) \ll f(n)$, and deduce the existence of

$n^{O(f(n))}$, but the lemma asserts that (under the hypothesis) every $x \in L$ has a witness that can be concisely represented by a circuit of much smaller size.

Williams' idea is to use the existence of such "compressible" witnesses in order to construct a quick non-deteministic machine for any $L \in NTIME[n^{f(n)}]$. Specifically, fix any $L \in NTIME[t]$, where $t(n) = n^{f(n)}$. Also fix a PCP system for L with a verifier V that runs in time $t_V = \text{poly}(n, \log(t)) = n^{o(f(n))}$ and uses $\ell = O(\log(t))$ random bits. (For concreteness, we use the PCP of Ben-Sasson and Viola [BV14], but previous ones such as [BGH+05] also suffice for the proof.)

Given input $x \in \{0,1\}^n$, the non-deterministic machine M will first guess a witness for x, and then verify the witness using the verifier V. However, it will perform both tasks in a very efficient manner. First, instead of guessing a t-bit witness, the machine M will guess a (much-smaller) description of a circuit $P_x \in \mathcal{P}/\text{poly}$ that encodes a witness; if $x \in L$, then such P_x exists by the easy witness lemma.[5] Secondly, instead of directly using the verifier V to verify P_x, the machine will construct a circuit $C_x^{P_x}$ that, when given $r \in \{0,1\}^\ell$ as input, simulates the execution of V on x using randomness r when V is given oracle access to the witness P_x; the machine M will then use the CAPP algorithm on the circuit $C_x^{P_x}$ to determine whether the verifier is likely to accept x or to reject x. More specifically, the machine M acts as follows:

1. The machine non-deterministically guesses a circuit $P_x \in \mathcal{P}/\text{poly}$ (in the hope that P_x represents a proof for x acceptable by V).

2. The machine constructs a circuit $C_x^{P_x}$ that gets as input $r \in \{0,1\}^{\ell(n)}$, and simulates the execution of the verifier V on input x, randomness r, and with oracle access to the proof represented by P_x; that is, $C_x^{P_x}(r) = V^{P_x}(x, r)$.

3. The machine runs the CAPP algorithm on the circuit $C_x^{P_x}$, and outputs the decision of the algorithm.

Note that if $x \in L$, then for *some* guess of P_x it holds that $C_x^{P_x}$ has acceptance probability one, and thus the machine M will accept x. On the other hand, if $x \notin L$, then for *any* guess of P_x it holds that $C_x^{P_x}$ has low acceptance probability (corresponding to the soundness of the PCP verifier), and thus the machine M will reject x. Since P_x is of polynomial size and the verifier runs in time $t_V(n)$, the size of $C_x^{P_x}$ is at most $t_V(n) \cdot \text{poly}(n) = n^{o(f(n))}$. Therefore, the CAPP algorithm that gets $C_x^{P_x}$ as input also runs in time $n^{o(f(n))}$. The machine M thus decides L in non-deterministic time $n^{o(f(n))}$, which contradicts the non-deterministic time hierarchy theorem.

---

witness circuits of size $n^{g(n)}$. We ignore this minor issue in the high-level overview.

[5] Actually, to apply the easy witness lemma we consider the deterministic verifier V' that, when given input and a proof, enumerates the random coins of V and decides by a majority vote. This verifier runs in time $2^\ell \cdot t_V = n^{O(f(n))}$, and we can invoke the lemma since we assumed that $NTIME[n^{O(f(n))}] \subseteq \mathcal{P}/\text{poly}$.

4 Preliminaries

We assume familiarity with basic notions of complexity theory; for background see, e.g., [Gol08, AB09]. Throughout the paper, fix any standard model of a Turing machine (we need a fixed model since we discuss time-constructible functions). Whenever we refer to circuits, we mean non-uniform circuit families over the De-Morgan basis (i.e., AND/OR/NOT gates) with fan-in at most two and unlimited fan-out, and without any specific structural restrictions (e.g., without any limitation on their depth). Moreover, we consider some fixed standard form of representation for circuits, where the representation size is polynomial in the size of the circuit.

4.1 Circuit acceptance probability problem

We now formally define the circuit acceptance probability problem (or CAPP, in short); this well-known problem is also sometimes called Circuit Derandomization, Approx Circuit Average, and GAP-SAT or GAP-UNSAT.

Definition 3 *(CAPP). The* circuit acceptance probability problem *with parameters $\alpha, \beta \in [0,1]$ such that $\alpha > \beta$ (or (α,β)-CAPP, in short) is the following promise problem:*

- *The YES instances are (representations of) circuits that accept at least α of their inputs.*
- *The NO instances are (representations of) circuits that accept at most β of their inputs.*

We define the CAPP problem (i.e., omitting α and β) as the $(2/3, 1/3)$-CAPP problem.

It is well-known that CAPP is complete for $prBPP$ under deterministic polynomial-time reductions; in particular, CAPP can be solved in deterministic polynomial time *if and only if* $prBPP = prP$.

Proposition 4 *(CAPP is equivalent to $prBPP = prP$). The circuit acceptance probability problem can be solved in deterministic polynomial time* if and only if *$prBPP = prP$.*

For a proof of Proposition 4 see any standard textbook on the subject (e.g. [Vad12, Cor. 2.31], [Gol08, Exer. 6.14]). In Proposition 4 we considered the complexity of CAPP as a function of the input size, which is the size of the (description of the) circuit. However, following [Wil13], it can also be helpful to consider the complexity of CAPP as a function of both the circuit size m (which corresponds to the input size) and of the number v of input variables to the circuit. In this case, a naive deterministic algorithm can solve the problem in time $2^v \cdot \mathrm{poly}(m)$, whereas the naive probabilistic algorithm solves the problem in time $v \cdot \mathrm{poly}(m) \leq \mathrm{poly}(m)$.

4.2 Witness circuits and the new easy witness lemma of [MW17]

We now recall the definition of witness circuits for a proof system.

