Maximum Circuit Lower Bounds for Exponential-Time Arthur Merlin

Maximum Circuit Lower Bounds for Exponential-Time Arthur Merlin

Lijie Chen
lijiechen@berkeley.edu
UC Berkeley
Berkeley, California, USA

Jiatu Li
jiatuli@mit.edu
MIT
Cambridge, Massachusetts, USA

Jingxun Liang
jingxunl@andrew.cmu.edu
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Abstract

We show that the complexity class of exponential-time Arthur Merlin with sub-exponential advice ($\mathsf{AMEXP}_{/2^{n^\varepsilon}}$) requires circuit complexity at least $2^n/n$. Previously, the best known such near-maximum lower bounds were for symmetric exponential time by Chen, Hirahara, and Ren (STOC'24) and Li (STOC'24), or randomized exponential time with MCSP oracle and sub-exponential advice by Hirahara, Lu, and Ren (CCC'23).

Our result is proved by combining the recent iterative win-win paradigm of Chen, Lu, Oliveira, Ren, and Santhanam (FOCS'23) together with the uniform hardness-vs-randomness connection for Arthur-Merlin protocols by Shaltiel-Umans (STOC'07) and van Melkebeek-Sdroievski (CCC'23). We also provide a conceptually different proof using a novel "critical win-win" argument that extends a technique of Lu, Oliveira, and Santhanam (STOC'21).

Indeed, our circuit lower bound is a corollary of a new explicit construction for properties in coAM. We show that for every dense property $P \in \mathsf{coAM}$, there is a quasi-polynomial-time Arthur-Merlin protocol with short advice such that the following holds for infinitely many n: There exists a canonical string $w_n \in P \cap \{0,1\}^n$ so that (1) there is a strategy of Merlin such that Arthur outputs w_n with probability 1 and (2) for any strategy of Merlin, with probability 2/3, Arthur outputs either w_n or a failure symbol $\perp$. As a direct consequence of this new explicit construction, our circuit lower bound also generalizes to circuits with an $\mathsf{AM} \cap \mathsf{coAM}$ oracle. To our knowledge, this is the first unconditional lower bound against a strong non-uniform class using a hard language that is only "quantitatively harder".

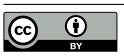
CCS Concepts

• **Theory of computation** → **Circuit complexity**; **Complexity classes**.

Keywords

circuit lower bounds, derandomization, Arthur Merlin protocols

ACM Reference Format:
Lijie Chen, Jiatu Li, and Jingxun Liang. 2025. Maximum Circuit Lower Bounds for Exponential-Time Arthur Merlin. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing (STOC '25), June 23–27, 2025, Prague, Czechia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3717823.3718224



STOC '25, Prague, Czechia

ACM ISBN 979-8-4007-1510-5/25/06
https://doi.org/10.1145/3717823.3718224

1 Introduction

Proving circuit lower bounds for uniform complexity classes is one of the central problems in complexity theory. Despite that (following a simple counting argument) almost all Boolean functions $f\colon \{0,1\}^n \to \{0,1\}$ require $2^n/n$-size circuit to compute [51], the progress on proving explicit circuit lower bounds has been relatively slow.

The progress on proving *exponential* lower bounds (thereby matching Shannon's counting argument) is even more limited. Kannan [35] proved that $\Sigma_3\mathsf{E} \cap \Pi_3\mathsf{E}$ requires maximum $(2^n/n)$ size circuits, the complexity of the hard function was later improved to $\Delta_3\mathsf{E} = \mathsf{E}^{\Sigma_2\mathsf{P}}$ by Miltersen, Vinodchandran, and Watanabe [42], via a simple binary search argument.

The limited progress was due to the lack of techniques for proving exponential-size circuit lower bounds. There has been much progress on proving super-polynomial-size circuit lower bounds (see Section 2.1 for details), which all follow the famous "win-win" paradigm. However, it has been observed [42] that this "win-win" paradigm could not give exponential-size lower bounds.[1]

A recent work by Chen, Hirahara, and Ren [18], following a new technique called "iterative win-win paradigm" (originally developed by [20] for pseudo-deterministic construction of primes), proved that $\Sigma_2\mathsf{E}$ (as well as $\mathsf{S}_2\mathsf{E}$ with one-bit advice) requires $2^n/n$-size circuits. Their results were later simplified and strengthened by Li [40], showing that $\mathsf{S}_2\mathsf{E}$ (with no advice) requires maximum circuit complexity on all but finitely many input lengths. With a different approach, Hirahara, Lu, and Ren [31] also proved a maximum circuit lower bound for $\mathsf{BPE}^{\mathsf{MCSP}}$ with $2^{\varepsilon n}$ bits of advice.

One surprising feature of the recent work [18, 40] is that their proofs *relativizes*. Given the limitations of relativizing proofs (for example, no relativizing proofs can prove the super-polynomial-size lower bound for MAEXP [14]), a natural question is whether we can combine the techniques behinds [18, 40] (e.g., the iterative win-win paradigm) with *non-relativizing* proof techniques to make further progress on proving exponential-size circuit lower bounds.

