Electronic Colloquium on Computational Complexity, Report No. 118 (2026)

Near-Maximum Circuit Lower Bounds for Exponential Time with Merlin-Arthur Queries

Hanlin Ren*
IAS
h4n1in.r3n@gmail.com

Ryan Williams†
MIT
rrw@mit.edu

July 10, 2026

Abstract

We prove a near-maximum $(2^n/n)$ circuit lower bound for the complexity class $\mathsf{E}^{\mathsf{prMA}}/_1$, corresponding to exponential time with access to a promise-MA oracle and one bit of advice. Our proof incorporates the iterative win-win paradigm (Chen–Lu–Oliveira–Ren–Santhanam, FOCS'23), the reduction from the Range Avoidance problem to circuit lower bounds (Jeřábek, *Ann. Pure Appl. Log. '04*; Korten, FOCS'21), and the PCP theorem. Crucial to our proof is the analysis of the complexity class $\mathsf{P}^{\mathsf{NP}}\left[\begin{smallmatrix}\#\text{rounds} = r\\ \text{length} = s\end{smallmatrix}\right]$, which is P^{NP} with $r(n)$ adaptive rounds of NP queries, where each NP query has witness length $s(n)$.

Contents

*Hanlin Ren is supported by the Massive Dynamics Member Fund at the Institute for Advanced Study.

†This work was initiated while visiting the Institute for Advanced Study, Princeton, NJ. This material is based upon work supported by the National Science Foundation under grants DMS-2424441 (at IAS) and CCF-2420092 (at MIT).

ISSN 1433-8092

1 Introduction

A simple counting argument dating back to Shannon [Sha49] shows that a uniformly random 2^n-bit truth table describes a *near-maximum hard* function requiring Boolean circuits of size $\Omega(2^n/n)$ on inputs of length n, with high probability: there are simply too many possible functions to be covered by all circuits of size $2^n/(10n)$. Significant research has gone into understanding how efficiently such a hard function can be constructed, and this problem is central to complexity theory. For example, if there are functions in NP with near-maximum hard (or even superpolynomially hard) finite slices, then $\mathsf{P} \neq \mathsf{NP}$. If there are near-maximum hard (or even exponentially-hard) functions in $\mathsf{E} = \mathsf{TIME}[2^{O(n)}]$, then $\mathsf{P} = \mathsf{BPP}$ [IW97]. However, the *non-uniformity* of circuit families makes the uniform construction of hard functions apparently extremely difficult. It has been known for decades that the (huge) class $\Sigma_3\mathsf{E} = \Sigma_3\mathsf{TIME}[2^{O(n)}]$ contains a function of near-maximum circuit size [Kan82] and that the complexity can be reduced slightly to $\mathsf{E}^{\Sigma_2\mathsf{P}}$ [MVW99].

In the last few years, substantial progress has been made on understanding the uniform complexity of near-maximum hard functions. Chen, Hirahara, and Ren [CHR24] showed that $\Sigma_2\mathsf{E}$ (indeed, even $\mathsf{S}_2\mathsf{E}/_1$) contains near-maximum hard functions, resolving a 40-year open problem, using a novel win-win paradigm for constructing hard functions. Li [Li24] gave a drastically simplified proof that also improved the lower bound in several aspects; see [CHLR26] for a combined exposition. More recently, Chen, Li, and Liang [CLL25] showed that exponential time Arthur-Merlin with subexponential advice ($\mathsf{AM}_{\mathsf{EXP}}/_{2^{n^\varepsilon}}$) contains near-maximum hard functions.

These recent results raise the question: can the complexity of exponentially hard functions be reduced even further? Natural candidates for such functions are those in complexity classes where we already know some nontrivial circuit complexity lower bounds, but the known lower bounds are much smaller than $\Omega(2^n/n)$ size. For example, it has been known since [BFT98] that the class *exponential-time Merlin Arthur* ($\mathsf{MA}_{\mathsf{EXP}}$) has functions without polynomial-size circuits. Can the complexity of the hard function in [CLL25] be reduced from Arthur-Merlin down to Merlin-Arthur (non-interactive proofs), and can the 2^{n^ε} size advice be removed or made much shorter? (Recall that Merlin-Arthur can always be simulated by Arthur-Merlin with polynomial overhead [Bab85], but showing the converse is a wide open problem.) An additional motivation for proving circuit lower bounds for classes related to Merlin-Arthur instead of Arthur-Merlin is their connections to *easy witness lemmas* [IKW02] and *the algorithmic method* [MW20; Che25; Che23].

Although Merlin-Arthur complexity classes have long been known to admit non-trivial circuit lower bounds via non-relativizing techniques [BFT98; San09], it has been an outstanding open problem to find a Merlin-Arthur-type complexity class which contains a near-maximum hard function. The proof of [BFT98] that $\mathsf{MA}_{\mathsf{EXP}} \not\subset \mathsf{P}/_{\mathrm{poly}}$ generalizes to show a "half-exponential" size lower bound: a lower bound for size functions h such that $h(h(\mathrm{poly}(n))) \leq 2^n$. Their lower bound provably does not relativize: they give an oracle relative to which $\mathsf{MA}_{\mathsf{EXP}}$ has small circuits. However, for the slightly larger class $\mathsf{E}^{\mathsf{prMA}}$ (exponential time with queries to a promise version of Merlin-Arthur), there *is* a relativizing proof of a half-exponential circuit complexity lower bound (see **??**) based on [CR11]. Half-exponential functions are a typical barrier for circuit size lower bounds: prior to the breakthrough of [CHR24], the best circuit size lower bound for $\Sigma_2\mathsf{E}$ was only half-exponential [MVW99].

1.1 Our Results

Recall that prMA is the class of promise problems solvable in polynomial time with a Merlin-Arthur protocol. In particular, a promise problem [ESY84] is given by a pair $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ where Π_{YES} and Π_{NO} are disjoint but do not necessarily cover all possible inputs, and the Merlin-Arthur protocol is only judged on its behavior for inputs in $\Pi_{\mathrm{YES}} \cup \Pi_{\mathrm{NO}}$ (we say such inputs "satisfy the promise").

Our main result is that there is a near-maximum hard function in $\mathsf{E}^{\mathsf{prMA}}/_1$, the class of problems computable in exponential time with oracle access to a prMA language and 1 bit of advice.

Theorem 1.1 (Main Result). *There is a function in* $\mathsf{E}^{\mathsf{prMA}}/_1$ *that requires circuits of* $\Omega(2^n/n)$ *size.*

In fact, the hard function given in **??** has the additional nice property that, on the correct advice bit, the exponential-time algorithm *only* queries the prMA oracle on queries that satisfy the promise of the prMA problem. Such reductions are called *smart reductions* [GS88; Gol06]; hence, our lower bound **??** holds for a function in the potentially smaller class $\mathsf{smart\text{-}E}^{\mathsf{prMA}}/_1$. We refer the reader to **??** for a more formal definition.

Prior work from about 15 years ago hinted that a result like **??** is possible. Aydınlıoğlu, Gutfreund, Hitchcock, and Kawachi [AGHK11] proved that if prAM is in polynomial time with an NP oracle ($\mathsf{prAM} \subseteq \mathsf{P}^{\mathsf{NP}}$) then E^{NP} contains near-maximum hard functions. Roughly speaking, they showed how to construct a hard truth table by using a prAM oracle to approximately count how many circuits agree with the current truth table prefix, allowing them to eliminate circuits efficiently. However, they also pointed out that their result does *not* imply an unconditional circuit lower bound for $\mathsf{E}^{\mathsf{prAM}}$: the hard function they produce *depends on oracle answers outside of the promise*. In that sense, their hard function is not single-valued as a computation on oracles. Indeed, they explicitly emphasized that $\mathsf{E}^{\mathsf{prAM}} \not\subset \mathsf{SIZE}[\Omega(2^n/n)]$ remained an interesting open problem. (This question is no longer open anymore: the recent result of [CHR24; Li24] shows that $\mathsf{S_2E} \not\subseteq \mathsf{SIZE}[2^n/n]$, and since $\mathsf{S_2E} \subseteq \mathsf{E}^{\mathsf{prAM}}$ [CR11], we have $\mathsf{E}^{\mathsf{prAM}} \not\subseteq \mathsf{SIZE}[2^n/n]$ as well.)

Fixed-polynomial size lower bounds for *polynomial time* with a prMA oracle have been known for many years. Santhanam [San09] proved that $\mathsf{MA}/_1$ and prMA require n^k-size circuits for all k; scaling up his result to $\mathsf{MA}_{\mathsf{EXP}}$ only implies a half-exponential size lower bound. In their work on improving Karp-Lipton collapses, Chakaravarthy and Roy [CR11] prove that $\mathsf{NP} \subseteq \mathsf{P/poly}$ implies $\mathsf{PH} = \mathsf{P}^{\mathsf{prMA}}$. This collapse immediately implies [Kan82] that for every k, there is a function in $\mathsf{P}^{\mathsf{prMA}}$ not in $\mathsf{SIZE}[n^k]$. Their resulting $\mathsf{P}^{\mathsf{prMA}}$ machine asks non-promise queries (so it is not "smart"), but the machine still gives consistent accept/reject answers regardless of how queries outside of the promise are answered. However, scaling their result up to $\mathsf{E}^{\mathsf{prMA}}$ also only yields a half-exponential size lower bound (see **??**). To circumvent these half-exponential barriers, we need a totally different approach.

A New Algorithm for Range Avoidance. Like previous works [CHR24; Li24; CLL25], our circuit lower bounds are proved by designing new algorithms for solving the *Range Avoidance* (AVOID) problem, introduced by [KKMP21]. In this problem, we are given a circuit C which has n inputs and $n+1$ outputs, and the task is to find a y of $n+1$ bits which is not in the range of C. This is a fundamental meta-complexity problem which defines a *formal task* associated with counting arguments like the one described in the first paragraph of this paper: if C takes in the description of a circuit of size $2^n/(10n)$ and outputs a 2^n-bit truth table, then solving AVOID on C directly corresponds to finding a near-maximum hard function. Range Avoidance has seen a flurry of work in recent years (for example [Kor21; GLW22; RSW22; ILW23; CHLR23; CL24; KP24] and Korten's recent survey [Kor25]).

Theorem 1.2. *For every constant* $d \geq 1$*, there is a deterministic polynomial-time algorithm* $\mathcal{A}$ *with access to a* prMA *oracle and one bit of advice* $\{\alpha_n\}_{n\in\mathbb{N}}$ *such that the following holds for infinitely many integers* n*. For every circuit* $C : \{0,1\}^n \to \{0,1\}^{n+1}$ *of size at most* n^d*,* $\mathcal{A}(C, \alpha_n)$ *is smart (i.e., only makes queries to the* prMA *oracle that is indeed in the promise) and outputs a string* $y \in \{0,1\}^{n+1} \setminus \mathrm{Range}(C)$*.*

Remark 1.3. As discussed in [CHR24, Section 1.2.1], to obtain circuit lower bounds, we need to design *single-valued* algorithms for AVOID. Roughly speaking, an algorithm $\mathcal{A}$ for a search problem is *single-valued*, if for each instance C, there is a canonical output y_C that is outputted by every (or most) computational paths of $\mathcal{A}$. We note that every smart $\mathsf{FP}^{\mathsf{prMA}}$ algorithm is necessarily single-valued, since the algorithm is deterministic and only makes queries inside the promise. (In contrast, if we drop the smartness requirement, then an $\mathsf{FP}^{\mathsf{prMA}}$ algorithm may not be single-valued, as the answer can depend on the values returned by the oracle on queries outside the prMA promise.)

1.2 Techniques

At a high level, we follow the "iterative win-win" method of [CLO+23] for pseudodeterministic constructions. Recall that a pseudodeterministic algorithm [GG11] is a randomized algorithm that on an input x, with high probability outputs a *canonical* solution $f(x)$ that does not depend on its internal randomness. (In other words, they are single-valued BPP or ZPP algorithms.)

The core ingredient underlying the iterative win-win method is a *hardness-randomness tradeoff* with *uniform reconstruction*. We view our main technical contribution as finding the right version of this ingredient that is tailored to the circuit lower bound we are proving.

1.2.1 The Iterative Win-Win Method

Let $\Pi = \{\Pi_n\}$ be a family of dense properties, i.e., $\Pi_n \subseteq \{0,1\}^n$ and $|\Pi_n| \geq 2^n/\mathrm{poly}(n)$. (In [CLO+23], Π is the set of prime numbers, while in our paper, Π is the set of hard truth tables.) Our goal is to design an efficient algorithm that successfully outputs an element of Π_n for infinitely many input lengths n.

Consider two input lengths n and N, where $n < N < \mathrm{poly}(n)$. Imagine that there is a deterministic "brute force" algorithm BF_n that successfully finds an element of Π_n in time T. (The relationship between T and n may be arbitrary, since we start from the *real* brute force algorithm where $T = \exp(n)$, and will reduce T to be gradually closer to $\mathrm{poly}(n)$ as the win-win argument proceeds.) On such pair of input lengths, we perform a win-win argument of the following form:

(Win) If Π_N is "helpful" for speeding up the brute force algorithm BF_n, then we obtain an algorithm in $\mathrm{poly}(N)$ time that correctly simulates BF_n. In [CLO+23], this algorithm is randomized, which means that we obtain a randomized algorithm in $\mathrm{poly}(N)$ time that finds an element in Π_n. Moreover, since the randomized algorithm always outputs the output of BF_n, it is also a *pseudodeterministic* algorithm. In our work, this algorithm will run in $\mathsf{smart\text{-}FP}^{\mathsf{prMA}}$.

(Improve) If Π_N is "not helpful" for speeding up BF_n, then we (somehow) obtain an algorithm $\mathcal{A}_N$ for finding an element of Π_N with time complexity $T' \leq \mathrm{poly}(T)$. Since $T' \leq 2^{O(n)} \ll 2^{O(N)}$, we make progress by bringing T' closer to $\mathrm{poly}(N)$ than $2^{O(N)}$. We then proceed to the next level of the win-win argument by replacing n with N, setting BF_N to be the "better" brute force algorithm $\mathcal{A}_N$, and replacing N with another input length $N' := \mathrm{poly}(N)$.

Choosing the parameters carefully, we can guarantee that at some level of the win-win argument, we will have $T \leq \mathrm{poly}(n)$ and we can afford to run BF_n (details omitted).

To summarize, for the sake of iterative win-win, we need that for every possible combination of BF_n and Π_N, either Π_N is "helpful" for speeding up BF_n, or BF_n implies an algorithm for finding an element in Π_N in time $\mathrm{poly}(T) = \mathrm{poly}(|\mathsf{BF}_n|)$. This sounds familiar to a *hardness-randomness tradeoff*.