Definition 5 *(verifiers and witnesses). Let $t : \mathbb{N} \to \mathbb{N}$ be a time-constructible, non-decreasing function, and let $L \subseteq \{0,1\}^*$. An algorithm $V(x,y)$ is a* t-time verifier for L *if V runs in time at most $t(|x|)$ and satisfies the following: For all strings x it holds that $x \in L$ if and only if there exists* a witness y *such that $V(x,y)$ accepts.*

Definition 6 *(witness circuits). Let $t : \mathbb{N} \to \mathbb{N}$ be a time-constructible, non-decreasing function, let $w : \mathbb{N} \to \mathbb{N}$, and let $L \subseteq \{0,1\}^*$. We say that a t-time verifier V has* witness circuits of size w *if for every $x \in L$ there exists a witness y_x such that $V(x,y_x)$ accepts and there exists a circuit $C_{y_x} : \{0,1\}^{\log(|y_x|)} \to \{0,1\}$ of size $w(|x|)$ such that $C_{y_x}(i)$ is the i^{th} bit of y_x. We say that $NTIME[t]$* has witness circuits of size w *if for every $L \in NTIME[t]$, every t-time verifier for L has witness circuits of size w.*

In Definitions 5 and 6 we considered verifiers that are deterministic algorithms that get the witness as an explicit input. As outlined in Section 3, in the proof we will consider PCP verifiers (which are probabilistic algorithms, and only get oracle access to their witness). However, we will not consider witness circuits for these PCP verifiers, but rather for deterministic verifiers (with explicit inputs) that are derived from the PCP verifiers (see the proof of Theorem 10 for precise details).

Let us now state the new easy witness lemma of [MW17]. Loosely speaking, the lemma asserts that for any two functions $t(n) \gg s(n)$ with sufficient "gap" between them, if $NTIME[\mathrm{poly}(t)] \subseteq SIZE[s]$, then $NTIME[t]$ has witness circuits of size $\hat{s}$, where $\hat{s}(n) > s(n)$ is the function s with some "overhead". To more conveniently account for the exact parameters, we introduce some auxiliary technical notation:

Definition 7 *(sufficiently gapped functions). Let $\gamma, \gamma', \gamma'' \in \mathbb{N}$ be universal constants.*[6] *For any function $s : \mathbb{N} \to \mathbb{N}$, let $s' : \mathbb{N} \to \mathbb{N}$ be the function $s'(n) = (s(\gamma \cdot n))^{\gamma}$, and let $\hat{s} : \mathbb{N} \to \mathbb{N}$ be the function $\hat{s}(n) = (s'(s'(s'(n))))^{\gamma'}$. We say that two functions $s, t : \mathbb{N} \to \mathbb{N}$ are* sufficiently gapped *if both functions are increasing and time-constructible, and s' is also time-constructible, and $s(n) < 2^{n/\gamma}/n$, and $t(n) \geq (\hat{s}(n))^{\gamma''}$.*

Lemma 8 *(easy witnesses for low nondeterministic time [MW17, Lem. 4.1]). Let $s, t : \mathbb{N} \to \mathbb{N}$ be sufficiently gapped functions, and assume that $NTIME[O(t(n))^{\gamma}] \subset SIZE[s]$, where γ is the constant from Definition 7. Then, $NTIME[t]$ has witness circuits of size $\hat{s}$.*

4.3 A barrier for proving "$\mathcal{BPP} = \mathcal{P} \Longrightarrow \mathcal{P} \neq \mathcal{NP}$"

We note that it is impossible to prove the statement "if $\mathcal{P} = \mathcal{BPP}$ then $\mathcal{P} \neq \mathcal{NP}$" without *unconditionally* proving that $\mathcal{P} \neq \mathcal{NP}$.

Proposition 9 *(a barrier for "derandomization implies lower bounds" statements). If the conditional statement "$\mathcal{BPP} = \mathcal{P} \Longrightarrow \mathcal{P} \neq \mathcal{NP}$" holds, then $\mathcal{P} \neq \mathcal{NP}$.*

[6] Specifically, the values of these constants are $\gamma = e$ and $\gamma' = 2g$ and $\gamma'' = d$, where e, g, and d are the universal constants from Lemma 4.1 in [MW17].

Proof. Assume towards a contradiction that $\mathcal{P} = \mathcal{NP}$. Then, we have that $\mathcal{P} = \mathcal{BPP}$ (since $\mathcal{BPP}$ is contained in the polynomial-time hierarchy [Sip83, Lau83], and the hierarchy collapses to $\mathcal{P}$ if $\mathcal{P} = \mathcal{NP}$). However, we can now use the hypothesized conditional statement to deduce that $\mathcal{P} \neq \mathcal{NP}$, which is a contradiction. ■

5 Proof of Theorem 2

Our first step is to prove a parametrized "derandomization implies lower bounds" theorem. This theorem is obtained by using the proof strategy of Williams [Wil13] with general parameters, while leveraging the new easy witness lemma of Murray and Williams [MW17]. (In their original paper, Murray and Williams [MW17] considered two specific parameter settings.) We then prove Theorem 2 as a corollary.

Loosely speaking, in the following theorem we assume that CAPP can be deterministically solved in time $T(m,v)$, and deduce that for any two functions $t(n) \gg s(n)$ such that $T\Big(\text{poly}(n, \hat{s}(n), \log(t(n))),\ \log(t(n))\Big) \ll t(n)$ it holds that $NTIME[\text{poly}(t(n))]$ does not have circuits of size $s(n)$.