1.1 Our Results

1.1.1 Maximum Circuit Lower Bound for AMEXP. In this work, we make progress on the question above by combining the non-relativizing techniques of *arithmetization* (specifically, the uniform hardness vs. randomness trade-off for AM [50, 53]) and the iterative win-win paradigm [18, 20]. We show that $\mathsf{AMEXP} \cap \mathsf{coAMEXP}$

[1] We note that exponential-size circuit lower bounds have more applications compared to super-polynomial-size circuit lower bounds: $2^{\Omega(n)}$-size lower bounds for E imply that P = BPP [33, 44], while super-polynomial lower bounds for E only give that BPP can be derandomized in sub-exponential time.

with a sub-exponential amount of advice requires maximum circuit complexity.

Theorem 1.1. $(\mathsf{AMEXP} \cap \mathsf{coAMEXP})_{/2^{n^\varepsilon}} \not\subset \mathsf{SIZE}[2^n/n]$ *for any constant* $\varepsilon \in (0,1)$.

Compared with previous works [18, 40] where the same maximum circuit lower bound was proved for $\mathsf{S_2E}$, our lower bound is proved for the smaller class $\mathsf{AMEXP} \cap \mathsf{coAMEXP}$. Indeed, $\mathsf{S_2E}$ is a randomized version of $\mathsf{E^{NP}}$, while $\mathsf{AMEXP} \cap \mathsf{coAMEXP}$ is a randomized version of $\mathsf{NEXP} \cap \mathsf{coNEXP}$. So in a sense, our result is much closer to NEXP than the previous one. On the other hand, our lower bound for $\mathsf{AMEXP} \cap \mathsf{coAMEXP}$ requires a sub-exponential amount of advice, while the lower bound in [40] requires no advice.

Moreover, our circuit lower bound not only holds for Boolean circuits, but also generalizes to circuits with an $\mathsf{AM} \cap \mathsf{coAM}$ oracle[2].

Theorem 1.2. *For any language* $L \in \mathsf{AM} \cap \mathsf{coAM}$, $(\mathsf{AMEXP} \cap \mathsf{coAMEXP})_{/2^{n^\varepsilon}} \not\subset \mathsf{SIZE}^L[2^n/n]$.

To the best of our knowledge, this is the first unconditional lower bound against a strong non-uniform class with a hard language that is only quantitatively harder (in terms of time complexity) than the non-uniform class.[3] In comparison, most of the existing unconditional lower bounds require qualitatively stronger hard languages; for instance, $\Sigma_2\mathsf{E} \not\subseteq \mathsf{SIZE}[2^n/n]$ [18, 40] requires a hard language in a higher level of the exponential-time hierarchy.

This lower bound can also be interpreted as a trade-off between time and non-uniformity. It means that it is impossible to speed up an arbitrary $(\mathsf{AM} \cap \mathsf{coAM})$-style algorithm with relatively short non-uniform advice using even near-maximum non-uniform advice.

Arthur-Merlin classes. An Arthur-Merlin protocol for a language L [9, 28] is a two-party constant-round interactive proof system where a computationally unbounded prover (called Merlin) aims to convince a probabilistic polynomial-time verifier (called Arthur) that $x \in L$ for a string x owned by both parties. A *strategy* of Merlin is a function that given a partial transcript of the protocol, outputs the next message to send to Arthur. The protocol should be *sound* in the sense that the verifier rejects any strategy of Merlin with high probability if $x \notin L$, and *complete* in the sense that there is a strategy of Merlin that could convince Arthur with high probability if $x \in L$.

The class $(\mathsf{AMEXP} \cap \mathsf{coAMEXP})_{/\alpha(n)}$ consists of languages L such that both L and $\overline{L}$ are decidable by $2^{\mathrm{poly}(n)}$-time Arthur-Merlin protocols where both parties receive an $\alpha(n)$-bit non-uniform advice on input length n. We note that there are multiple formal definitions of $\mathsf{AMEXP} \cap \mathsf{coAMEXP}$ with advice depending on the behavior of the AM protocols given *incorrect advice*; see the full version [19] for our definition and related discussion.

[2]Note that $\mathsf{SIZE}^L[s(n)]$ refers to languages that admit a family of size-$s(n)$ circuits with L-oracle gates. Since L oracle gates could have unbounded fan-in, the size of the circuits is defined as the number of wires. For a concrete example, one may think of a factoring oracle, i.e., given positive integers N and k encoded in binary, it decides whether there is a divisor d of N such that $2 \le d \le k$.

[3]Here we only consider lower bounds against non-uniform classes that are at least as strong as general Boolean circuits. In restricted circuit settings, it is known (for instance) that exponential-size uniform-$\mathsf{AC^0}$ requires sub-exponential-size non-uniform $\mathsf{AC^0}$ circuits, which follows from the $\mathsf{AC^0}$ upper and lower bound for the parity function [3, 24, 30, 54].