The hardness-randomness tradeoff. It is well-known [NW94; IW97] that circuit lower bounds imply derandomization. More precisely, letting $L \in \mathsf{BPP}$, there is a deterministic machine $\mathrm{IW97}(x, f)$ that takes an instance x and a hard truth table f, runs in $\mathrm{poly}(|x|, |f|)$ time, and outputs $L(x)$. Treating the problem of finding an element in Π_N as L and (the computational history of) BF_n as f, we can see that:[1]

- either the circuit complexity of BF_n is small (for reasons that will be evident later, we will say that Π_N is "helpful" for speeding up BF_n in this case);
- or BF_n implies a faster ($\mathrm{poly}(T)$-time) algorithm for constructing Π_N by invoking IW97.

We note that in our paper, since Π_n is the set of hard truth tables, instead of [NW94; IW97], we need a hardness-randomness tradeoff for solving AVOID. Looking ahead, we will use the Jeřábek–Korten reduction ([Jeř04; Kor21], see also **??**).

[1] Although "finding an element in Π_N" is technically not a (decisional) BPP problem, the hardness-randomness tradeoffs in [NW94; IW97] can still solve it by iterating through a complexity-theoretic PRG constructed from the hard truth table.

Uniform reconstruction. We still need to justify that if the circuit complexity of BF_n is small, then Π_N is helpful for speeding up BF_n. At first glance, this seems to follow from the *reconstruction argument* of [NW94]: Given a truth table f and an adversary Π_N breaking the PRG constructed from f, we can efficiently compute a small circuit for f. However, the reconstruction procedure of [NW94] requires some advice bits depending on f, thus we do not obtain a genuine speed-up algorithm for BF_n. (This is also the same reason that [NW94; IW97] requires a lower bound against *non-uniform circuits* instead of a *uniform probabilistic* (BPP) algorithm.)

For the task of constructing prime numbers, [CLO+23] gets around this issue using the hardness-randomness tradeoff with *uniform reconstruction* in [CT21]. Applying a highly non-trivial arithmetization of BF_n in [GKR15], the reconstruction algorithm can compute the needed advice bits "on the fly". We omit the details here, but the end result is that there is indeed a BPP algorithm (without any non-uniform advice) that computes (each output bit of) BF_n given oracle access to Π_N; in this sense, we say that Π_N is "helpful" for speeding up BF_n.

1.2.2 Towards Solving Range Avoidance

As discussed before, we use the Jeřábek–Korten reduction as the hardness-randomness tradeoff. In what follows, we let T be the length of the "hard truth table" f, let n be the size of the AVOID instance G we are solving, and think of T as an arbitrary parameter between $\mathrm{poly}(n)$ and $2^{O(n)}$.

Theorem 1.4 (Informal; [Jeř04; Kor21])**.** *There is an algorithm* $\textsf{Jeřábek–Korten}(G, f)$ *running in deterministic* $\mathrm{poly}(n, T)$ *time with access to an* NP *oracle such that, if the circuit complexity of* f *is* $\geq \mathrm{poly}(n, \log T)$*, then* $\textsf{Jeřábek–Korten}(G, f)$ *solves* AVOID *on the instance* G*.*

This is indeed a hardness-randomness tradeoff: Given a truth table f with large enough circuit complexity (and arbitrary length T), Jeřábek–Korten solves a derandomization task (namely AVOID). However, as discussed before, we need a hardness-randomness tradeoff with *uniform reconstruction*. That is, imagine that $\textsf{Jeřábek–Korten}(G, f)$ *fails* to solve AVOID on G, in what complexity class can we compute f *uniformly*?

A naïve attempt. Suppose that f is (some encoding of) the computational history of a $\mathsf{TIME}[T]^{\mathsf{NP}}$ computation. If $\textsf{Jeřábek–Korten}(G, f)$ fails to solve AVOID on G, then by **??**, f admits a small circuit. Our goal here is to compute f in a uniform complexity class. Now, this sounds familiar to the *Karp–Lipton theorem* [KL80].

The Karp–Lipton theorems are a family of theorems showing that non-uniform collapses imply uniform collapses. For example:

> If $\mathsf{EXP} \subset \mathsf{P}/_{\mathrm{poly}}$ then $\mathsf{EXP} = \Sigma_2\mathsf{P}$ ([KL80], attributed to Albert Meyer).

Such theorems are usually proved by exploiting small circuits which encodes the computational history of a complex computation. Indeed, the $\Sigma_2\mathsf{P}$ algorithm in the above theorem guesses a small circuit encoding the computational history of the EXP computation and uses a universal quantifier to verify that all of the exponentially many computation steps are executed correctly.

We apply a similar idea in our case. Every problem in $\mathsf{TIME}[T]^{\mathsf{NP}}$ reduces to finding the lexicographically smallest[2] satisfying assignment of a size-$\mathrm{poly}(T)$ (satisfiable) 3-CNF [Kre88] and we can define the computational history f to simply be this lex-first satisfying assignment. If f has circuit complexity at most $s = \mathrm{poly}(n, \log T)$, then we can compute each bit of f in $\Sigma_4\mathsf{TIME}[\mathrm{poly}(s)]$: use an existential quantifier to guess a size-s circuit C (whose truth table is supposedly f), use a universal quantifier to guess a size-s circuit C' (competing against C), use an existential quantifier to guess the smallest index i such

[2]We actually consider lexicographically *largest* satisfying assignment in our formal proof, but it is easy to see that they are equivalent.

that $C(i) < C'(i)$, and finally use a universal quantifier to verify that for every $j < i$, $C(j) = C'(j)$. (We have omitted many details here such as verifying C and C' encode satisfying assignments.) It follows that we obtain a reconstruction algorithm in $\Sigma_4\mathsf{P}$, which implies a near-maximum circuit lower bound for $\Sigma_4\mathsf{E}$. Unfortunately, this is worse than the near-maximum circuit lower bound proved by simple diagonalization [Kan82; MVW99], which results in a hard function in no more than $\Sigma_3\mathsf{E}$.

We remark that by applying a suitable PCP [AS98; ALM+98] on the 3-CNF produced by [Kre88] and using an efficient comparator for Reed–Muller codewords [Hir15], it is possible to obtain an $\mathsf{S}_2\mathsf{P}$ reconstruction algorithm. However, this still only results in a near-maximum circuit lower bound for $\mathsf{S}_2\mathsf{E}$, which is no better than previous results [CHR24; Li24].[3]

The crucial insight. We observe that the Jeřábek–Korten procedure is weaker than a full-fledged P^{NP} algorithm. In particular, it only makes $r := \mathrm{poly}(n, \log T)$ many adaptive rounds of NP queries (see **??**). It can be shown, using nearly the same argument as [Kre88], that any $\mathsf{TIME}[T]^{\mathsf{NP}}$ computation with only r adaptive rounds of NP queries reduces to finding a satisfying assignment of a size-$\mathrm{poly}(T)$ 3-CNF with the *lexicographically smallest length-r' prefix*, where $r' := r \log T$. Since $r' \le \mathrm{poly}(n, \log T)$ is small, our uniform reconstruction algorithm can afford to enumerate these r' bits one by one.

In particular, assuming the computational history has circuit complexity at most s, the uniform reconstruction can be done in $\mathsf{TIME}[\mathrm{poly}(r', s)]^{\Sigma_2\mathsf{P}}$. We maintain the current prefix pfx which is initially empty; after r' rounds, pfx should be equal to the lexicographically smallest length-r' prefix of any satisfying assignment. One can test whether there is a satisfying assignment with (circuit complexity $\le s$ and) pfx as a prefix in $\Sigma_2\mathsf{P}$: use an existential quantifier to guess a size-s circuit C and use a universal quantifier to verify that C is a satisfying assignment with prefix pfx. In each round, if there is a satisfying assignment with (pfx concatenated with 0) as a prefix, then we append 0 to pfx; otherwise we append 1.

This gives a uniform reconstruction procedure in $\mathsf{FP}^{\Sigma_2\mathsf{P}}$. Applying a standard PCP theorem [AS98; ALM+98] to the 3-CNF, one can verify whether C is a satisfying assignment by random coin tosses, which improves the complexity of the reconstruction procedure to $\mathsf{FP}^{\mathsf{prMA}}$. Moreover, a careful examination of our proof reveals that, on all good input lengths, the reconstruction only makes smart queries to the prMA oracle. (We can use a bit of advice to signal which input lengths are good.)

We develop the necessary machinery for manipulating P^{NP} computations with a bounded number of adaptive rounds of NP queries, including a [Kre88]-style characterization, a PCP theorem, and the reconstruction argument. Note that we actually deal with the class $\mathsf{P}^{\mathsf{NP}}\left[\begin{smallmatrix}\#\text{rounds} = r \\ \text{length} = s\end{smallmatrix}\right]$, where each NP query has witness length at most s. The main reason to deal with the number of rounds and query lengths separately is that, using the standard search-to-decision reduction for SAT, we need $s \cdot r$ adaptive rounds of NP queries to compute the "computational history" of $\mathsf{P}^{\mathsf{NP}}\left[\begin{smallmatrix}\#\text{rounds} = r \\ \text{length} = s\end{smallmatrix}\right]$ (see **??**); fortunately this does not affect our overall proofs.

Finally, our techniques fall short of proving a near-maximum lower bound for smaller classes such as (the promise version of) $\mathsf{MA}_{\mathsf{EXP}}$. In **??**, we discuss the possibility of extending our results to these smaller classes.

2 Preliminaries

We use $x \circ y$ to denote the concatenation of two binary strings x and y.

A function $f : \mathbb{N} \to \mathbb{N}$ is *nice* if there is a deterministic algorithm that given n as input, outputs the value $f(n)$ in time $\mathrm{polylog}(n, f(n))$. This is just a somewhat arbitrary definition of functions that are

[3] In fact, this *is* the proof of $\mathsf{S}_2\mathsf{E}/_1 \not\subseteq \mathsf{SIZE}[2^n/n]$ presented in [CHR24]. The only difference is that instead of a general PCP, [CHR24] applied a specific Reed–Muller encoding tailored to the Jeřábek–Korten procedure. This allowed [CHR24] to prove their lower bounds in a relativizing way.

"not too pathological"; most functions that appear as complexity bounds are nice, such as $f(n) = n^c$, $f(n) = 2^{cn}$, or $f(n) = 2^{n^c}$ for constants $c > 0$.

2.1 Smart Reductions to Promise Problems

A *promise problem* [ESY84] is a pair of languages $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ with $\Pi_{\text{YES}} \cap \Pi_{\text{NO}} = \varnothing$. An algorithm $\mathcal{A}$ *solves* the promise problem if $\mathcal{A}$ accepts every input in Π_{YES} and rejects every input in Π_{NO}. The algorithm is allowed to behave arbitrarily on inputs not in the promise ($x \notin \Pi_{\text{YES}} \cup \Pi_{\text{NO}}$).

Let $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$ be a promise problem and L be a language. We say that there is a *smart* reduction from L to Π [GS88], denoted as $L \in \mathsf{smart\text{-}P}^{\Pi}$, if there is a deterministic polynomial-time oracle algorithm $\mathcal{A}$ with oracle access to Π such that, for every input x, $\mathcal{A}^{\Pi}(x)$ outputs the value of $L(x)$ and never makes any query outside the promise $\Pi_{\text{YES}} \cup \Pi_{\text{NO}}$.

We say that $L \in \mathsf{smart\text{-}P}^{\Pi}/_{a(n)}$ if the oracle algorithm $\mathcal{A}$ can access an advice string $\alpha_n \in \{0,1\}^{a(n)}$ that only depends on the input length n. The algorithm $\mathcal{A}$ is required to be correct and only make smart queries when given the correct advice string; however, when given an incorrect advice string, $\mathcal{A}$ is allowed to behave arbitrarily.

2.2 Merlin-Arthur Classes

We recall the definition of prMA:

Definition 2.1. A promise problem $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$ is in prMA if there is a deterministic polynomial-time algorithm $\mathcal{A}(x, w, r)$ such that $|w|, |r| \leq \text{poly}(|x|)$ and the following holds:

- **Completeness:** For every input $x \in \Pi_{\text{YES}}$, there exists some w such that
$$\Pr_r[\mathcal{A}(x, w, r) \text{ accepts}] = 1.$$
- **Soundness:** For every input $x \in \Pi_{\text{NO}}$ and every w,
$$\Pr_r[\mathcal{A}(x, w, r) \text{ accepts}] \leq 1/2.$$

Similarly, we say Π is in $\mathsf{prMA_E}$ (the *exponential-time* analogue of prMA) if $\mathcal{A}$ is allowed to run in $2^{O(|x|)}$ time and $|w|, |r| \leq 2^{O(|x|)}$.

One can replace the soundness parameter $1/2$ with any small constant by repetition. One can also replace the completeness parameter with any constant larger than the soundness parameter (in our case $1/2$) without altering the definition of prMA [ZF87; GZ11].

When we consider oracle access to the class prMA, it is without loss of generality to have the following specific prMA-complete problem in mind: We are given a circuit $A(w, r)$ as input, and
$$\Pi_{\text{YES}} := \{A : \exists w, \Pr_r[A(w, r) = 1] = 1\},$$
$$\Pi_{\text{NO}} := \{A : \forall w, \Pr_r[A(w, r) = 1] \leq 1/2\}.$$

In **??**, we also need the definition of $(\mathsf{MA} \cap \mathsf{coMA})/_{a(n)}$. Note that this is different from $\mathsf{MA}/_{a(n)} \cap \mathsf{coMA}/_{a(n)}$ in that there needs to be a *single* machine with the same sequence of advice strings that decides the problem in MA and coMA simultaneously; see also Section 3.5 of the journal version of [Che25].

Definition 2.2. Let $a(n)$ be a nice function. A language L is in $(\mathsf{MA} \cap \mathsf{coMA})/_{a(n)}$ if there exists a deterministic polynomial-time algorithm $\mathcal{A}(x, w, r, \alpha)$ and a sequence of advice strings $\{\alpha_n\}_{n \in \mathbb{N}}$ with each $|\alpha_n| = a(n)$, such that the following hold:

- $|w|, |r| \leq \text{poly}(|x|)$ and $\mathcal{A}(x, w, r, \alpha)$ outputs a value in $\{0, 1, \perp\}$. Think of w as Merlin's proof of the statement "$L(x) = b$" for some bit b, r as Arthur's randomness, and α as the advice string; if $\mathcal{A}$ accepts Merlin's proof then it outputs b, otherwise it outputs $\perp$.
- **Completeness:** For every $x \in \{0,1\}^\star$, there exists a string w such that

$$\Pr_r[\mathcal{A}(x, w, r, \alpha_{|x|}) = L(x)] = 1.$$

- **Soundness:** For every $x \in \{0,1\}^\star$ and w,

$$\Pr_r[\mathcal{A}(x, w, r, \alpha_{|x|}) = 1 - L(x)] \leq 1/2.$$

The exponential-time analogue of the above class, namely $(\mathsf{MA_E} \cap \mathsf{coMA_E})/_{a(n)}$, is defined similarly.