Theorem 10 *(derandomization implies lower bounds, with flexible parameters). There exist constants $c, c' \in \mathbb{N}$ and $\alpha < 1$ such that the following holds. For $T : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, assume that $(1, 1/3)$-CAPP on circuits of size m with at most v input variables can be solved in deterministic time $T(m,v)$. Let $s, t : \mathbb{N} \to \mathbb{N}$ be sufficiently gapped functions such that $s(n) > n$ and for some constant $\epsilon > 0$ it holds that*

$$T\Big((n \cdot \hat{s}(n) \cdot \log(t(n)))^c,\ \ \alpha \cdot \log(t(n))\Big) \leq t(n)^{(1-\epsilon)\cdot\alpha}\ .$$

Then, $NTIME[t(n)^{c'}] \not\subseteq SIZE[s(n)]$.

Proof. The starting point of the proof is the non-deterministic time hierarchy [Coo72]: For an appropriate function $t' = t'(n)$ (that will be determined in a moment), there exists a set $L \in NTIME[t']$ that cannot be decided by non-deterministic machines running in time $(t')^{1-\Omega(1)}$. Specifically, for a sufficiently small constant $\alpha > 0$, let $t'(n) = (t(n))^{(1-\epsilon/2)\cdot\alpha}$, and let $L \in NTIME[t'] \setminus NTIME\left[(t')^{\frac{1-\epsilon}{1-\epsilon/2}}\right]$.[7] Now, for a sufficiently large constant c', assume towards a contradiction that $NTIME[t(n)^{c'}] \subseteq SIZE[s(n)]$. Our goal is to construct a non-deterministic machine that decides L in time $(t')^{\frac{1-\epsilon}{1-\epsilon/2}}$, which will yield a contradiction.

To do so, consider the PCP verifier of [BV14] for L, denoted by V. On inputs of length n, the verifier V runs in time $\text{poly}(n, \log(t'(n)))$, uses $\ell = \log(t'(n)) + O(\log\log(t'(n)))$ bits of randomness, and has perfect completeness and soundness (much) lower than $1/3$. [8] Furthermore, using the hypothesis that $NTIME[t(n)^{c'}] \subseteq SIZE[s(n)]$ and the

[7] Such a function exists by standard non-deterministic time hierarchy theorems (e.g., [Coo72]), since $t'(n) > n^{\Omega(1)}$, which implies that the gap between t' and $(t')^{1-\Omega(1)}$ is sufficiently large.

[8] Note that the only upper-bound that we need on the number of oracle queries issued by V is the trivial bound given by the running time of V.

"easy witness lemma" (i.e., Lemma 8), for every $x \in L$ there exists a circuit $P_x \in SIZE[\hat{s}(n)]$ such that $\Pr_r[V^{P_x}(x,r) \text{ accepts}] = 1$. (We actually apply Lemma 8 to the deterministic verifier V' that enumerates the random coins of V, which runs in time $2^\ell \cdot \text{poly}(n, \log(t')) = \text{poly}(t') = \text{poly}(t)$. We can use the lemma since we assumed that $NTIME[t(n)^{c'}] \subseteq SIZE[s(n)]$, for a sufficiently large c'.)

Given input $x \in \{0,1\}^n$, the non-deterministic machine M acts as follows. The machine non-deterministically guesses a (description of a) circuit P_x of size $\hat{s}(n)$, and constructs a circuit $C_x^{P_x} : \{0,1\}^\ell \to \{0,1\}$ such that $C_x^{P_x}(r) = V^{P_x}(x,r)$. Then, the machine feeds the description of $C_x^{P_x}$ as input to the CAPP algorithm that exists by the hypothesis, and outputs the decision of the algorithm. By the properties of the PCP verifier and of the CAPP algorithm, if $x \in L$ then for some guess of P_x the machine will accept x, and if $x \notin L$ then for any guess of P_x the machine will reject x.

To conclude let us upper-bound the running-time of the machine M. The circuit $C_x^{P_x}$ has $\ell = \log(t') + O(\log\log(t')) < \alpha \cdot \log(t)$ input bits, and its size is $m(n) = \text{poly}(n, \log(t')) \cdot \hat{s}(n)$; thus, its representation size is $\text{poly}(m(n))$. Therefore, the circuit $C_x^{P_x}$ can be constructed in time $\text{poly}(m(n))$, and the CAPP algorithm runs in time $T(m(n), \ell)$. The total running-time of the non-deterministic machine M is thus at most $T\left((n \cdot \hat{s}(n) \cdot \log(t(n)))^c, \alpha \cdot \log(t)\right)$, for some constant c. By our hypothesized upper-bound on T, the running time of M is at most $t(n)^{(1-\epsilon)\cdot\alpha} = (t')^{\frac{1-\epsilon}{1-\epsilon/2}}$, which yields a contradiction. ■

We now prove Theorem 2 as a corollary of Theorem 10. Recall that in Theorem 2 we asserted that $pr\mathcal{BPP} = pr\mathcal{P}$ implies that $NTIME[n^{f(n)}] \not\subseteq \mathcal{P}/\text{poly}$ for "essentially" any super-constant function f. We now specify exactly what this means. In the proof, instead of proving that $NTIME[n^{f(n)}] \not\subseteq \mathcal{P}/\text{poly}$, we will actually prove the stronger statement $NTIME[n^{f(n)}] \not\subseteq SIZE[n^{g(n)}]$, where $g(n) \ll f(n)$ and $g(n) = \omega(1)$. Therefore, the proof works for any f such that a suitable g exists. We note in advance that this minor technical detail imposes no meaningful restrictions on f (see next).