2.3 Self-Corrector for Low-Degree Polynomials

Let $\mathcal{D}$, $\mathcal{A}$ be finite sets, we say that two functions $f, g : \mathcal{D} \to \mathcal{A}$ are *δ-close* if

$$\Pr_{x \leftarrow \mathcal{D}}[f(x) \neq g(x)] \leq \delta.$$

We need a random self-correction procedure for low-degree polynomials:

Theorem 2.3 (Low-Degree Self-Corrector [GS92; Sud95])**.** *There is a probabilistic oracle algorithm* Corr *such that the following holds. Let $\mathbb{F}$ be a finite field, $m, \Delta \in \mathbb{N}$ such that $|\mathbb{F}| > \omega(\Delta^2)$. Let $P : \mathbb{F}^m \to \mathbb{F}$ be a polynomial of total degree at most Δ, and $g : \mathbb{F}^m \to \mathbb{F}$ be any function that is $1/4$-close to P, then for all $\vec{x} \in \mathbb{F}^m$, $\mathsf{Corr}^g(\mathbb{F}, m, \Delta, \vec{x})$ runs in time $\text{poly}(\Delta, \log|\mathbb{F}|, m)$ and outputs $P(\vec{x})$ with probability at least $2/3$. Moreover, if $g = P$, then for all $\vec{x} \in \mathbb{F}^m$, $\mathsf{Corr}^g(\mathbb{F}, m, \Delta, \vec{x})$ outputs $P(\vec{x})$ with probability 1.*

3 Bounded-Adaptive Queries to an NP Oracle

Crucial to our near-maximum circuit lower bounds is the investigation of a particular subclass of $\mathsf{P^{NP}}$, where we make *multiple* rounds of NP queries with *short* witnesses. Fix parameters $t(n), r(n), s(n)$, we say an algorithm runs in

$$\mathsf{TIME}[t(n)]^{\mathsf{NP}}\begin{bmatrix}\#\text{rounds} = r(n)\\ \text{length} = s(n)\end{bmatrix},$$

if it is a deterministic algorithm running in $t(n)$ time with access to an NP oracle, such that:

- **Round complexity:** The algorithm makes at most $r(n)$ rounds of NP queries. Each round can issue an unbounded number of queries in parallel, and the queries can depend on the answers of NP queries in previous rounds.
- **Witness length:** Every NP query can be solved by a nondeterministic Turing machine guessing at most $s(n)$ nondeterministic bits. That is, they can be reduced to SAT instances with at most $s(n)$ many inputs (in deterministic $t(n)$ time).

Since $t(n)$ is an upper bound on the time complexity of a $\mathsf{TIME}[t(n)]^{\mathsf{NP}}\begin{bmatrix}\#\text{rounds} = r(n)\\ \text{length} = s(n)\end{bmatrix}$ machine, throughout this paper we will assume that $r(n), s(n) \leq t(n)$. We also assume that the number of NP queries made in each round, as well as the sizes of the NP queries, are at most $t(n)$.

This class is important for two reasons:

1. The Jeřábek–Korten reduction [Jeř04; Kor21], an important ingredient for our circuit lower bounds (and also for [CHR24; Li24]), runs in $\mathsf{P}^{\mathsf{NP}}\begin{bmatrix}\mathsf{\#rounds} = r(n)\\ \mathsf{length} = s(n)\end{bmatrix}$ for some modest parameters $r(n)$ and $s(n)$. It turns out that our main proof in **??** needs to keep track of $r(n)$ and $s(n)$ carefully. This complexity upper bound of the Jeřábek–Korten reduction is proven in **??**.

2. Our main proof also needs an "encoded computational history" of this class, which is a slight adaptation of the PCP theorem [AS98; ALM+98]. This class is capable of computing the encoded computational history of itself with a moderate overhead; in fact, the encoded computational history of $\mathsf{P}^{\mathsf{NP}}\begin{bmatrix}\mathsf{\#rounds} = r(n)\\ \mathsf{length} = s(n)\end{bmatrix}$ can be computed in $\mathsf{P}^{\mathsf{NP}}\begin{bmatrix}\mathsf{\#rounds} = r(n)\cdot s(n)\\ \mathsf{length} = s(n)\end{bmatrix}$. This is established in **??**.

3.1 The Jeřábek–Korten Reduction

A core ingredient in the previous near-maximum circuit lower bounds [CHR24; Li24] and our new lower bounds is the Jeřábek–Korten reduction from AVOID to constructing a hard truth table. This reduction first appeared in [Jeř04] in the language of bounded arithmetic, and was rephrased in the language of total search problems and computational complexity in [Kor21].

Given an AVOID instance $G : \{0,1\}^n \to \{0,1\}^{2n}$ and the truth table of a function f with sufficiently high circuit complexity, the reduction $\mathsf{Jeřábek–Korten}(G, f)$ runs in deterministic $\mathrm{poly}(|G|, |f|)$ time with access to an NP oracle and outputs a string $y \in \{0,1\}^{2n} \setminus \mathrm{Range}(G)$. In the following theorem, we show that the number of *rounds of parallel* NP oracle queries in this reduction can be made $O(n + \log|f|)$, and the witness length of each NP query is at most n.

Theorem 3.1. *There is an algorithm* Jeřábek–Korten *that takes an* AVOID *instance* $G : \{0,1\}^n \to \{0,1\}^{2n}$ *(with* $|G| \geq n$*) and a truth table* $f \in \{0,1\}^T$ *as inputs, and outputs either*

- *the message* "`easy`" *and a circuit* C *of size* $O(|G| \log T)$ *that computes the truth table* f*, or*
- *the message* "`hard`" *and a string* $y \in \{0,1\}^{2n} \setminus \mathrm{Range}(G)$.

The algorithm Jeřábek–Korten *runs in* $\mathsf{TIME}[\mathrm{poly}(T, |G|)]^{\mathsf{NP}}\begin{bmatrix}\mathsf{\#rounds} = O(n + \log(T/n))\\ \mathsf{length} = n\end{bmatrix}$.

Proof. We shall follow the Jeřábek–Korten procedure [Jeř04; Kor21] closely, and make some minor modifications to optimize the number of query rounds. First, if $T \leq n$, then the theorem is trivial and no oracle queries are needed: we can just output "`easy`" along with a circuit C of size $O(T) \leq O(n)$ and $\log_2 T$ inputs, where C has the truth table of f hard-coded in a standard way.

So assume $T > n$. Let $k = \lceil \log_2(T/n) \rceil$ and let $T' = n2^k$. Let $f' = f0^{T'-T}$; that is, f' is f padded with $T' - T$ zeroes. Note that $|f'| < 2|f|$.

Our algorithm proceeds in k stages, starting with stage k and decreasing down to stage 1. Initially we set $y_k = f'$, a string of $n \cdot 2^k$ bits.

At the beginning of stage i, the current string y_i under consideration has length $n \cdot 2^i$. We partition y_i into contiguous $2n$-bit blocks $z_1, \ldots, z_{2^{i-1}}$, and form SAT queries $\phi_1, \ldots, \phi_{2^{i-1}}$, where ϕ_j is satisfiable if and only if there is an x_j such that $G(x_j) = z_j$. We are encoding a nondeterministic computation that takes at most $|G| \cdot \mathrm{polylog}|G|$ time, guesses x_j and evaluates the circuit G. Using the standard translation of nondeterministic time $t(n)$ into SAT queries of length $t(n) \cdot \mathrm{polylog}(t(n))$ (which holds for all reasonable models of computation [Sch78; Coo88; GS89]) we can construct each ϕ_j so that $|\phi_j| \leq |G| \cdot \mathrm{polylog}(|G|)$ for all j.

If SAT answers NO on some ϕ_j, then we immediately return "`hard`" and the corresponding string z_j as a solution to AVOID. Otherwise, for every Φ_j, let $x_j \in \{0,1\}^n$ denote its lexicographically smallest satisfying assignment. The simplest way to compute x_j would cost n parallel rounds per stage: one

round for each variable value. Letting $b \geq 1$ be a parameter, we observe that one can make only $m := \lceil n/b \rceil$ extra rounds per stage in a way that multiplies the overall running time by an $O(2^b)$ factor. For simplicity, let us round up the number of variables in each ϕ_j to be exactly $bm \leq 2n$ by adding some dummy variables (later, we will just remove them so that the total number is n again). Suppose for some $\ell \in \{0, \ldots, m-1\}$ and for every $j = 1, \ldots, 2^{i-1}$, the $(\ell \cdot b)$-bit prefix p_j of x_j has already been determined. Then in the next round, we form a collection of $2^b \cdot 2^{i-1}$ parallel SAT queries $\psi_{q,\ell,j}$ over all $q \in \{0,1\}^b$, such that $\psi_{q,\ell,j}$ is satisfiable if and only if there is an $x' \in \{0,1\}^n$ such that $G(x') = z_j$, the first $\ell \cdot b$ bits of x' equals p_j, and the next b bits of the prefix of x' equals q. Similarly to the ϕ_j, we can encode each $\psi_{q,\ell,j}$ so that they each have size at most $|G| \cdot \mathrm{polylog}(|G|)$ as well. The yes/no answers from all $2^b \cdot 2^{i-1}$ queries $\psi_{q,\ell,j}$ directly determine the next b bits of each x_j, and we update each prefix p_j accordingly by appending the lexicographically first q such that $\psi_{q,\ell,j}$ is satisfiable. Observe that the running time in each stage is multiplied by a factor of 2^b.

After we have determined all n bits of all x_j, we form a new string $y_{i-1} = x_1 \cdots x_{2^{i-1}}$ which has length $n \cdot 2^{i-1}$, which we pass to stage $i-1$, repeating the above process.

When we reach stage 1, the resulting y_0 has length n. In this case, we return "easy", since we can (by a standard argument, see [Kor21; GGM86]) construct a circuit of size $O(|G| \cdot \log T)$ whose truth table is exactly f', and therefore we can construct a circuit for f as well.

As we take $t = \lceil n/b \rceil + 1$ rounds in each stage, the total number of parallel rounds is $O(t \cdot k)$. Letting $b = \lceil \log(T \cdot |G|) \rceil$, we multiply the running time by $T \cdot |G|$ (a polynomial factor) and the number of parallel rounds becomes $O((\lceil n/b \rceil + 1)k) \leq O((\lceil n/\lceil \log(T \cdot |G|) \rceil + 1 \rceil) \log(T/n)) \leq O(n + \log(T/n))$. It is easy to see that every SAT query has witness length at most n. □

3.2 Encoded Computational History

We consider the *encoded computational history* of $\mathsf{P}^{\mathsf{NP}}\left[\begin{smallmatrix}\#\mathsf{rounds} = r(n)\\ \mathsf{length} = s(n)\end{smallmatrix}\right]$ machines.

The computational history of an NP machine on an input x is straightforward to define: we can simply provide an accepting computation path of the machine (a list of transitions taken by the machine from the initial configuration to the accept state). Applying the PCP theorem, we obtain an *encoded* computational history of the NP machine that we can check efficiently with randomness.

For a P^{NP} machine, we can reasonably define the computation history on an input x to be the history of the P machine's transitions along with a record of the queries asked. For each query that answers YES, we also provide accepting computation paths from the NP machine in the history just after the query is made; for each NO query, we insert an all-zero string. Observe that from this definition, finding the lexicographically largest accepting computation history determines the correct query answers (because we will always prefer an accepting computation history over an all-zeros string). Similarly, we obtain the *encoded* computational history by applying a PCP theorem to the raw computational history, and this facilitates fast lexicographic comparisons [Hir15].

In what follows, we generalize the above observation to $\mathsf{P}^{\mathsf{NP}}\left[\begin{smallmatrix}\#\mathsf{rounds} = r(n)\\ \mathsf{length} = s(n)\end{smallmatrix}\right]$. We also define its *encoded* computational history by applying the PCP theorem.

Lemma 3.2. *Let M be a $\mathsf{TIME}[t(n)]^{\mathsf{NP}}\left[\begin{smallmatrix}\#\mathsf{rounds} = r(n)\\ \mathsf{length} = s(n)\end{smallmatrix}\right]$ machine that outputs $\ell(n)$ bits. For every input x, there is a circuit C_x computable in deterministic $\mathrm{poly}(t(n))$ time given x, such that the following holds.*

Let $\alpha \in \{0,1\}^{\mathrm{poly}(t(n))}$ denote the lexicographically largest satisfying assignment of C_x. Then α can be computed in $\mathsf{TIME}[\mathrm{poly}(t(n))]^{\mathsf{NP}}\left[\begin{smallmatrix}\#\mathsf{rounds} = s(n) \cdot r(n)\\ \mathsf{length} = s(n)\end{smallmatrix}\right]$. Moreover, letting $r'(n) := r(n)\lceil \log(t(n)+1) \rceil$, for every $\alpha' \in \{0,1\}^{\mathrm{poly}(t(n))}$ such that $C_x(\alpha') = 1$ and the first $r'(n)$ bits of α and α' are equal, the substring of α from the $(r'(n)+1)$-th bit to the $(r'(n)+\ell(n))$-th bit is equal to the output of $M(x)$.

Proof. Let C_x denote the following circuit. Its input consists of $\mathsf{hist} = (y_1, \ldots, y_{r(n)}, out, z_{1,1}, \ldots, z_{r(n),t(n)})$ (in this order), with the following intended meanings:

- for each $i \in [r(n)]$, $y_i \in [0, t(n)]$ is the number of NP queries made in the i-th round of $M(x)$ that are accepted;
- $out \in \{0,1\}^{\ell(n)}$ is the output of $M(x)$; and
- for each $i \in [r(n)]$ and $j \in [t(n)]$, $z_{i,j} \in \{0,1\}^s$ is an NP witness for the j-th query made in the i-th round of $M(x)$.

We call such a tuple a "computational history" of $M(x)$; as we will demonstrate later, the lexicographically largest hist accepted by C_x is the "correct" computational history of $M(x)$.

Given any string hist, we can simulate the computation of $M(x)$ according to hist in $t(n)$ time: Let $q_{i,j}(\mathsf{hist})$ denote the j-th NP query made in the i-th round, then we answer YES to this query if the entry $z_{i,j}$ (in hist) satisfies $q_{i,j}(\mathsf{hist})$, and we answer NO otherwise. We stress that $q_{i,j}$ depends on the witnesses in previous rounds (i.e., $\{z_{i',j'}\}_{i'<i}$). We define $C_x(\mathsf{hist})$ accepts if and only if in the above simulation:

- for each $i \in [r(n)]$, the number of accepted NP queries in the i-th round is exactly y_i; and
- the final output of $M(x)$ is equal to out.

Let $\mathsf{hist}^\star = (y_1^\star, \ldots, y_{r(n)}^\star, out^\star, z_{1,1}^\star, \ldots, z_{r(n),t(n)}^\star)$ denote the "correct" computational history of $M(x)$. More precisely:

- for each $i \in [r(n)]$, $y_i^\star$ is the number of NP queries in the i-th round of $M(x)$ that are accepted;
- $out^\star$ is the final output of $M(x)$; and
- for each $i \in [r(n)]$ and $j \in [t(n)]$, if the j-th NP query in the i-th round of $M(x)$ is accepted, then $z_{i,j}^\star$ is the lexicographically largest witness of this query; otherwise $z_{i,j}^\star = 1^s$.

It is not hard to see that:

Claim 3.3. $\mathsf{hist}^\star$ *is the lexicographically largest satisfying assignment of* C_x.

Proof. Clearly, $C_x(\mathsf{hist}^\star) = 1$. Assume that $\mathsf{hist}' = (y_1', \ldots, y_{r(n)}', out', z_{1,1}', \ldots, z_{r(n),t(n)}')$ is any string lexicographically larger than $\mathsf{hist}^\star$ such that $C_x(\mathsf{hist}') = 1$, we now derive a contradiction.

Let $i \in [r(n)]$ and $j \in [t(n)]$. It is easy to show using induction that if $y_{i'}' = y_{i'}^\star$ for every $i' < i$, then $q_{i,j}(\mathsf{hist}') = q_{i,j}(\mathsf{hist}^\star)$, i.e., the two simulations of $M(x)$ on $\mathsf{hist}^\star$ and hist' makes the same NP query in the i-th round. The base case (where $i = 1$) is trivial. When $i > 1$, since $y_i^\star$ is the number of satisfiable NP queries among $\{q_{i-1,j}(\mathsf{hist}^\star)\}_{j\in[t(n)]} = \{q_{i-1,j}(\mathsf{hist}')\}_{j\in[t(n)]}$, it must be the case that the answers to every $q_{i-1,j}(\mathsf{hist}')$ are the same as those in $q_{i-1,j}(\mathsf{hist}^\star)$. Therefore, the NP queries made by $M(x)$ in the i-th round are the same when simulated using $\mathsf{hist}^\star$ or using hist'.

Suppose that there is some $i \in [r(n)]$ such that $y_i' \neq y_i^\star$. Let i be the smallest such index, then for every $j < i$, we have $y_j' = y_j^\star$. By the induction argument above, we have $q_{i,j}(\mathsf{hist}') = q_{i,j}(\mathsf{hist}^\star)$ for every $j \in [t(n)]$. However, there are only $y_i^\star < y_i'$ many satisfiable queries in the i-th round, therefore the number of $j \in [t(n)]$ such that $z_{i,j}'$ satisfies $q_{i,j}(\mathsf{hist}')$ cannot be equal to y_i'. This contradicts our assumption that $C_x(\mathsf{hist}')$ accepts.

It follows that $y_i' = y_i^\star$ for every $i \in [r(n)]$. The above induction implies that the NP queries and answers of $M(x)$ are exactly the same, when simulated using hist' or using $\mathsf{hist}^\star$. It follows that the output of $M(x)$ on hist' is exactly out, hence if $out' \neq out$ then $C_x(\mathsf{hist}') = 0$.

Finally, recall that $z_{i,j}^\star$ is the lexicographically largest witness of $q_{i,j}(\mathsf{hist}^\star)$ (when satisfiable) or 1^s (when unsatisfiable). Therefore if some $z_{i,j}' > z_{i,j}^\star$, it must be the case that $q_{i,j}(\mathsf{hist}')$ is satisfiable but $z_{i,j}'$ does not satisfy it. Hence the number of satisfied queries in the i-th round cannot be equal to $y_i' = y_i^\star$, a contradiction to our assumption that $C_x(\mathsf{hist}')$ accepts. ◇

That is, $\alpha := \mathsf{hist}^\star$ is the lexicographically largest satisfying assignment of C_x. We can compute $\mathsf{hist}^\star$ by simulating $M(x)$ using the NP oracle. Note that we also need to find the lexicographically largest satisfying assignments of each NP query (i.e., $z^\star_{i,j}$). This can be done by a binary search procedure with a blow-up of the round complexity by a multiplicative factor of $s(n)$.

Algorithm 1 Computing $\mathsf{hist}^\star$ in $\mathsf{TIME}[\mathrm{poly}(t(n))]^{\mathsf{NP}}\left[\begin{smallmatrix}\#\text{rounds} = r(n)\cdot s(n)\\ \text{length} = s(n)\end{smallmatrix}\right]$

for $i \leftarrow 1$ to $r(n)$ **do**
  Let $q_{i,1}, \ldots, q_{i,t(n)}$ be the NP queries issued in the i-th round
  $z_{i,j} \leftarrow \varepsilon$ for every $j \in [t(n)]$ ▷ *ε is the empty string (of length 0)*
  for $k \leftarrow 1$ to $s(n)$ **do**
    for $j \leftarrow 1$ to $t(n)$ **do**
      ▷ *We can issue the following $t(n)$ NP queries in parallel; each query has witness length at most $s(n)$*
      if there is a satisfying assignment of $q_{i,j}$ with prefix $(z_{i,j} \circ 1)$ **then**
        $z_{i,j} \leftarrow z_{i,j} \circ 1$
      else
        $z_{i,j} \leftarrow z_{i,j} \circ 0$
  for $j \leftarrow 1$ to $t(n)$ **do**
    Each $q_{i,j}$ is answered according to whether $z_{i,j}$ satisfies it
    if $q_{i,j}$ is not satisfied **then**
      $z_{i,j} \leftarrow 1^s$

Finally, the length of $(y_1, \ldots, y_{r(n)})$ is $r'(n) \le r(n)\lceil \log(t(n)+1)\rceil$. In the proof of **??** we have already seen that for any hist' that agrees with $\mathsf{hist}^\star$ on the first $r'(n)$ bits (i.e., agrees on $y_1, \ldots, y_{r(n)}$), if $C_x(\mathsf{hist}') = 1$, then the output part of hist' should be equal to $out^\star$. This establishes the "Moreover" part of the lemma. □

A PCP theorem. We need a PCP theorem where the PCP *contains an encoded version of the raw witness as a prefix*; this PCP theorem will be applied to the circuit C_x as defined in **??**. Fix an error-correcting code (looking ahead, we will use the Reed–Muller code). In our discussion, every PCP proof $\Pi = (\tilde{w}, \Pi')$ consists of two parts $\tilde{w}$ and Π', where $\tilde{w}$ is an encoding of a witness w and Π' denotes auxiliary proof oracles.

Roughly speaking, we need the following completeness and soundness guarantees:

- **Completeness:** Every valid witness w corresponds to a PCP proof $(\tilde{w}, \Pi')$ accepted with probability 1, where $\tilde{w}$ is the encoding of w.
- **Soundness:** If $\tilde{w}$ is far from encodings of valid witnesses, then for every Π', the PCP proof $(\tilde{w}, \Pi')$ is rejected with noticeable probability.

Such guarantees are satisfied by the classical Reed–Muller based PCP constructions [AS98; ALM+98]. A nice exposition of these PCPs can be found in Harsha's PhD thesis [Har04] and we will, in particular, verify that the robust PCP for SAT (ROBUST-PCP-CIRCUIT-SAT) in [Har04, Section 5.4.1] satisfies the above guarantees.

To be more precise, let us consider PCPs for the language CIRCUIT-SAT. Let C be a circuit of size n, we say that $w \in \{0,1\}^n$ is a *valid witness* for "$C \in$ CIRCUIT-SAT" if w consists of a satisfying assignment of C together with the values of intermediate gates of C on this assignment. Set $m := \left\lceil \frac{\log n}{\log\log n} \right\rceil$, $\mathbb{F}$ be a finite field of size at least $\alpha \cdot (mn^{1/m})^3 \le \mathrm{polylog}(n)$ where $\alpha \ge 1$ is a large enough universal constant, and $H \subseteq \mathbb{F}$ be a subset of size $n^{1/m}$ that contains $\{0,1\}$. (Here we assume, without loss of generality, that $n^{1/m}$ is an integer.) Fix a bijection between $[n]$ and H^m, then every string $w \in \{0,1\}^n$ can be seen as

a function $w : H^m \to \{0,1\}$, hence extends to a polynomial $\tilde{w} : \mathbb{F}^m \to \mathbb{F}$ with individual degree at most $n^{1/m}$. We call $\tilde{w}$ the *Reed–Muller encoding* of w.

Theorem 3.4. *For every small enough constant $\delta > 0$, there exists a PCP verifier V for* CIRCUIT-SAT *such that for every circuit C of size n:*

- **Completeness:** *For every valid witness w for "$C \in$ CIRCUIT-SAT", there exists a PCP proof $\Pi = (\tilde{w}, \Pi')$ such that $\tilde{w}$ is the Reed–Muller encoding of w and*

$$\Pr_z[V^{\Pi}(C, z) \text{ accepts}] = 1.$$

  Moreover, such a proof Π can be computed in deterministic $\mathrm{poly}(n)$ time given C and w as inputs.

- **Soundness:** *For every PCP proof $\Pi = (\tilde{w}, \Pi')$, if $\tilde{w}$ is δ-far from any polynomial $\tilde{w}' : \mathbb{F}^m \to \mathbb{F}$ of total degree at most $m \cdot |H|$ such that $\tilde{w}'|_{H^m}$ is a valid witness for "$C \in$ CIRCUIT-SAT", then*

$$\Pr_z[V^{\Pi}(C, z) \text{ accepts}] \leq 1/2.$$

- **Complexity:** *The PCP proof has length $|\Pi| \leq \mathrm{poly}(n)$. The verifier V takes a random string of length $|z| = O(\log n)$ and makes $\mathrm{polylog}(n)$ queries to Π.*

Proof Sketch. The PCP in [Har04, Section 5] consists of functions $\tilde{A} : \mathbb{F}^m \to \mathbb{F}$, $P : \mathbb{F}^{3m+3} \to \mathbb{F}$, and $\Pi_{\mathsf{ZoS}} : \mathbb{F}^{3m+3} \to \mathbb{F}^{6m+6}$, where $\tilde{A}$ is supposed to be the Reed–Muller encoding of a valid witness, P is an auxiliary proof oracle for ROBUST-PCP-CIRCUIT-SAT, and Π_{ZoS} is an auxiliary proof oracle for ZERO-ON-SUBCUBE. We denote $\tilde{w} = \tilde{A}$ and $\Pi' = (P, \Pi_{\mathsf{ZoS}})$.

- **Completeness:** The discussion before [Har04, Proposition 5.4.1] shows how to compute $\tilde{A}$ and P in polynomial time such that $\tilde{A}|_{H^m} = w$. Moreover, P is a Yes instance for ZERO-ON-SUBCUBE if and only if w is a valid witness for "$C \in$ CIRCUIT-SAT". Then, [Har04, Section 5.3.2] shows how to compute Π_{ZoS} in polynomial time given P.

- **Soundness:** This follows from [Har04, Lemma 5.4.4]. Note that $|\mathbb{F}| \geq \alpha \cdot (mn^{1/m})^3$ for some large enough constant α, and δ is a small enough constant, hence the hypotheses of [Har04, Lemma 5.4.4] are satisfied and this lemma is applicable. The soundness parameter can be amplified to $1/2$ by repeating the verification procedure $O(\delta^{-1})$ times.

- **Complexity:** As calculated after [Har04, Remark 5.4.3], the query complexity is $O(m|\mathbb{F}| \log |\mathbb{F}|) \leq \mathrm{polylog}(n)$ and the randomness complexity is $O(m \log |\mathbb{F}|) \leq O(m \log(mn^{1/m})) \leq O(\log n)$. The length of the PCP proof is $\mathrm{poly}(|\mathbb{F}|^m) \leq \mathrm{poly}(n)$. □

In what follows, we will deal with circuits taking $t'(n) \leq \mathrm{poly}(t(n))$ inputs. Therefore, we need to substitute n by $t'(n)$ in the instantiation of Reed–Muller codes. This in particular means that $m := \lceil \frac{\log t'(n)}{\log \log t'(n)} \rceil$, $\mathbb{F}$ is a finite field of size $\Theta(m \cdot t'(n)^{1/m})^3$, and $H \subseteq \mathbb{F}$ is a subset of size $t'(n)^{1/m}$.

The "encoded history". Let M be $\mathsf{TIME}[t(n)]^{\mathsf{NP}}\left[\begin{smallmatrix}\#\text{rounds} = r(n)\\ \text{length} = s(n)\end{smallmatrix}\right]$ machine, $x \in \{0,1\}^n$, and let C_x denote the circuit guaranteed in **??**. Let V be a PCP verifier for CIRCUIT-SAT as described in **??**, then V accepts PCP proofs of the form $\Pi = (\tilde{w}, \Pi')$ where $\tilde{w}$ is the Reed–Muller encoding of a valid witness for C_x and Π' denotes auxiliary proof oracles. Note that the Reed–Muller code is systematic (i.e., the original data appears as a prefix of the encoding), hence $\tilde{w}$ contains a satisfying assignment of C_x as a prefix. This satisfying assignment, in turn (by **??**), contains (pfx, out) as a prefix, where $\mathsf{pfx} \in \{0,1\}^{r'(n)}$ and out is (purportedly) the output of $M(x)$.