Definition 11 *(admissible functions). We say that a function $f : \mathbb{N} \to \mathbb{N}$ is* admissible *if f is super-constant (i.e. $f(n) = \omega(1)$), and if there exists another super-constant function $g : \mathbb{N} \to \mathbb{N}$ that satisfies the following: The function g is super-constant, and $t(n) = n^{f(n)}$ and $s(n) = n^{g(n)}$ are sufficiently gapped, and $\hat{s}(n) = n^{o(f(n))}$.*

Essentially any increasing function $f(n) = \omega(1)$ such that $f(n) \leq n$ is admissible, where the only additional constraints that the admissibility condition imposes are time-constructibility of various auxiliary functions (we require t and s to be sufficiently gapped, which enforces time-constructibility constraints); for a precise (and tedious) discussion, see Appendix A. We can now formally state Theorem 2 and prove it:

Corollary 12 *(Theorem 2, restated formally). Assume that $(1, 1/3)$-CAPP can be solved by a deterministic algorithm with running time $T(m, v) \leq 2^{(1-\epsilon)\cdot v} \cdot \text{poly}(m)$, for some constant $\epsilon > 0$. Then, for every admissible function f there exists a set in $NTIME[n^{O(f(n))}] \setminus \mathcal{P}/\text{poly}$.*

Recall that by Proposition 4, if $prBPP = prP$, then there exists a deterministic algorithm for $(1, 1/3)$-CAPP running in time $\text{poly}(m)$ (i.e., polynomial in the size of the circuit). Therefore, Theorem 1 follows from Corollary 12.

Proof of Corollary 12. Since f is admissible, there exists a function g that satisfies the requirements of Definition 11. We use Theorem 10 with the functions $t(n) = n^{f(n)}$ and $s(n) = n^{g(n)}$.

Let us verify that the hypotheses of Theorem 10 hold. Since f is admissible we have that s and t are sufficiently gapped, and indeed we also have that $s(n) > n$. Also, for any constants $c \in \mathbb{N}$ and $\alpha < 1$ it holds that

$$\begin{aligned} T\Big((n \cdot \hat{s}(n) \cdot \log(t(n)))^c,\ \ \alpha \cdot \log(t(n))\Big) &< \text{poly}(\hat{s}(n)) \cdot \text{poly}\log(t(n)) \cdot t^{(1-\epsilon)\cdot\alpha} \\ &< n^{(1-\epsilon+o(1))\cdot f(n)}\ , \end{aligned}$$

which is $t(n)^{(1-\Omega(1))\cdot\alpha}$. (The first inequality relies on the hypothesis regarding T and on the fact that $\hat{s}$ is super-constant, and the second equality relies on the hypothesis that f is admissible, which implies that $\hat{s}(n) = n^{o(f(n))}$.)

Thus, Theorem 10 implies that there exists a set in $NTIME[n^{O(f(n))}] \setminus SIZE[n^{g(n)}]$. Since $g(n) = \omega(1)$, in particular this set does not belong to $\mathcal{P}/\text{poly}$. ■

Additional relaxations of the hypotheses in Theorem 10. Since the proof of Theorem 10 relies on the strategy of [Wil13], it is well-known that the hypothesis of the theorem can be further relaxed. We now mention two (known) relaxations, in the hope that they might be useful in some settings (e.g., when we are interested in proving lower bounds for restricted circuit classes; see Footnote 3).

First, we do not have to *unconditionally* assume that the CAPP algorithm exists: It suffices to assume that the algorithm exists *under the hypothesis that* $NTIME[t(n)^{c'}] \subseteq SIZE[s(n)]$. And secondly, since we are using the CAPP algorithm as a sub-routine of a non-deterministic machine, the CAPP algorithm itself can also be non-deterministic. (However, the non-determinism should help the algorithm accept every circuit with acceptance probability one, and reject every circuit with low acceptance probability; it is not a-priori clear how non-determinism can be useful for such a task.)

6 An alternative proof of Theorem 1

In this section we present an alternative proof of Theorem 1. The alternative proof does not rely on the work of Murray and Williams [MW17], but rather on a modification of the circuit lower bound of Santhanam [San09]. The alternative proof falls short of establishing the stronger Theorem 2, but it is simpler and more direct than the proofs presented in Section 5. (Recall that the proofs presented in Section 5 rely on the new "easy witness lemma" of Murray and Williams; the technical core of the proof of the latter in [MW17] is a strengthening of the lower bound of Santhanam.)

Let us present a high-level overview of the proof. Santhanam's theorem [San09] asserts that for every constant $k \in \mathbb{N}$ there exists a set that can be computed by a polynomial-time MA verifier that uses a single bit of non-uniform advice, but cannot be computed by circuits of size at most n^k. By scaling his original argument, one can show that there exists a set that can be computed in $MATIME[n^{\omega(1)}]/1$ (i.e., by an MA verifier that runs in arbitrarily-small super-polynomial time and uses one bit of non-uniform advice) but cannot be computed by circuits of size $n^{\omega(1)}$. More generally,

Theorem 13 *(a parametrization of Santhanam's [San09] lower bound). Let $s : \mathbb{N} \to \mathbb{N}$ be an increasing, super-linear and time-computable function such that $s(3n) \leq s(n)^3$. Then, for $t : \mathbb{N} \to \mathbb{N}$ such that $t(n) = \text{poly}(s(\text{poly}(s(n))))$ it holds that $MATIME[t]/1 \not\subseteq SIZE[s]$.*[9]

The proof of Theorem 13 imitates the original argument from [San09] using more general parameters. Since the proof requires no new significant ideas, we defer its presentation to Appendix B, and encourage the reader to think of Theorem 13 as a direct consequence of [San09].

Now, to prove Theorem 1, fix some "well-behaved" function $s(n) = n^{\omega(1)}$, and let t be as in Theorem 13. Theorem 13 asserts that there exists a set S in $MATIME[t]/1 \setminus SIZE[s]$. The proof relies on two observations:

1. *If $pr\mathcal{BPP} = pr\mathcal{P}$, then $MATIME[t]/1 \subseteq NTIME[\text{poly}(t)]/1$* (see Proposition 15 below). This relies on a straightforward derandomization of the MA verifier using a deterministic polynomial-time CAPP algorithm (i.e., relying on Proposition 4). Hence, the set S is in $NTIME[\text{poly}(t)]/1 \setminus SIZE[s]$.