Theorem 3.5. *Let M be a* $\mathsf{TIME}[t(n)]^{\mathsf{NP}}\begin{bmatrix}\#\mathsf{rounds} = r(n)\\ \mathsf{length} = s(n)\end{bmatrix}$ *machine that outputs $\ell(n)$ bits, and $r'(n) := r(n) \cdot \lceil \log(t(n)+1) \rceil$. For any small enough constant $\delta > 0$, there is a "PCP verifier" $V^{\Pi}(x,z)$ such that for every $x \in \{0,1\}^n$, there exists a string $\mathsf{pfx}_x \in \{0,1\}^{r'(n)}$ such that the following holds:*

- **Completeness:** *There exists a PCP proof $\Pi = (\tilde{w}, \Pi')$ such that $\Pr_z[V^{\Pi}(x,z)$ accepts$] = 1$, pfx_x is a prefix of Π, and $\tilde{w}$ is a polynomial of total degree $\leq m|H|$. Such a proof can be computed in*

$$\mathsf{TIME}[\mathrm{poly}(t(n))]^{\mathsf{NP}}\begin{bmatrix}\#\mathsf{rounds} = r(n)\cdot s(n)\\ \mathsf{length} = s(n)\end{bmatrix}.$$

- **Soundness:** *Let $\Pi = (\tilde{w}, \Pi')$ be any PCP proof such that $\Pr_z[V^{\Pi}(x,z)$ accepts$] > 1/2$. Then $\tilde{w}$ is δ-close to some polynomial $\tilde{w}' : \mathbb{F}^m \to \mathbb{F}$ of total degree $m|H|$ such that the first $r'(n)$ bits of $\tilde{w}'$ are $\leq \mathsf{pfx}_x$ (in lexicographic order). Moreover, if these first $r'(n)$ bits are exactly equal to pfx_x, then the $(r'(n)+1 \sim r'(n)+\ell(n))$-th bits of $\tilde{w}'$ are equal to the output of $M(x)$.*

- **Complexity:** *The PCP proof has length $|\Pi| \leq \mathrm{poly}(t(n))$. The verifier V takes a random string of length $|z| = O(\log t(n))$ and makes $\mathrm{polylog}(t(n))$ queries to Π.*

Proof. Let $V^{\Pi}(x,z)$ be the verifier that given an input x, computes the circuit C_x as in **??** and runs $V_1^{\Pi}(C_x, z)$, where V_1 is the PCP verifier in **??**. For each input x, let α_x denote the lexicographically largest satisfying assignment of C_x, and let pfx_x denote the length-$r'(n)$ prefix of α_x.

- **Completeness:** Let w be the witness for "$C_x \in$ CIRCUIT-SAT" induced by the assignment α_x. Then, by the **Completeness** bullet in **??**, there exists a PCP proof $\Pi = (\tilde{w}, \Pi')$ such that $\tilde{w}$ is the Reed–Muller encoding of w and

$$\Pr_z[V^{\Pi}(x,z) \text{ accepts}] = \Pr_z[V_1^{\Pi}(C_x,z) \text{ accepts}] = 1.$$

  Moreover, Π can be computed in deterministic $\mathrm{poly}(t(n))$ time given w (and C_x). Since w can be computed in

$$\mathsf{TIME}[\mathrm{poly}(t(n))]^{\mathsf{NP}}\begin{bmatrix}\#\mathsf{rounds} = s(n)\cdot r(n)\\ \mathsf{length} = s(n)\end{bmatrix},$$

  given x, Π can be computed within asymptotically the same resource bound.

- **Soundness:** Since $\Pr_z[V^{\Pi}(x,z)$ accepts$] > 1/2$, by the **Soundness** bullet in **??**, $\tilde{w}$ is δ-close to some polynomial $\tilde{w}' : \mathbb{F}^m \to \mathbb{F}$ of total degree at most $m \cdot |H|$ such that $\tilde{w}'|_{H^m}$ is a valid witness for "$C_x \in$ CIRCUIT-SAT". Since α_x is the lexicographically largest such witness, $\tilde{w}'|_{H^m}$ is lexicographically $\leq \alpha_x$. Moreover, by **??**, if the length-$r'(n)$ prefix of $\tilde{w}'$ coincides with pfx_x, then the $(r'(n)+1 \sim r'(n)+\ell(n))$-th bits of $\tilde{w}'$ is equal to the output of $M(x)$.

- **Complexity:** Let $t'(n) \leq \mathrm{poly}(t(n))$ denote the size of C_x. The **Complexity** bullet in **??** implies that V accepts a proof of length $\mathrm{poly}(t'(n)) \leq \mathrm{poly}(t(n))$, takes a random string of length $O(\log t'(n)) \leq O(\log t(n))$, and makes $\mathrm{polylog}(t'(n)) \leq \mathrm{polylog}(t(n))$ queries to Π. □

4 A Near-Maximum Circuit Lower Bound

In this section, we prove **??**.

4.1 Instance-Wise Hardness-Randomness Tradeoff for AVOID

We start with the following theorem, which is essentially an *instance-wise hardness-randomness tradeoff* for solving AVOID in $\mathsf{P}^{\mathsf{NP}}\left[\begin{smallmatrix}\#\mathsf{rounds}=r(n)\\ \mathsf{length}=s(n)\end{smallmatrix}\right]$ for some modest parameters $r(n)$ and $s(n)$. (See also **??**). Crucially, this hardness-randomness tradeoff has a *uniform reconstruction procedure* computable in $\mathsf{smart\text{-}P}^{\mathsf{prMA}}$.

Theorem 4.1. *Let $r(n), s(n) \le t(n)$ be nice functions and let $f : \{0,1\}^n \to \{0,1\}^{t(n)}$ be a multi-output function computable in $\mathsf{TIME}[t(n)]^{\mathsf{NP}}\left[\begin{smallmatrix}\#\mathsf{rounds}=r(n)\\ \mathsf{length}=s(n)\end{smallmatrix}\right]$. For every nice function $k(n) \le t(n)$, there are algorithms Solve_f and Recon_f, which can be efficiently computed from the description of a* SAT*-oracle machine for f, such that the following holds:*

- $\mathsf{Solve}_f(x, G)$ *takes $x \in \{0,1\}^n$ and an* AVOID *instance $G : \{0,1\}^{k(n)} \to \{0,1\}^{2k(n)}$ of size at most $k(n)^4$ as inputs, outputs a string $y \in \{0,1\}^{2k(n)}$, and runs in*

$$\mathsf{TIME}[\mathrm{poly}(t(n))]^{\mathsf{NP}}\left[\begin{smallmatrix}\#\mathsf{rounds}=r^{\mathsf{Solve}}(n) := r(n)s(n) + O(k(n) + \log t(n))\\ \mathsf{length}=s^{\mathsf{Solve}}(n) := \max\{s(n), k(n)\}\end{smallmatrix}\right].$$

- $\mathsf{Recon}_f(x, i)$ *takes $x \in \{0,1\}^n$ and $i \in [t(n)]$ as input, runs in deterministic $\mathrm{poly}(k(n), r(n), \log t(n))$ time with access to a* prMA *oracle Π_f, and outputs a bit.*

  (As a minor remark, $\mathsf{Recon}_f(x, i)$ does not need access to the bad AVOID *instance G.)*

- *Let $x \in \{0,1\}^n$ and G be an* AVOID *instance such that $\mathsf{Solve}_f(x, G)$ fails to output a string outside $\mathrm{Range}(G)$. Then for every $i \in [t(n)]$, the output of $\mathsf{Recon}_f(x, i)$ is the i-th bit of $f(x)$, and $\mathsf{Recon}_f(x, i)$ only makes smart* prMA *oracle queries.*

Proof. We first describe the procedure $\mathsf{Solve}_f(x, G)$. Let $r'(n) := O(r(n) \log t(n))$, we first apply the PCP theorem (**??**) on the function f and obtain a "PCP verifier" $V^{\Pi}(x, z)$. For every input $x \in \{0,1\}^n$, we also obtain a PCP proof $\Pi_x \in \{0,1\}^{\mathrm{poly}(t(n))}$ as in the **Completeness** item of **??**. Then, $\mathsf{Solve}_f(x, G)$ invokes $\mathsf{Jeřábek–Korten}(G, \Pi_x)$ (**??**), using Π_x as the "hard truth table" to solve the AVOID instance G. We say that $\mathsf{Solve}_f(x, G)$ *succeeds* if $\mathsf{Jeřábek–Korten}(G, \Pi_x)$ outputs the message "hard" and a non-output of G.

We can compute Π_x in $\mathsf{TIME}[\mathrm{poly}(t(n))]^{\mathsf{NP}}\left[\begin{smallmatrix}\#\mathsf{rounds}=r(n)\cdot s(n)\\ \mathsf{length}=s(n)\end{smallmatrix}\right]$. Then, $\mathsf{Jeřábek–Korten}(G, \Pi_x)$ runs in $\mathsf{TIME}[\mathrm{poly}(t(n), |G|)]^{\mathsf{NP}}\left[\begin{smallmatrix}\#\mathsf{rounds}=O(k(n)+\log t(n))\\ \mathsf{length}=k(n)\end{smallmatrix}\right]$. Since $|G| \le k(n)^4 \le \mathrm{poly}(t(n))$, $\mathsf{Solve}_f(x, G)$ can be computed in

$$\mathsf{TIME}[\mathrm{poly}(t(n))]^{\mathsf{NP}}\left[\begin{smallmatrix}\#\mathsf{rounds}=r(n)\cdot s(n)+O(k(n)+\log t(n))\\ \mathsf{length}=\max\{s(n),k(n)\}\end{smallmatrix}\right].$$

Next we describe the procedure $\mathsf{Recon}_f(x, i)$; assuming $\mathsf{Solve}_f(x, G)$ does not succeed, the goal of this procedure is to output the i-th bit of $f(x)$ in $\mathrm{poly}(k(n), r(n), \log t(n))$ time with smart access to a prMA oracle. For a PCP proof $\Pi = (\tilde{w}, \Pi')$, let $\tilde{w}' : \mathbb{F}^m \to \mathbb{F}$ denote the polynomial of total degree at most $m|H|$ that is closest to $\tilde{w}$, and define $\widetilde{\mathsf{pfx}}(\Pi)$ and $\widetilde{out}(\Pi)$ to be the first $r'(n)$ bits and the $(r'(n)+1 \sim r'(n)+t(n))$-th bits of $\tilde{w}'$, respectively. (Recall the parameters here: $m = O\left(\frac{\log t(n)}{\log\log t(n)}\right)$, $|H| := \mathrm{poly}(t(n)^{1/m}) \le \mathrm{polylog}(t(n))$ and $|\mathbb{F}| = \Theta((m|H|)^3)$.) We also say that Π is *sound* if $\Pr_z[V^{\Pi}(x, z) \text{ accepts}] > 1/2$. By the **Soundness** item of **??**, if Π is any sound PCP proof that maximizes $\widetilde{\mathsf{pfx}}(\Pi)$ (over all sound PCP proofs), then $\widetilde{out}(\Pi)$ is equal to $f(x)$. Hence, the task of $\mathsf{Recon}_f(x, i)$ is to find (a succinct description of) such a PCP proof Π and output the i-th bit in $\widetilde{out}(\Pi)$.

Note that whenever $\mathsf{Solve}_f(x, G)$ does not succeed, by **??**, the circuit complexity of Π_x is at most $k'(n) := O(k(n)^4 \log t(n))$. The algorithm $\mathsf{Recon}_f(x, i)$ proceeds in $r'(n)$ rounds. It maintains a string pfx (initially the empty string). In each round, it asks the following prMA query q_{pfx}:

- Merlin provides a circuit $C : \{0,1\}^{\log|\Pi_x|} \to \{0,1\}$ of size at most $k'(n)$.
- Arthur samples randomness $z \leftarrow O(\log t(n))$ (for the PCP verifier) and $z_{\mathsf{Corr}} \leftarrow \{0,1\}^{\mathrm{polylog}(t(n))}$ (for the Low-Degree Self-Corrector).
- If $V^C(x,z)$ rejects then Arthur rejects. Otherwise, let Π_C be the truth table of C (which is a PCP proof), Arthur accepts if and only if $\mathsf{pfx} \circ 1$ is a prefix of $\widetilde{\mathsf{pfx}}(\Pi_C)$ (where each bit of $\widetilde{\mathsf{pfx}}(\Pi_C)$ is evaluated by the Low-Degree Self-Corrector).

If q_{pfx} accepts, then we let $\mathsf{pfx} \leftarrow \mathsf{pfx} \circ 1$; otherwise we let $\mathsf{pfx} \leftarrow \mathsf{pfx} \circ 0$. Then we proceed to the next round. At the end of $r'(n)$ rounds, we obtain a string $\mathsf{pfx} \in \{0,1\}^{r'(n)}$. We make the final query $q_{\mathsf{pfx},i}$, which is the same query as q_{pfx} except that Arthur additionally rejects if the i-th bit $\widetilde{out}(\Pi_C)$ is 0 (again, we access $\widetilde{out}(\Pi_C)$ using the Low-Degree Self-Corrector). If $q_{\mathsf{pfx},i}$ accepts, then $\mathsf{Recon}_f(x,i)$ outputs 1; otherwise $\mathsf{Recon}_f(x,i)$ outputs 0.

Clearly, $\mathsf{Recon}_f(x,i)$ runs in $\mathrm{poly}(k(n), r(n), \log t(n))$ time and the size of each query q_{pfx} and $q_{\mathsf{pfx},i}$ is also upper bounded by $\mathrm{poly}(k(n), r(n), \log t(n))$.

Assuming $\mathsf{Solve}_f(x,G)$ does not succeed, we show that every query made by $\mathsf{Recon}_f(x,i)$ is smart (i.e., in the promise of prMA). Recall that Π_x is the PCP proof for x guaranteed in the **Completeness** item of **??**. We first use an induction to show that pfx (i.e., the string finally obtained in $\mathsf{Solve}_f(x,G)$) is equal to the length-$r'(n)$ prefix of Π_x. For notation convenience, let $\mathsf{pfx}_{\le j}$ denote the length-j prefix of pfx, hence the prMA query made in the j-th round is $q_{\mathsf{pfx}_{\le j-1}}$. Let $j < r'(n)$, suppose that after j rounds, pfx is equal to the length-j prefix of Π_x (this is clearly true for $j = 0$), we now consider the $(j+1)$-th round:

- If the $(j+1)$-th bit of Π_x is 1, then in $q_{\mathsf{pfx}_{\le j}}$, there exists a response of Merlin that makes Arthur accept with probability 1. Hence, $q_{\mathsf{pfx}_{\le j}}$ satisfies the promise of YES instances of prMA.

  To see this, let C be the size-$k'(n)$ circuit whose truth table is Π_x. Then $\Pr_z[V^C(x,z) \text{ accepts}] = 1$. Since $\Pi_x = (\tilde{w}_x, \Pi'_x)$ for some polynomial $\tilde{w}_x$ of total degree at most $m|H|$, the Low-Degree Self-Corrector always returns the correct value in $\widetilde{\mathsf{pfx}}(\Pi_x)$ and $\widetilde{out}(\Pi_x)$ with probability 1. Finally, $\mathsf{pfx}_{\le j} \circ 1$ is a prefix of $\widetilde{\mathsf{pfx}}(\Pi_x)$. It follows that Arthur accepts with probability 1 given Merlin's proof C.