2. *If $NTIME[\text{poly}(t)]/1 \not\subseteq SIZE[s]$, then $NTIME[\text{poly}(t)]$ (without the bit of advice) is also not contained in $SIZE[s]$* (see Proposition 16 below). This "elimination of advice" uses a simple reduction, which relies on a property of $NTIME[\text{poly}(t)]/1$ that does not hold for $MATIME[t]/1$ (i.e., the simple reduction follows through only in this step, after we "derandomized" $MATIME[t]/1$).[10]

Thus, by combining Theorem 13 and the two foregoing observations, we obtain a proof of Theorem 1: If $pr\mathcal{BPP} = pr\mathcal{P}$, then $NTIME[\text{poly}(t)] = NTIME[n^{\omega(1)}]$ is not contained in $SIZE[s] \supseteq \mathcal{P}/\text{poly}$.

6.1 Details for the alternative proof

We recall the standard definition of MA verifiers that receive non-uniform advice. In the following definition, t is the time bound for the verifier and ℓ is the length of the non-uniform advice.

[9] For a standard definition of the class $MATIME[t]/1$, see Definition 14.

[10] Specifically, we rely on the fact that for every non-deterministic machine M that uses non-uniform advice, and any fixing $\{a_n\}$ of the non-uniform advice for M, the machine M with advice $\{a_n\}$ either accepts or rejects every input (i.e., $M^{\{a_n\}}$ defines a "language" in $\{0,1\}^*$).

Definition 14 *(MA verifiers with non-uniform advice). For $t, \ell : \mathbb{N} \to \mathbb{N}$, a set $S \subseteq \{0,1\}^*$ is in $MATIME[t]/\ell$ if there exists a probabilistic machine V, called a* verifier*, such that the following holds: The verifier V gets input $x \in \{0,1\}^*$, and a witness $w \in \{0,1\}^*$, and an advice string $a \in \{0,1\}^*$, and runs in time $t(|x|)$; and there exists a sequence $\{a_n\}_{n\in\mathbb{N}}$ of advice such that $|a_n| = \ell(n)$ and:*

1. *For every $x \in S$ there exists $w \in \{0,1\}^{t(|x|)}$ such that $\Pr[V(x, w, a_{|x|}) = 1] \geq 2/3$.*
2. *For every $x \notin S$ and every $w \in \{0,1\}^{t(|x|)}$ it holds that $\Pr[V(x, w, a_{|x|}) = 1] \leq 1/3$.*

It is common to denote by $MATIME[t]$ the class $MATIME[t]/0$ (i.e., when the verifier receives no non-uniform advice). Note that $MA = \bigcup_{c\in\mathbb{N}} MATIME[n^c]$, and also note that the definition of MA does not change if we insist on perfect completeness (see, e.g., [Gol08, Exer. 6.12(2)]).

Let us first formally state and prove the two observations from above, and then combine them to obtain the theorem. The first observation is that if $pr\mathcal{BPP} = pr\mathcal{P}$ then we can derandomize MA verifiers that receive non-uniform advice.

Proposition 15 *(derandomization of MA with advice). If $pr\mathcal{BPP} = pr\mathcal{P}$, then for any $t : \mathbb{N} \to \mathbb{N}$ it holds that $MATIME[t]/1 \subseteq NTIME[\text{poly}(t)]/1$.*

Proof. Let S be a set in $MATIME[t]/1$, let V be the $MATIME[t]/1$ verifier for S, and let $\{a_n\}_{n\in\mathbb{N}}$ be a sequence of "good" advice that allows V to decide S. We want to construct a non-deterministic machine M that runs in time $\text{poly}(t)$ and decides S with one bit of non-uniform advice.

Given input $x \in \{0,1\}^n$ and advice a_n, the machine M guesses a witness $w \in \{0,1\}^{t(n)}$, and constructs a circuit $C = C_{V,x,w,a_n} : \{0,1\}^t \to \{0,1\}$ that gets as input $r \in \{0,1\}^{t(n)}$ and computes $V(x, w, r, a_n)$. Since V is a verifier for S and a_n is the "good" advice for V, if $x \in S$ then there exists w such that the acceptance probability of C is at least $2/3$, and if $x \notin S$ then for any w the acceptance probability of C is at most $1/3$. Since $pr\mathcal{BPP} = pr\mathcal{P}$ (and using Proposition 4), the machine M can distinguish between the two foregoing cases using a deterministic polynomial-time CAPP algorithm. The running time of the machine M is dominated by the running time of the latter algorithm, which is at most $\text{poly}(t(n))$. ∎

The second observation that we state and prove is that if $NTIME[t']/1$ is not contained in a non-uniform class of circuits, then $NTIME[O(t')]$ (i.e., without the non-uniform advice) is also not continaed in a (related) non-uniform class of circuits.

Proposition 16 *(eliminating the advice). Let $s, s', t : \mathbb{N} \to \mathbb{N}$ such that t is increasing, and $s'(n) = s(n-1)$. If $NTIME[t]/1 \not\subseteq SIZE[s]$, then $NTIME[O(t)] \not\subseteq SIZE[s']$.*

Proof. We actually prove the counter-positive statement: If $NTIME[O(t)] \subseteq SIZE[s']$, then $NTIME[t]/1 \subseteq SIZE[s]$. To see this, fix some $S \in NTIME[t]/1$, and let us construct a circuit of size s that decides S.

Let M be a t-time non-deterministic machine and let $\{a_n\}$ be a sequence of advice bits such that M correctly decides S when given advice $\{a_n\}$. Now, let S^{adv} be the set of pairs (x,σ), where $x \in \{0,1\}^*$ and $\sigma \in \{0,1\}$, such that M accepts x when given advice σ. Note that $S^{adv} \in NTIME[O(t)]$, since a non-deterministic machine that gets as input (x,σ) can simulate the machine M with advice σ and decide accordingly. Since we assumed that $NTIME[O(t)] \subseteq SIZE[s']$, there exists a circuit family $\{C_n\}$ of size s' such that for each n it holds that $C_n : \{0,1\}^{n+1} \to \{0,1\}$ decides $S^{adv} \cap \{0,1\}^n$. By hard-wiring the "correct" advice bit a_n in place of the last input bit into every C_n, we obtain a circuit family $\{C'_n\}$ of size at most s that decides S. ■

We can now combine the foregoing ingredients into an alternative proof of Theorem 1. The theorem that we obtain is similar in form to Theorem 10: Assuming a derandomization hypothesis (in this case, that $pr\mathcal{BPP} = pr\mathcal{P}$), and taking two functions t and s with sufficient "gap" between them, we deduce that $NTIME[t] \not\subseteq SIZE[s]$.