- If the $(j+1)$-th bit of Π_x is 0, then in $q_{\mathsf{pfx}_{\le j}}$, no matter what circuit Merlin provides to Arthur, Arthur rejects w.p. $\ge 1/2$. Hence, $q_{\mathsf{pfx}_{\le j}}$ satisfies the promise of NO instances of prMA.

  To see this, let C be any circuit and $\Pi_C = (\tilde{w}_C, \Pi'_C)$ be its truth table (which corresponds to a PCP proof). If $\Pr_z[V^C(x,z) \text{ accepts}] \le 1/2$, then Arthur rejects with probability $\ge 1/2$. Otherwise, by the **Soundness** of **??**, $\tilde{w}_C$ is $1/4$-close to a polynomial of total degree $m|H|$, and $\widetilde{\mathsf{pfx}}(\Pi_C)$ is no larger than the length-$r'(n)$ prefix of Π_x, which is strictly smaller than $\mathsf{pfx}_{\le j} \circ 1$. Since $|\mathbb{F}| > \omega(|H|^2)$, the Low-Degree Self-Corrector is always correct with high probability (the correct probability can be boosted to $1 - 1/\mathrm{poly}(t(n))$ by repetition), which means that Arthur will detect that $\mathsf{pfx}_{\le j} \circ 1$ is not a prefix of $\widetilde{\mathsf{pfx}}(\Pi_C)$ with high probability and hence reject.

It follows that the $(j+1)$-th prMA query satisfies the prMA promise, and it is a YES instance if and only if the $(j+1)$-th bit of Π_x is 1. Therefore, after $r'(n)$ rounds, pfx is equal to the length-$r'(n)$ prefix of Π_x.

Similarly, we can show that the final query $q_{\mathsf{pfx},i}$ is inside the prMA promise and is a YES instance if and only if the i-th bit of $f(x)$ is 1. The proof is very similar so we only provide a sketch:

- If the i-th bit of $f(x)$ is indeed 1, then Arthur accepts with probability 1 when Merlin sends a small circuit whose truth table is Π_x.

- If the i-th bit of $f(x)$ is 0, then for every PCP proof Π, either $\Pr_z[V^C(x,z) \text{ accepts}] \leq 1/2$, or $\widetilde{\mathsf{pfx}}(\Pi)$ is strictly smaller than the length-$r'(n)$ prefix of Π_x, or $\widetilde{\mathsf{pfx}}(\Pi)$ is equal to that prefix. In the first two cases, Arthur rejects w.p. $\geq 1/2$. In the last case, by the **Soundness** item of **??**, we have $\widetilde{out}(\Pi) = f(x)$, hence the i-th bit of $\widetilde{out}(\Pi)$ is 0 and Arthur rejects w.h.p. as well.

In conclusion, when $\mathsf{Solve}_f(x, G)$ does not succeed, $\mathsf{Recon}_f(x, i)$ outputs the i-th bit of $f(x)$ in deterministic $\mathrm{poly}(k(n), r(n), \log t(n))$ time with smart access to a prMA oracle. This finishes the proof. □

Remark 4.2. **??** is an *instance-wise hardness-randomness tradeoff* for solving AVOID in the following sense. Given any multi-output function f computable in a certain deterministic class $\mathcal{A}$ that is almost-all-input hard against a certain randomized class $\mathcal{B}$, we can solve the range avoidance problem in a related deterministic class $\mathcal{C}$. The statement of **??** implies such an instance-wise hardness-randomness tradeoff where

- $\mathcal{A} = \mathsf{TIME}[t(n)]^{\mathsf{NP}}\left[\begin{smallmatrix}\#\mathsf{rounds} = r(n)\\ \mathsf{length} = s(n)\end{smallmatrix}\right]$,
- $\mathcal{B} = \text{smart-}\mathsf{TIME}[\mathrm{poly}(k(n), r(n), \log t(n))]^{\mathsf{prMA}}$, and
- $\mathcal{C} = \mathsf{TIME}[\mathrm{poly}(t(n))]^{\mathsf{NP}}\left[\begin{smallmatrix}\#\mathsf{rounds} = r(n)s(n) + O(k(n) + \log t(n))\\ \mathsf{length} = \max\{s(n), k(n)\}\end{smallmatrix}\right]$.

It is instructive to compare this statement to that of [CT21]. In [CT21] it was proven that given any multi-output function computable in a certain deterministic class $\mathcal{A}$ that is almost-all-input hard against a certain randomized class $\mathcal{B}$, we can solve GapSAT (the canonical prRP-complete problem) in a related deterministic class $\mathcal{C}$, where:

- $\mathcal{A}$ is the class of logspace-uniform circuits of depth d and time T;
- $\mathcal{B}$ is the class of randomized algorithms running in time $\mathrm{poly}(d, \log T, n)$; and
- $\mathcal{C}$ is the class of deterministic algorithms running time $\mathrm{poly}(T, n)$.

4.2 The Lower Bound

Fix a P-uniform family of AVOID instances $\{G_n\}_{n=2^k, k\in\mathbb{N}}$, where for each $n = 2^k$ a power of 2, $G_n : \{0,1\}^n \to \{0,1\}^{2n}$ is a circuit of size at most n^4. We show that there exists a $\text{smart-}\mathsf{FP}^{\mathsf{MA}}/_1$ algorithm solving AVOID on this family infinitely often. Looking ahead, combined with the Jeřábek–Korten reduction from AVOID to finding hard truth tables, our algorithm will be able to solve general AVOID instances (without uniformity constraints) on infinitely many input lengths.

Theorem 4.3. *There is an infinitely-often* $\text{smart-}\mathsf{FP}^{\mathsf{prMA}}/_1$ *algorithm* $\mathcal{A}$ *solving* AVOID*. In particular, there are infinitely many input lengths* $\{n_i = 2^{k_i}\}$ *such that* $\mathcal{A}(1^{n_i})$ *outputs a string* $y \in \{0,1\}^{2n_i} \setminus \mathrm{Range}(G_{n_i})$*, and for every input length* $n = 2^k$*,* $\mathcal{A}(1^n)$ *never makes any query outside the* prMA *promise.*

Proof. Let $C \geq 3$ be a sufficiently large integer constant in the following. Let $\{n_i\}$, $\{T_i\}$, $\{r_i\}$, $\{s_i\}$ be parameters to be fixed (corresponding to input lengths, running times, number of parallel rounds, and query lengths, respectively). Let $\{G_n\}_{n=2^k,k\in\mathbb{N}}$ denote a P-uniform family of AVOID instances where for each $n = 2^k$ a power of 2, $G_n : \{0,1\}^n \to \{0,1\}^{2n}$ is a circuit of size at most n^4. Let $n_0 = 2^q$ be some fixed input length, let $T_0 = 2^{cn_0}$ for a universal constant $c \geq 1$, and let $r_0 = 0$, $s_0 = 0$. Let BF_0 denote the **Brute Force** algorithm for solving AVOID on G_{n_0}, which simply enumerates every string in $\{0,1\}^{n_0}$, computes the whole range of G_{n_0}, and outputs the lexicographically first string not in $\mathrm{Range}(G_{n_0})$. Then, BF_0 runs in deterministic T_0 time and makes no queries to the SAT oracle, so $r_0 = 0$ and $s_0 = 0$ suffice.

Fix a universal constant $D \geq 1$ large enough so that the running time of Solve (from **??**) is asymptotically bounded by $t(n)^D$, and set C so that $C > D + 1$. Let $n_0 = 2^q$ be large enough that $2^{cn_0} \geq n_0^C$. For each $i \geq 0$, set

$$n_{i+1} := n_i^C \qquad \text{and} \qquad T_i := \max\{2^{cD^i n_0}, n_{i+1}\},$$

and note that by our choice of n_0, our definition of T_i is consistent with our earlier definition of $T_0 = 2^{cn_0}$.

We inductively define an algorithm BF_i for the instance G_{n_i}. At stage i, fix $x_0 := 0^{n_0}$ and let $f_i : \{0,1\}^{n_0} \to \{0,1\}^{T_i}$ be the function that ignores its input and outputs the $2n_i$-bit string produced by BF_i on G_{n_i}, padded with extra zeros so that the total output has length T_i. Since $T_i \geq 2n_i$ and BF_i runs in $\mathsf{TIME}[T_i]^{\mathsf{NP}}\begin{bmatrix}\mathsf{\#rounds} = r_i \\ \mathsf{length} = s_i\end{bmatrix}$, the function f_i can also be implemented in time $O(T_i)$ with r_i rounds of NP queries with witness length s_i. Also, since $T_i \geq n_{i+1}$ and $|G_{n_{i+1}}| \leq n_{i+1}^4$, **??** applies to f_i with $k(n_0) = n_{i+1}$. (Technically, we need f_i to be defined on all inputs; we can simply have f_i output all-zeroes on inputs that are not of length n_0.) From the theorem, we obtain:

- an algorithm $\mathsf{BF}_{i+1} := \mathsf{Solve}_{f_i}(x_0, G_{n_{i+1}})$ that attempts to solve AVOID on $G_{n_{i+1}}$ and runs in

$$\mathsf{TIME}[T_i^D]^{\mathsf{NP}}\begin{bmatrix}\mathsf{\#rounds} = r_{i+1} := r_i s_i + O(n_{i+1} + \log T_i) \\ \mathsf{length} = s_{i+1} := \max\{s_i, n_{i+1}\}\end{bmatrix}, \text{as well as}$$

- an algorithm $\mathsf{Recon}_i(j) := \mathsf{Recon}_{f_i}(x_0, j)$ that runs in deterministic $\mathrm{poly}(n_{i+1}, r_i, \log T_i)$ time with access to a prMA oracle. If BF_{i+1} fails to solve AVOID on $G_{n_{i+1}}$, then for every $j \in [T_i]$, $\mathsf{Recon}_i(j)$ outputs the j-th bit of $f_i(x_0)$ while making only smart prMA queries.

By our choice of C and D, the running time of $\mathsf{BF}_{i+1} = \mathsf{Solve}_{f_i}(x_0, G_{n_{i+1}})$ is bounded from above by

$$T_i^D = \max\{2^{cD^{i+1}n_0}, n_{i+1}^D\} \leq \max\{2^{cD^{i+1}n_0}, n_{i+1}^C\} = \max\{2^{cD^{i+1}n_0}, n_{i+2}\} = T_{i+1}.$$

Therefore T_{i+1} is a time upper bound for BF_{i+1}.

Let us bound the other parameters. First, we observe that $s_i = n_i$ for every $i \geq 1$, since $s_0 = 0$ and $s_{i+1} = \max\{s_i, n_{i+1}\}$. Second, we claim that $r_i \leq O(n_i)$ for every $i \geq 1$. Indeed, let $c' \geq 1$ be a constant such that $r_{i+1} = r_i s_i + c' \cdot (n_{i+1} + \log T_i)$, then we can inductively prove that $r_i \leq 3c' n_i$ for every $i \geq 0$. This is because $r_0 = 0$, and if $r_i \leq 3c' n_i$ then

$$r_{i+1} \leq r_i s_i + c' \cdot (n_{i+1} + \log T_i) \leq 3c' \cdot n_i^2 + c' \cdot (n_{i+1} + \log T_i) \leq 3c' \cdot n_{i+1},$$

because $s_i = n_i$, $C \geq 3$ (which means $n_i^2 \leq o(n_{i+1})$), and $\log T_i \leq O(D^i n_0 + \log n_{i+1}) \leq n_{i+1}$ for all sufficiently large n_0. Hence one call to $\mathsf{Recon}_i(j)$ takes $\mathrm{poly}(n_i)$ time with a prMA oracle, and reconstructing all first $2n_i$ bits of f_i takes $\mathrm{poly}(n_i)$ time with a prMA oracle.

Let $i_\star$ be the least index such that $2^{cD^{i_\star} n_0} \leq n_{i_\star + 1}$. Such an $i_\star$ exists because $C > D$ and $n_{i+1} = n_0^{C^{i+1}}$. By definition of $i_\star$, for every $i \geq i_\star$ we have $T_i = \max\{2^{cD^i n_0}, n_{i+1}\} = n_{i+1}$. To see this, note that once $2^{cD^i n_0} \leq n_{i+1}$ is true for some i, it is true for all larger i (because $D < C$, so $2^{cD^{i+1} n_0} \leq n_{i+1}^C = n_{i+2}$). Therefore $T_i = n_{i+1} = n_i^C$, so BF_i can be simulated in deterministic $\mathrm{poly}(n_i)$ time with oracle access to SAT.

We claim that the above sequence of input lengths $\{n_i\}$ contributes at least one "good" input length on which AVOID is correctly solved in polynomial time. If there is some $i < i_\star$ for which BF_{i+1} fails on $G_{n_{i+1}}$, let i be the least such index. Then BF_i succeeds on G_{n_i}, and since BF_{i+1} fails, the reconstruction guarantee above implies that for every $j \in [2n_i]$, $\mathsf{Recon}_i(j)$ outputs the j-th bit of $f_i(x_0)$, and these first $2n_i$ bits are the output of BF_i on G_{n_i}. Therefore, by running $\mathsf{Recon}_i(1), \ldots, \mathsf{Recon}_i(2n_i)$, we obtain a string in $\{0,1\}^{2n_i} \setminus \mathrm{Range}(G_{n_i})$ in deterministic $\mathrm{poly}(n_i)$ time with smart prMA queries. On the other hand, if no such $i < i_\star$ exists, then $\mathsf{BF}_{i_\star}$ succeeds on $G_{n_{i_\star}}$, and it runs in deterministic $\mathrm{poly}(n_{i_\star})$ time with oracle access to SAT by the previous paragraph.

We now run the above construction independently with different starting input lengths n_0: namely, we consider all $n_0 = 2^q$ such that q is not divisible by C. Observe that *every* sufficiently large $n = 2^k$ belongs to exactly one such sequence $\{n_i\}$, obtained by writing $k = qC^i$ for q not divisible by C. (For those finitely many $n_0 = 2^q$ that are not large enough in the analysis above, our final algorithm will simply output 0^{2n} and make no oracle queries; note this won't affect the infinitely-often conclusion of the theorem.)

Using the P-uniformity of $\{G_n\}$ and the uniformity of the transformations in **??**, we obtain two uniform polynomial-time oracle algorithms $\mathcal{A}_0$ and $\mathcal{A}_1$ with oracle access to prMA:

- On input 1^n, $\mathcal{A}_0$ locates the (unique) sequence of inputs $\{n_i\}$ containing n, the corresponding n_0, and the index i such that $n_i = n$ in this sequence. If $i \geq i_\star$ (recall $i_\star$ is defined as the least index such that $2^{cD^{i_\star} n_0} \leq n_{i_\star+1}$), then $\mathcal{A}_0$ simulates BF_i using queries to SAT. Otherwise, $\mathcal{A}_0$ outputs 0^{2n}.