Theorem 17 *(Theorem 1, an alternative technical statement). Let $s : \mathbb{N} \to \mathbb{N}$ be an increasing, super-linear and time-computable function such that $s(3n) \leq s(n)^3$, and let $t : \mathbb{N} \to \mathbb{N}$ such that $t(n) = \text{poly}(s(\text{poly}(s(n))))$, for sufficiently large polynomials. Assume that $pr\mathcal{BPP} = pr\mathcal{P}$. Then, $NTIME[t] \not\subseteq SIZE[s]$.*

Proof. Let $s_0 = s^3$, and let $t_0 = \text{poly}(s_0(\text{poly}(s_0)))$, for sufficiently large polynomials. According to Theorem 13, there exists a set S in $MATIME[t_0]/1$ such that $S \notin SIZE[s_0]$. By Proposition 15, and relying on the hypothesis that $pr\mathcal{BPP} = pr\mathcal{P}$, it holds that $S \in NTIME[t_1]/1 \setminus SIZE[s_0]$, where $t_1 = \text{poly}(t_0)$. Using Proposition 16, it holds that $NTIME[t] \not\subseteq SIZE[s'_0]$, where $t = O(t_1) = \text{poly}(s(\text{poly}(s)))$ and $s'_0(n) = s_0(n-1)$. Finally, since s is increasing and $s(n) \leq s(n/3)^3$, we have that $s'_0(n) = s_0(n-1) \geq s_0(n/3) \geq s(n)$, and hence $NTIME[t] \not\subseteq SIZE[s]$. ■

One advantage of Theorem 17, in comparison to Theorem 10, is that the required "gap" between s and t (in order to conclude that $NTIME[t] \not\subseteq SIZE[s]$) depends on s in a milder way; in particular (and ignoring polynomial factors, for simplicity), in Theorem 17 we need that t will be larger than $s \circ s$, whereas in Theorem 10 we need that t will be larger than $\hat{s} \approx s \circ s \circ s$.

Acknowledgements

The author thanks his advisor, Oded Goldreich, for his close guidance in the research and writing process, and for very useful comments on drafts of the paper. The author also thanks Ryan Williams for several very helpful email exchanges, and in particular for stressing the point that the conclusion in the main theorem can be interpreted in two different ways (as explained in Section 1.1). The author is very grateful to Igor Oliveira for proposing the alternative proof of Theorem 1 (i.e., the one presented in Section 6), and for his permission to include the alternative proof in this paper.

References

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

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal of Computing*, 13(4):850–864, 1984.

[BGH+05] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Short pcps verifiable in polylogarithmic time. In *Proc. 20th Annual IEEE Conference on Computational Complexity (CCC)*, pages 120–134, 2005.

[BV14] Eli Ben-Sasson and Emanuele Viola. Short PCPs with projection queries. In *Proc. 41st International Colloquium on Automata, Languages and Programming (ICALP)*, pages 163–173. 2014.

[Coo72] Stephen A. Cook. A hierarchy for nondeterministic time complexity. In *Proc. 4th Annual ACM Symposium on Theory of Computing (STOC)*, pages 187–192, 1972.

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

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

[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)*, pages 220–229. 1999.

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[Lau83] Clemens Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, 17(4):215–217, 1983.

[MW17] Cody Murray and Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime: An easy witness lemma for NP and NQP. *Electronic Colloquium on Computational Complexity: ECCC*, 24:188, 2017.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

[San09] Rahul Santhanam. Circuit lower bounds for Merlin-Arthur classes. *SIAM Journal of Computing*, 39(3):1038–1061, 2009.

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

[TV07] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007.

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

[Wil13] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM Journal of Computing*, 42(3):1218–1244, 2013.

[Yao82] Andrew C. Yao. Theory and application of trapdoor functions. In *Proc. 23rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, 1982.

Appendix A Sufficient conditions for admissibility

The point of the current appendix is to show that essentially any increasing function $f(n) = \omega(1)$ such that $f(n) \leq n$ is admissible (in the sense of Definition 11).

Claim 18. *Let $f(n) = \omega(1)$ be any increasing function such that $f(n) \leq n$ for all n, and $t(n) = n^{f(n)}$ is time-constructible, and $s(n) = n^{\log(f(\log(n)))}$ is time-constructible, and $s'(n)$ is time-constructible. Then, f is admissible.*

Proof. Let $g(n) = \log(f(\log(n)))$ and let $s(n) = n^{g(n)}$. We need to verify that g is super-constant (which holds because f is super-constant), and that t and s are sufficiently gapped, and that $\hat{s}(n) = n^{o(f(n))}$. To see that t and s are sufficiently gapped, first note that both functions are increasing (since f is increasing, and hence g is also increasing) and are time-constructible, as is s' (we assumed time-constructibility in the hypothesis). Also note that $s(n) \leq n^{\log\log(n)} < 2^{n/\gamma}/n$.