- On input 1^n, $\mathcal{A}_1$ locates the sequence of inputs $\{n_i\}$ containing n, runs the corresponding reconstruction algorithm $\mathsf{Recon}_i(j)$ for each $j \in [2n]$, and outputs the resulting $2n$-bit string.

Both algorithms run in deterministic polynomial time. Moreover, every query made by $\mathcal{A}_0$ is automatically inside the promise of prMA, while $\mathcal{A}_1$ is smart exactly on those lengths for which the solver BF_{i+1} fails.

Finally, for each input length sequence $\{n_i\}$ defined by some $n_0 = 2^q$, let $n(q)$ be some good input length in that sequence. If $n(q)$ comes from the output of $\mathcal{A}_0$ then we set the advice bit $\alpha_{n(q)} := 0$, otherwise we set $\alpha_{n(q)} := 1$. On all other lengths n, we set $\alpha_n := 0$. We define $\mathcal{A}(1^n, \alpha_n) := \mathcal{A}_{\alpha_n}(1^n)$. Then $\mathcal{A}$ runs in polynomial time and is smart on every power-of-two input length. Moreover, for every sufficiently large q there is a distinct input length $n = n(q)$ such that $\mathcal{A}$ outputs a string in $\{0,1\}^{2n} \setminus \mathrm{Range}(G_n)$. Observe that for all distinct $q \neq q'$ that are large enough, the corresponding good lengths $n(q)$ and $n(q')$ must be distinct. Since there are infinitely many such q, there are infinitely many good input lengths. This concludes the proof. □

For each $n = 2^k$, define $G_n : \{0,1\}^n \to \{0,1\}^{2n}$ to be the truth table generator; that is, the input of G_n is a circuit $C : \{0,1\}^{k+1} \to \{0,1\}$ described in n bits, and the output is the truth table of C of length $2^{k+1} = 2n$. As shown in [Kor21], the AVOID problem for general circuits reduces to the AVOID problem on the family $\{G_n\}$ via an $\mathsf{FP}^{\mathsf{NP}}$ reduction.

Lemma 4.4 ([Kor21, Theorem 7], see also [CHR24, Lemma 5.14]). *There is an* $\mathsf{FP}^{\mathsf{NP}}$ *algorithm* $\mathcal{A}_{\mathrm{Korten}}$ *such that:*

1. $\mathcal{A}_{\mathrm{Korten}}$ *takes an* s*-size circuit* $C : \{0,1\}^n \to \{0,1\}^{n+1}$ *and a truth table* $f \in \{0,1\}^{2^k}$ *as inputs, where* $2^k \geq s^3$ *and* $n \leq s$.

2. *If the circuit complexity of* f *is at least* $c_1 \cdot k \cdot s$ *for a sufficiently large universal constant* c_1*, then* $\mathcal{A}_{\mathrm{Korten}}(C, f)$ *outputs a string* $y \in \{0,1\}^{n+1} \setminus \mathrm{Range}(C)$.

Therefore, we have:

Theorem 4.5. *There is an infinitely often* $\mathsf{smart\text{-}FP}^{\mathsf{prMA}}/_1$ *algorithm* $\mathcal{A}$ *such that for infinitely many* $s \in \mathbb{N}$*, for all* s*-size circuit* $C : \{0,1\}^n \to \{0,1\}^{n+1}$ *when* $n \leq s$*,* $\mathcal{A}(C)$ *outputs a string* $y_C \in \{0,1\}^{n+1} \setminus \mathrm{Range}(C)$.

Proof. Let $\mathcal{A}'$ denote the $\mathsf{smart\text{-}FP}^{\mathsf{prMA}}/_1$ algorithm in **??** for the family of truth table generators $\{G_n\}$. There are infinitely many integers k such that $\mathcal{A}'(1^{2^k})$ outputs a truth table of length 2^{k+1} that cannot be computed by circuits of size $0.1 \cdot 2^k/k$.

Given a circuit C of size s, let $k := \lceil 4 \log s \rceil$ and $f \in \{0,1\}^{2^k}$ be the output of $\mathcal{A}'(1^{2^{k-1}})$. Our algorithm $\mathcal{A}(C)$ outputs $\mathcal{A}_{\mathrm{Korten}}(C, f)$. By **??**, our algorithm is correct on infinitely many $s \in \mathbb{N}$. □

Just as in [CHLR26, Section 4.3], the ability to solve the Range Avoidance problem in $\text{smart-FP}^{\text{prMA}}/_1$ allows us to prove a number of "near-maximum" complexity lower bounds for $\text{smart-E}^{\text{prMA}}/_1$, via direct reductions to AVOID.

Corollary 4.6. *The following "near-maximum" complexity lower bounds hold for* $\text{smart-E}^{\text{prMA}}/_1$*:*

1. $\text{smart-E}^{\text{prMA}}/_1 \not\subseteq \mathsf{SIZE}[2^n/n]$.
2. *Let* $\delta_1, \delta_2 > 0$ *be constants such that* $\delta_1 + 2\delta_2 < 1$*. Then there is a language* $L \in \text{smart-E}^{\text{prMA}}/_1$ *that cannot be* $(1/2 + 2^{-\delta_2 n})$*-approximated by size-*$2^{\delta_1 n}$ *circuits.*
3. *For any constant* $k \geq 1$ *and nice function* $\alpha(n) \geq \omega(1)$, $\text{smart-E}^{\text{prMA}}/_1 \not\subseteq \mathsf{TIME}[2^{kn}]/_{2^n - \alpha(n)}$.

Finally, we note that our AVOID algorithm implies a $\text{smart-FP}^{\text{prMA}}/_1$ algorithm for explicit constructions of Ramsey graphs, two-source extractors, rigid matrices, optimal linear codes meeting the GV bound, etc. These follow directly from known uniform reductions to AVOID [Kor21; GLW22; RSW22].

5 Discussion

We believe the approach of this paper has the potential to obtain near-maximum circuit lower bounds for classes such as prMA_{E}. In this section, we discuss the immediate technical obstacles we are facing and provide some speculations on future directions. (We also note that our approach is radically different from the half-exponential lower bound in **??**. In particular, non-relativizing techniques such as the PCP theorem are crucial to our proofs.)

A significant bottleneck in our proof is the *highly adaptive* search-to-decision reduction for SAT: the Jeřábek–Korten reduction (**??**) essentially makes $O(\log T)$ rounds of non-adaptive queries to searchSAT, and the computational history of $\mathsf{P}^{\mathsf{NP}}\begin{bmatrix}\#\text{rounds} = r(n)\\ \text{length} = s(n)\end{bmatrix}$ (**??**) is also computable in $r(n)$ rounds of non-adaptive queries to searchSAT. Replacing the searchSAT oracle with a (decisional) SAT oracle incurs an $O(n/\log T)$ and $O(s(n))$ multiplicative overhead on the number of rounds, respectively. In fact, it is not hard to see from our proof that an $\mathsf{FP}^{\mathsf{NP}}_{tt}$ algorithm for searchSAT would imply near-maximum circuit lower bounds for smaller classes; here, $\mathsf{FP}^{\mathsf{NP}}_{tt}$ is the class of search problems computable by a deterministic polynomial-time algorithm with *non-adaptive* access to NP oracles.

Theorem 5.1. *Assuming there is an* $\mathsf{FP}^{\mathsf{NP}}_{tt}$ *algorithm for* searchSAT*. Recall* $\mathsf{E}^{\text{prMA}[2^{\varepsilon n}]}$ *is the class of languages computable in exponential time with* $2^{\varepsilon n}$ *many adaptive queries to a* prMA *oracle. Then, for every constant* $\varepsilon > 0$*,*

$$\text{smart-E}^{\text{prMA}[2^{\varepsilon n}]}/_1 \not\subseteq \mathsf{SIZE}[2^n/(2n)] \quad \textit{and} \quad (\mathsf{MA}_{\mathsf{E}} \cap \mathsf{coMA}_{\mathsf{E}})/_{2^{\varepsilon n}} \not\subseteq \mathsf{SIZE}[2^n/(2n)].$$

We treat **??** as an invitation to the open questions (**??** and **??**) that we will discuss later, rather than a main result. Therefore we only provide a proof sketch:

Proof Sketch. Suppose that $\text{searchSAT} \in \mathsf{FP}^{\mathsf{NP}}_{tt}$. Then the round complexity in our technical lemmas can be radically reduced:

- The Jeřábek–Korten reduction (**??**) runs in $\mathsf{TIME}[\text{poly}(|T|, |G|)]^{\mathsf{NP}}\begin{bmatrix}\#\text{rounds} = O(\log T)\\ \text{length} = n\end{bmatrix}$.
- The computational history of $\mathsf{TIME}[t(n)]^{\mathsf{NP}}\begin{bmatrix}\#\text{rounds} = r(n)\\ \text{length} = s(n)\end{bmatrix}$ (**??**), hence the valid PCP in the **Completeness** case of **??**, can be computed in $\mathsf{TIME}[\text{poly}(t(n))]^{\mathsf{NP}}\begin{bmatrix}\#\text{rounds} = r(n)\\ \text{length} = s(n)\end{bmatrix}$.

(Here, a minor technical detail is that the computational history Π we find in **??** is no longer the lexicographically largest one, since our $\mathsf{FP}^{\mathsf{NP}}_{tt}$ algorithms for $\mathrm{searchSAT}$ does not necessarily return the lexicographically largest satisfying assignment. However, Π is still a valid computational history with a lexicographically largest possible length-$r'(n)$ prefix. One can also verify that the proof of **??** only requires Π_x to have the largest possible length-$r'(n)$ prefix.)

- Hence, in our instance-wise hardness-randomness tradeoff (**??**), given a multi-output function computable in $\mathsf{TIME}[t(n)]^{\mathsf{NP}}\begin{bmatrix}\mathsf{\#rounds} = r(n)\\ \mathsf{length} = s(n)\end{bmatrix}$, Solve can be computed in

$$\mathsf{TIME}[\mathrm{poly}(t(n))]^{\mathsf{NP}}\begin{bmatrix}\mathsf{\#rounds} = r(n) + O(\log t(n))\\ \mathsf{length} = \max\{s(n), k(n)\}\end{bmatrix}.$$

  Recall that Recon makes $O(r(n)\log t(n))$ many queries to the prMA oracle.

- The parameters r_i (round complexity) in our AVOID algorithm (**??**) satisfies that $r_0 = 0$ and $r_{i+1} = r_i + O(\log T_i) = r_i + O(cD^i n_0 + \log n_{i+1})$. Since $n_i = n_0^{C^i}$, by setting C to be a large enough constant relative to $1/\varepsilon$ and D, we will always have $r_i \le n_i^{\varepsilon/2}$ and $\log T_i \le n_i^{\varepsilon/2}$ (for $i \ge 1$). Hence, each output bit of our final AVOID algorithm on input length n_i can be computed in $\mathrm{poly}(n_i)$ time with $O(r_i \log T_i) \le (n_i)^{\varepsilon}$ many adaptive queries to the prMA oracle.

Let L be the language defined by our AVOID algorithm, where the input instances $\{G_N : \{0,1\}^{N/2} \to \{0,1\}^N\}_{N=2^n, n\in\mathbb{N}}$ are truth table generators [KKMP21; CHLR26]; that is, on input $i \in \{0,1\}^n$, $L(i)$ is the i-th output bit of our AVOID algorithm on G_{2^n}. Since our AVOID algorithm is correct on infinitely many input lengths, $L \notin \mathsf{SIZE}[\frac{2^n}{2n}]$. On the other hand, $L(i)$ can be computed by a deterministic $\mathrm{poly}(2^n)$-time algorithm with $(2^n)^{\varepsilon} = 2^{\varepsilon n}$ many adaptive and smart queries to a prMA oracle, hence $L \in \text{smart-}\mathsf{E}^{\mathsf{prMA}[2^{\varepsilon n}]}/_1$.

Furthermore, we claim that $L \in (\mathsf{MA_E} \cap \mathsf{coMA_E})/_{2^{\varepsilon n}}$. The reason is that in the algorithm $\mathsf{Recon}_f(x,i)$ in **??**, only the last prMA query depends on i. Hence, we can simply hardwire the answers of all other queries as an advice string of length $2^{\varepsilon n}$ and run the last query. Moreover, the last query is of the following form: Merlin sends Arthur a circuit C that succinctly represents a PCP proof Π_C, Arthur verifies that Π_C is accepted by the PCP verifier, has the correct prefix, and a certain bit of Π_C (depending on i) is equal to 1; and $L(i) = 1$ if and only if this query accepts. This immediately gives an $\mathsf{MA_E}/_{2^{\varepsilon n}}$ algorithm for computing L. To obtain an $(\mathsf{MA_E} \cap \mathsf{coMA_E})/_{2^{\varepsilon n}}$ algorithm, we simply change "a certain bit of Π_C is equal to 1" to this bit being equal to b in Arthur's acceptance criterion, where $b \in \{0,1\}$ is an input bit. □

It is known that $\mathrm{searchSAT}$ admits an efficient *randomized* algorithm with parallel access to the NP oracle by using the isolation lemma [MVV87; VV86; BCGL92]. Hence, the assumption made in **??** seems plausible. However, *derandomizing* this algorithm appears strictly harder than proving exponential circuit lower bounds for $\mathsf{prMA_E}$. Indeed, we only know that $\mathrm{searchSAT} \in \mathsf{FP}^{\mathsf{NP}}_{tt}$ under lower bounds against *single-valued nondeterministic* (SVN) circuits [AK01; KM02; MV05; SU06]. It is unclear whether such a strong derandomization assumption is necessary, and we feel that it is important to understand the *minimum assumption* needed to put $\mathrm{searchSAT}$ into $\mathsf{FP}^{\mathsf{NP}}_{tt}$.