Thus, it is left to verify that $\hat{s}(n) = n^{o(f(n))}$. The proof of this fact amounts to the following elementary calculation. First note that

$$s'(n) = (s(\gamma \cdot n))^{\gamma} = (\gamma \cdot n)^{\gamma \cdot \log(f(\log(\gamma \cdot n)))} < n^{\log^2(f(\log^2(n)))} .$$

Thus, for any function $k = k(n)$ and constant $c \geq 2$ such that $k(n) \leq \log^c(f(\log^{3c}(n)))$ (which in particular implies that $k(n) \leq \log^c(n)$), we have that

$$s'(n^k) < n^{k \cdot \log^2(f(\log^2(n^k)))} \leq n^{\log^{2c}(f(\log^{3c}(n)))} . \tag{A.1}$$

In particular, using Eq. (A.1) with $k(n) = \log^2(f(\log^2(n)))$ and $c = 2$, we deduce that $s'(s'(n)) < n^{\log^4(f(\log^6(n)))}$. Then, using Eq. (A.1) again with $k(n) = \log^4(f(\log^6(n)))$ and $c = 4$, we deduce that $s'(s'(s'(n))) < n^{\log^8(f(\log^{12}(n)))}$. Therefore, we have that $\hat{s}(n) < n^{\gamma' \cdot \log^8(f(\log^{12}(n)))} < n^{\gamma' \cdot \mathrm{poly}\log(f(n))} = n^{o(f(n))}$. ∎

Appendix B A parametrization of Santhanam's lower bound

In this appendix we detail the proof of Theorem 13. The proof imitates the original proof of Santhanam's main theorem in [San09, Thm. 1], but uses general parameters.

The first technical ingredient in Santhanam's proof is the $\mathcal{PSPACE}$-complete set of Trevisan and Vadhan [TV07]. We use this set, but instead of relying on the fact that the set is $\mathcal{PSPACE}$-complete, we will use padding to claim that the set is complete for $DSPACE[n^{\omega(1)}]$ under $n^{\omega(1)}$-time reductions.

Lemma 19 *(scaling the $\mathcal{PSPACE}$-complete set of [TV07] to $DSPACE[n^{\omega(1)}]$). There exists $L \subseteq \{0,1\}^*$ and a probabilistic polynomial-time oracle Turing machine M that satisfy the following:*

1. *Let $t : \mathbb{N} \to \mathbb{N}$ be a super-linear, time-computable function. Then, for every set $L' \in DSPACE[t]$ there exists a deterministic Turing machine $R_{L'}$ that runs in time $\mathrm{poly}(t)$ such that for every $x \in \{0,1\}^*$ it holds that $x \in L' \iff R_{L'}(x) \in L$.*
2. *On input $x \in \{0,1\}^*$, the machine M only issues queries of length $|x|$.*
3. *For any $x \in L$ it holds that $\Pr[M^{\mathbf{1}_L}(x) = 1] = 1$, where $\mathbf{1}_L : \{0,1\}^n \to \{0,1\}$ is the indicator function of $L_n = L \cap \{0,1\}^n$.*
4. *For any $x \notin L$ and any $f : \{0,1\}^n \to \{0,1\}$ it holds that $\Pr[M^f(x) = 0] \geq 2/3$.*

Proof. We take L to be the $\mathcal{PSPACE}$-complete set from [San09, Lem. 12], which is the same set constructed in [TV07]. Items (2) – (4) follow immediately from the original statement in [San09].[11] Item (1) follows since L is $\mathcal{PSPACE}$-complete, and using a padding argument. Specifically, for any t and L', consider the machine $R_{L'}$ that combines a reduction of L' to $L'' = \{(x, 1^t) : x \in L'\}$ with a reduction of L'' to L. The first reduction maps $x \mapsto (x, 1^t)$, and since $L'' \in \mathcal{PSPACE}$, there exists a second reduction of L'' to L that can be computed in time $\mathrm{poly}(t + |x|) < \mathrm{poly}(t)$ (the inequality is since t is super-linear). ■

Relying on Lemma 19, we now prove the Theorem 13.

Theorem 20 *(Theorem 13, restated). Let $s : \mathbb{N} \to \mathbb{N}$ be an increasing, super-linear and time-computable function such that $s(3n) \leq s(n)^3$. Then, for $t : \mathbb{N} \to \mathbb{N}$ such that $t(n) = \mathrm{poly}(s(\mathrm{poly}(s(n))))$ it holds that $MATIME[t]/1 \not\subseteq SIZE[s]$.*

Proof. Let $t_0 : \mathbb{N} \to \mathbb{N}$ such that $t_0(n) = s^4(n)$, and let $t_1 = \mathrm{poly}(t_0)$ and $t = t_2 = \mathrm{poly}(t_0(\mathrm{poly}(t_0)))$, for sufficiently large polynomials. Let L be the set from Lemma 19. Our goal is to prove that there exists a set in $MATIME[t_2]/1$ that is not in $SIZE[t^{1/4}]$. The proof proceeds by a case analysis.

[11] The original statement asserts that any $x \notin L$ is rejected with probability at least $1/2$ (rather than $2/3$), but this probability can be amplified to $2/3$ using standard error-reduction.

Case 1: $L \in SIZE[t_0]$**.** By a standard diagonalization argument, there exists a set $L^{\mathtt{diag}} \in DSPACE[t_1] \setminus SIZE[t_0]$. [12] Our main goal now will be to prove that $DSPACE[t_1] \subseteq MATIME[t_2]$, which will imply that $L^{\mathtt{diag}} \in MATIME[t_2] \setminus SIZE[t_0]$. (Indeed, in this case we are proving a stronger result, since the MA verifiers do not need advice, and since the circuits are of size t_0 rather than $s = t_0^{1/4}$.)

To do so, let $L' \in DSPACE[t_1]$, and consider the following MA verifier for L'. On input $x \in \{0,1\}^n$, the verifier computes $x' = R_{L'}(x)$, where $R_{L'}$ is the machine from Lemma 19. Note that $n' = |x'| \leq \mathrm{poly}(t_1)$, and that $x \in L' \iff x' \in L$. Now, the verifier parses the witness $w \in \{0,1\}^{\mathrm{poly}(t_0(n'))}$ as a description of a circuit $C : \{0,1\}^{n'} \to \{0,1\}$ of size $t_0(n')$, and runs the machine M from Lemma 19 on input x', while answering each oracle query of M using the circuit C.