Question 5.2. *What is the weakest circuit lower bound assumption that implies* $\mathrm{searchSAT} \in \mathsf{FP}^{\mathsf{NP}}_{tt}$*? Can we design an* $\mathsf{FP}^{\mathsf{NP}}_{tt}$ *algorithm for* $\mathrm{searchSAT}$ *under the assumption that* $\mathsf{E}^{\mathsf{NP}}_{tt}$ *requires exponential size* circuits *(instead of, e.g.,* SVN circuits*)?*

It seems equally important, if not more, to understand the minimum assumption for solving the Range Avoidance problem in $\mathsf{FP}^{\mathsf{NP}}_{tt}$. By [Kor21], the assumption that E^{NP} requires exponential size circuits (*not* SVN circuits!) implies an $\mathsf{FP}^{\mathsf{NP}}$ algorithm for AVOID; moreover, this is provably the minimum assumption. Puzzlingly, to the best of our knowledge, the weakest assumptions known to imply an $\mathsf{FP}^{\mathsf{NP}}_{tt}$ algorithm for AVOID are still lower bounds against SVN circuits [KM02; MV05].

Question 5.3. *What is the weakest circuit lower bound assumption that implies* AVOID $\in \mathsf{FP}^{\mathsf{NP}}_{tt}$*?*

It is conceivable that with a better hardness-randomness tradeoff than Jeřábek–Korten, we will be able to obtain a better uniform reconstruction algorithm, hence a better near-maximum circuit lower bound. Let Π be an encoded computational history, then either Π is "hard enough" to solve AVOID, or Π is "easy" so that one can "reconstruct" Π efficiently. Is there a suitable definition of "hardness" so that any "hard enough" Π can be used to solve AVOID in $\mathsf{FP}^{\mathsf{NP}}_{tt}$, while Π being "easy" implies an efficient MA algorithm for reconstructing Π?

Acknowledgments

We thank Lijie Chen for fruitful discussions during early stages of this work. We are grateful to Jingxun Liang, Zhenjian Lu, and Igor C. Oliveira for helpful comments on a draft version of this paper.

References

[AGHK11] Barış Aydınlıoğlu, Dan Gutfreund, John M. Hitchcock, and Akinori Kawachi. "Derandomizing Arthur-Merlin games and approximate counting implies exponential-size lower bounds". In: *Comput. Complex.* 20.2 (2011), pp. 329–366.

[AK01] Vikraman Arvind and Johannes Köbler. "On pseudorandomness and resource-bounded measure". In: *Theor. Comput. Sci.* 255.1-2 (2001), pp. 205–221.

[AKSS95] Vikraman Arvind, Johannes Köbler, Uwe Schöning, and Rainer Schuler. "If NP has polynomial-size circuits, then MA = AM". In: *Theor. Comput. Sci.* 137.2 (1995), pp. 279–282.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. "Proof verification and the hardness of approximation problems". In: *J. ACM* 45.3 (1998), pp. 501–555.

[AS98] Sanjeev Arora and Shmuel Safra. "Probabilistic checking of proofs: A new characterization of NP". In: *J. ACM* 45.1 (1998), pp. 70–122.

[Bab85] László Babai. "Trading group theory for randomness". In: *STOC*. ACM, 1985, pp. 421–429.

[BCGL92] Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby. "On the theory of average case complexity". In: *J. Comput. Syst. Sci.* 44.2 (1992), pp. 193–219.

[BFT98] Harry Buhrman, Lance Fortnow, and Thomas Thierauf. "Nonrelativizing separations". In: *CCC*. IEEE Computer Society, 1998, pp. 8–12.

[Che23] Lijie Chen. "New lower bounds and derandomization for ACC, and a derandomization-centric view on the algorithmic method". In: *ITCS*. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, 34:1–34:15.

[Che25] Lijie Chen. "Nondeterministic quasi-polynomial time is average-case hard for ACC circuits". In: *SIAM J. Comput.* 54.4 (2025), FOCS19-332–FOCS19-397.

[CHLR23] Yeyuan Chen, Yizhi Huang, Jiatu Li, and Hanlin Ren. "Range avoidance, remote point, and hard partial truth table via satisfying-pairs algorithms". In: *STOC 2023*. 2023, pp. 1058–1066.

[CHLR26] Lijie Chen, Shuichi Hirahara, Zeyong Li, and Hanlin Ren. "Symmetric exponential time requires near-maximum circuit size". In: *J. ACM* 73.1 (Feb. 2026). ISSN: 0004-5411.

[CHR24] Lijie Chen, Shuichi Hirahara, and Hanlin Ren. "Symmetric exponential time requires near-maximum circuit size". In: *STOC*. ACM, 2024, pp. 1990–1999.

[CL24] Yilei Chen and Jiatu Li. "Hardness of range avoidance and remote point for restricted circuits via cryptography". In: *STOC 2024*. ACM, 2024, pp. 620–629.

[CLL25] Lijie Chen, Jiatu Li, and Jingxun Liang. "Maximum circuit lower bounds for exponential-time Arthur Merlin". In: *STOC*. ACM, 2025, pp. 1348–1358.

[CLO+23] Lijie Chen, Zhenjian Lu, Igor C. Oliveira, Hanlin Ren, and Rahul Santhanam. "Polynomial-time pseudodeterministic construction of primes". In: *FOCS*. IEEE, 2023, pp. 1261–1270.

[Coo88] Stephen A. Cook. "Short propositional formulas represent nondeterministic computations". In: *Inf. Process. Lett.* 26.5 (1988), pp. 269–270.

[CR08] Venkatesan T. Chakaravarthy and Sambuddha Roy. "Finding irrefutable certificates for S_2^p via Arthur and Merlin". In: *STACS*. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2008, pp. 157–168.

[CR11] Venkatesan T. Chakaravarthy and Sambuddha Roy. "Arthur and Merlin as oracles". In: *Comput. Complex.* 20.3 (2011), pp. 505–558.

[CT21] Lijie Chen and Roei Tell. "Hardness vs randomness, revised: uniform, non-black-box, and instance-wise". In: *FOCS*. IEEE, 2021, pp. 125–136.

[ESY84] Shimon Even, Alan L. Selman, and Yacov Yacobi. "The complexity of promise problems with applications to public-key cryptography". In: *Inf. Control.* 61.2 (1984), pp. 159–173.

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

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to construct random functions". In: *J. ACM* 33.4 (1986), pp. 792–807.

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

[GLW22] Venkatesan Guruswami, Xin Lyu, and Xiuhan Wang. "Range avoidance for low-depth circuits and connections to pseudorandomness". In: *APPROX/RANDOM 2022*. LIPIcs. 2022, 20:1–20:21.

[Gol06] Oded Goldreich. "On promise problems: A survey". In: *Essays in Memory of Shimon Even*. Lecture Notes in Computer Science. Springer, 2006, pp. 254–290.

[GS88] Joachim Grollmann and Alan L. Selman. "Complexity measures for public-key cryptosystems". In: *SIAM J. Comput.* 17.2 (1988), pp. 309–335.

[GS89] Yuri Gurevich and Saharon Shelah. "Nearly linear time". In: *Logic at Botik '89, Symposium on Logical Foundations of Computer Science, Pereslav-Zalessky, USSR, July 3-8, 1989, Proceedings*. Ed. by Albert R. Meyer and Michael A. Taitslin. Vol. 363. Lecture Notes in Computer Science. Springer, 1989, pp. 108–118.

[GS92] Peter Gemmell and Madhu Sudan. "Highly resilient correctors for polynomials". In: *Inf. Process. Lett.* 43.4 (1992), pp. 169–174.

[GZ11] Oded Goldreich and David Zuckerman. "Another proof that BPP $\subseteq$ PH (and more)". In: *Studies in Complexity and Cryptography*. Lecture Notes in Computer Science. Springer, 2011, pp. 40–53.

[Har04] Prahladh Harsha. "Robust PCPs of proximity and shorter PCPs". PhD thesis. Massachusetts Institute of Technology, 2004.

[Hir15] Shuichi Hirahara. "Identifying an honest $\mathsf{EXP}^{\mathsf{NP}}$ oracle among many". In: *CCC*. Vol. 33. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015, pp. 244–263.

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

[ILW23] Rahul Ilango, Jiatu Li, and R. Ryan Williams. "Indistinguishability obfuscation, range avoidance, and bounded arithmetic". In: *STOC 2023*. ACM, 2023, pp. 1076–1089.

[IW97] Russell Impagliazzo and Avi Wigderson. "P = BPP if E requires exponential circuits: derandomizing the XOR lemma". In: *STOC*. ACM, 1997, pp. 220–229.

[Jeř04] Emil Jeřábek. "Dual weak pigeonhole principle, Boolean complexity, and derandomization". In: *Ann. Pure Appl. Log.* 129.1-3 (2004), pp. 1–37.

[Kan82] Ravi Kannan. "Circuit-size lower bounds and non-reducibility to sparse sets". In: *Information and Control* 55.1-3 (1982), pp. 40–56.

[KKMP21] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos H. Papadimitriou. "Total functions in the polynomial hierarchy". In: *ITCS*. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 44:1–44:18.

[KL80] Richard M. Karp and Richard J. Lipton. "Some connections between nonuniform and uniform complexity classes". In: *STOC*. 1980, pp. 302–309.

[KM02] Adam R. Klivans and Dieter van Melkebeek. "Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses". In: *SIAM J. Comput.* 31.5 (2002), pp. 1501–1526.

[Kor21] Oliver Korten. "The hardest explicit construction". In: *FOCS*. IEEE, 2021, pp. 433–444.

[Kor25] Oliver Korten. "Range avoidance and the complexity of explicit constructions". In: *Bull. EATCS* 145 (2025).

[KP24] Oliver Korten and Toniann Pitassi. "Strong vs. weak range avoidance and the linear ordering principle". In: *FOCS*. IEEE, 2024, pp. 1388–1407.

[Kre88] Mark W. Krentel. "The complexity of optimization problems". In: *J. Comput. Syst. Sci.* 36.3 (1988), pp. 490–509.

[Li24] Zeyong Li. "Symmetric exponential time requires near-maximum circuit size: simplified, truly uniform". In: *STOC*. ACM, 2024, pp. 2000–2007.

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

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. "Matching is as easy as matrix inversion". In: *Comb.* 7.1 (1987), pp. 105–113.

[MVW99] Peter Bro Miltersen, N. V. Vinodchandran, and Osamu Watanabe. "Super-polynomial versus half-exponential circuit size in the exponential hierarchy". In: *COCOON*. Vol. 1627. Lecture Notes in Computer Science. Springer, 1999, pp. 210–220.

[MW20] Cody D. Murray and R. Ryan Williams. "Circuit lower bounds for nondeterministic quasi-polytime from a new easy witness lemma". In: *SIAM J. Comput.* 49.5 (2020).

[NW94] Noam Nisan and Avi Wigderson. "Hardness vs randomness". In: *J. Comput. Syst. Sci.* 49.2 (1994), pp. 149–167.

[RSW22] Hanlin Ren, Rahul Santhanam, and Zhikun Wang. "On the range avoidance problem for circuits". In: *FOCS*. IEEE, 2022, pp. 640–650.

[San09] Rahul Santhanam. "Circuit lower bounds for Merlin–Arthur classes". In: *SIAM Journal on Computing* 39.3 (2009), pp. 1038–1061.

[Sch78] Claus-Peter Schnorr. "Satisfiability is quasilinear complete in NQL". In: *J. ACM* 25.1 (1978), pp. 136–145.

[Sha49] Claude E. Shannon. "The synthesis of two-terminal switching circuits". In: *Bell System technical journal* 28.1 (1949), pp. 59–98.

[SU06] Ronen Shaltiel and Christopher Umans. "Pseudorandomness for approximate counting and sampling". In: *Comput. Complex.* 15.4 (2006), pp. 298–341.

[Sud95] Madhu Sudan. *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems*. Vol. 1001. Lecture Notes in Computer Science. Springer, 1995. ISBN: 3-540-60615-7.

[VV86] Leslie G. Valiant and Vijay V. Vazirani. "NP is as easy as detecting unique solutions". In: *Theor. Comput. Sci.* 47.3 (1986), pp. 85–93.

[ZF87] Stathis Zachos and Martin Fürer. "Probabilistic quantifiers vs. distrustful adversaries". In: *FSTTCS*. Lecture Notes in Computer Science. Springer, 1987, pp. 443–455.

A Half-Exponential Circuit Lower Bounds for $\mathsf{E}^{\mathsf{prMA}}$

In this section, we observe that one can prove a half-exponential circuit lower bound for the class $\mathsf{E}^{\mathsf{prMA}}$ using relativizing techniques. This basically follows from the same argument as the Karp–Lipton collapse for $\mathsf{P}^{\mathsf{prMA}}$ [CR11].

Theorem A.1. *Let $f(n)$ be a nice function such that $f(f(n^c)^c) = 2^{o(n)}$ for every constant $c \geq 1$. Then*

$$\mathsf{E}^{\mathsf{prMA}} \not\subseteq \mathsf{SIZE}[f(n)].$$

Moreover, this holds in every relativizing world.

Proof. We need the following two facts (both are relativizing):

- $\mathsf{TIME}[f(n)^{O(1)}]^{\mathsf{prAM}} \not\subseteq \mathsf{SIZE}[f(n)]$. Since $\mathsf{E}^{\mathsf{prAM}} \not\subseteq \mathsf{SIZE}[2^n/n]$, this follows from a padding argument. (Note that $\mathsf{S}_2\mathsf{E}$ contains a language with circuit complexity $\geq 2^n/n$ [CHR24; Li24] and $\mathsf{S}_2\mathsf{E} \subseteq \mathsf{E}^{\mathsf{prAM}}$ [CR08; CR11].)
- If $\mathrm{SAT} \in \mathsf{SIZE}[f(n)]$, then $\mathsf{prAM} \subseteq \mathsf{prMATIME}[f(n^{O(1)})^{O(1)}]$ [AKSS95].

There are two cases. If $\mathrm{SAT} \notin \mathsf{SIZE}[f(n)]$, then clearly $\mathsf{E}^{\mathsf{prMA}} \not\subseteq \mathsf{SIZE}[f(n)]$ as well. Otherwise, since $\mathrm{SAT} \in \mathsf{SIZE}[f(n)]$, we have $\mathsf{prAM} \subseteq \mathsf{prMATIME}[f(n^{O(1)})^{O(1)}]$. It follows that

$$\mathsf{TIME}[f(n)^{O(1)}]^{\mathsf{prAM}} \subseteq \mathsf{TIME}[f(n)^{O(1)}]^{\mathsf{prMATIME}[f(n^{O(1)})^{O(1)}]} \subseteq \mathsf{TIME}[f(f(n^{O(1)})^{O(1)})^{O(1)}]^{\mathsf{prMA}} \subseteq \mathsf{E}^{\mathsf{prMA}},$$

hence $\mathsf{E}^{\mathsf{prMA}} \not\subseteq \mathsf{SIZE}[f(n)]$. □

It is unclear whether such a relativizing lower bound can be proved for $\mathsf{smart\text{-}E}^{\mathsf{prMA}}/_1$.

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il