Note that, since $L \in SIZE[t_0]$, there exists a circuit C over n' input bits of size $t_0(n')$ that correctly computes L on inputs of length n'. Therefore, by Lemma 19, when $x \in L'$ there exists a witness such that the verifier accepts x with probability one; and the verifier rejects any $x \notin L'$, with probability at least $2/3$, regardless of the witness. The total running time of the verifier is dominated by the time it takes to simulate M using the circuit C, which is at most $\mathrm{poly}(n') \cdot \mathrm{poly}(t_0(n')) \leq t_2(n)$.

Case 2: $L \notin SIZE[t_0]$**.** In this case we show an explicit set $L^{\mathtt{pad}}$, which will be a padded version of L, such that $L^{\mathtt{pad}}$ can be decided in $MATIME[t_2]$ with one bit of advice, but cannot be decided by circuits of size $s = t_0^{1/4}$. To do so, let $\phi_L : \mathbb{N} \to \mathbb{N}$ be such that $\phi_L(n)$ is the minimum circuit size for $L_n = L \cap \{0,1\}^n$. Also, for any integer m, let $y(m)$ be the largest power of two that is smaller than m (i.e., $y(m) = \max_{\ell < \log(m)}\{2^\ell\}$), and let $n(m) = m - y(m)$. We define the set $L^{\mathtt{pad}}$ as follows:

$$L^{\mathtt{pad}} = \left\{ (x, 1^y) : \; x \in L, \text{ and } |x| = n(|x| + y), \text{ and } t_0(|x| + y) \leq \phi_L^3(|x|) < t_0(|x| + 2y) \right\}.$$

Let us first see that $L^{\mathtt{pad}}$ cannot be decided by circuits of size $t_0^{1/4}$. Assume towards a contradiction that there exists a circuit family $\{C_m\}$ of size $t_0^{1/4}$ that decides $(L^{\mathtt{pad}})_m = L^{\mathtt{pad}} \cap \{0,1\}^m$ correctly for every m. Since $L \notin SIZE[t_0]$, there exists an infinite set $I \subseteq \mathbb{N}$ such that for every $n \in I$ it holds that $\phi_L(n) > t_0(n)$. For a sufficiently large $n \in I$, we will construct a circuit $C'_n : \{0,1\}^n \to \{0,1\}$ of size less than $\phi_L(n)$ that computes L_n, which yields a contradiction to the definition of φ_L.

Specifically, consider the circuit $C'_n : \{0,1\}^n \to \{0,1\}$ that acts as follows. Let y be a power of two such that $t_0(n + y) \leq \phi_L^3(n) < t_0(n + 2y)$; there exists such a y since $\phi_L^3(n) > t_0^3(n) \geq t_0(3n) \geq t_0(n + 2^{\lceil \log(n) \rceil})$. [13] The value of this y is hard-coded into C'_n. Given $x \in \{0,1\}^n$, the circuit C'_n first pads x with 1^y, and then simulates the circuit C_m on $(x, 1^y)$ (where $m = |x| + y$) and outputs $C_m(x, 1^y)$. By the definition of $L^{\mathtt{pad}}$ it holds

[12] Specifically, let $\{C_n\}_{n \in \mathbb{N}}$ such that for every sufficiently large $n \in \mathbb{N}$ it holds that C_n is the lexicographically-first circuit over n bits of size at most $t_0^2(n)$ whose truth-table cannot be computed by a circuit of size $t_0(n)$. Let $L^{\mathtt{diag}} = \{x : C_{|x|}(x) = 1\}$. By definition, $L^{\mathtt{diag}} \notin SIZE[t_0]$, and the proof that $L^{\mathtt{diag}} \in DSPACE[t_1]$ follows the well-known proof idea that is used, e.g., in the proof of Kannan's theorem.

[13] And since the intervals $[t_0^{1/3}(n + y), t_0^{1/3}(n + 2y))$ for y's that are power of two cover $[t_0^{1/3}(3n), \infty)$.

that C'_n correctly computes L_n. The size of C'_n is dominated by the size of C_m, and is thus at most $O(t_0^{1/4}(n+y)) = o(t_0^{1/3}(n+y))$. Since $t_0^{1/3}(n+y) \leq \phi_L(n)$ and n is sufficiently large, the size of C'_n is less than $\phi_L(n)$, which yields a contradiction.

Let us now see that $L^{\mathtt{pad}}$ can be decided by an MA verifier that runs in time t_2 and uses one bit of advice. Given an input z of length m, the advice will say whether or not $(L^{\mathtt{pad}})_m \neq \emptyset$; if the advice is zero, the verifier immediately rejects. Otherwise, the verifier computes $n = n(m)$ and $y = y(m)$, and parses the input z as $(x, 1^y)$ where $|x| = n$ (if the verifier fails to parse the input, it immediately rejects). The verifier parses the witness $w \in \{0,1\}^{\mathrm{poly}(t_0(n+2y))}$ as a circuit $C : \{0,1\}^n \to \{0,1\}$ of size at most $t_0(n+2y)$, and emulates the machine M from Lemma 19 for L on input x, answering each oracle query of M using the circuit C. The verifier outputs the decision of M.

Since $\phi_L(n) < t_0(n+2y)$, there exists a circuit C of size at most $t_0(n+2y)$ that computes L_n. For any $z \in L^{\mathtt{pad}}$, when the witness represents this circuit, the verifier accepts z with probability one. Also, for any $z \notin L$, the verifier rejects x with probability $2/3$, regardless of the witness. Finally, note that the running time of the verifier is dominated by the time that it takes to run the machine M while simulating the oracle answers, which is at most $\mathrm{poly}(n) \cdot \mathrm{poly}(t_0(2m)) \leq t_2(m)$. ■

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il