1.1.2 Hitting Dense coAM *Properties.* Recent developments on maximum circuit lower bounds highlight a folklore view that proving a circuit lower bound for exponential-time classes is equivalent to designing an algorithm that explicitly constructs hard truth table [18, 39, 40].

More formally, consider the property Π_{hard} defined as the set of strings that are not truth tables of circuits of size at most $2^n/n$. If (for instance) there is a deterministic polynomial-time algorithm that given 1^{2^n} outputs a string $tt_n \in \Pi_{\mathsf{hard}} \cap \{0,1\}^{2^n}$ for infinitely many n, we can define L_{hard} as:

$$x \in L_{\mathsf{hard}} \iff \text{the } x\text{-th bit of } tt_{|x|} \text{ is 1},$$

so that $L_{\mathsf{hard}} \in \mathsf{E} \coloneqq \mathsf{DTIME}[2^n]$ (by calling the deterministic algorithm) and $L_{\mathsf{hard}} \notin \mathsf{SIZE}[2^n/n]$ (by the definition of Π_{hard}).

This connection can be adapted to AMEXP lower bounds with suitable technical definitions: If there is a *single-valued* Arthur-Merlin protocol (with short non-uniform advice) that given 1^{2^n} outputs a string in $\Pi_{\mathsf{hard}} \cap \{0,1\}^{2^n}$ for infinitely many n, we can obtain the lower bound in Theorem 1.1. Similarly, we can obtain the lower bound in Theorem 1.2 if we replace Π_{hard} with the property Π^L_{hard} that contains maximally hard truth tables against L-oracle circuits. Here, a single-valued Arthur-Merlin protocol outputs a *canonical* string with high probability if Arthur does not reject during the interaction (see the full version [19] for a formal definition).

Note that Π_{hard} and Π^L_{hard} are decidable in coAM. Moreover, by Shannon's counting argument [51], Π_{hard} and Π^L_{hard} are both *dense* properties. Indeed, both our lower bounds follow from the following general result: We show that for every dense coAM property P, there is a single-valued Arthur-Merlin protocol with short non-uniform advice that given 1^n outputs a canonical $x_n \in P \cap \{0,1\}^n$ for infinitely many $n \in \mathbb{N}$. Formally:

Theorem 1.3 (Main Theorem)**.** *Let* $k > 1$ *be an arbitrary constant and* $P \in \mathsf{coAM}$ *be a language such that* $|P_n| \ge 2^{n-1}$ *for every* $n \in \mathbb{N}$. *There is a sequence of strings* $\{x_n \in \{0,1\}^n\}_{n\in\mathbb{N}}$ *and an Arthur-Merlin algorithm* A *that runs in time* $2^{\log^{O(k)} n}$ *and takes* $2^{\log^{1/k} n}$ *bits of advice* $\{\alpha_n\}_{n\in\mathbb{N}}$ *such that the following properties hold:*

- (Conformity). *For every* $n \in \mathbb{N}$, *there is a strategy of Merlin such that* $\Pr[A(1^n, \alpha_n) = x_n] = 1$.
- (Resiliency). *For every* $n \in \mathbb{N}$ *and any string* $\zeta_n \in \{0,1\}^{2^{\log^{1/k} n}}$, *there is a string* $y_n \in \{0,1\}^n$ *such that for any strategy of Merlin,* $\Pr[A(1^n, \zeta_n) \in \{y_n, \perp\}] \ge 2/3$.
- (Hitting). *For infinitely many* $n \in \mathbb{N}$, $x_n \in P$.

Here, conformity and resiliency formalize the intuition of a non-uniform single-valued Arthur-Merlin algorithm with arbitrary (i.e. possibly non-Boolean) output. We also note that, besides being single-valued, our Arthur-Merlin protocol enjoys an additional property: The AM protocol is partially single-valued (i.e. it either rejects or outputs a canonical string) even when given incorrect advice. This property is crucial for proving our circuit lower bounds and may also be useful in other applications.

Due to space constraints, we provide only a high-level technical overview in this version. Full details, including proofs and extended discussions, are available in the full version [19].

1.2 Related Works on Explicit Construction Algorithms

Our main theorem (see Theorem 1.3) is also interesting in its own right as an unconditional explicit construction algorithm for any dense property in coAM, contributing to a recent program of solving explicit construction problems using techniques from complexity theory. We provide a summary of related works from the perspective of explicit construction problems for dense properties in P, BPP, and stronger classes.

- Dense property in P: Chen et al. [20] (built on an earlier result [45]) proved that for any dense property Π decidable in P, there is a randomized polynomial-time algorithm that for infinitely many n, it outputs a canonical string in $\Pi \cap \{0,1\}^n$ with high probability given 1^n.[4] In particular, there is an efficient algorithm that constructs a canonical n-bit prime given 1^n for infinitely many n.[5]
- Dense property in BPP: Oliveira and Santhanam [45] proved similar pseudodeterministic algorithms exist for any dense properties decidable in BPP, but only achieves subexponential running time. Subsequently, Lu, Oliveira, and Santhanam [41] constructed a polynomial-time pseudodeterministic algorithm that takes an $O(n^\epsilon)$-bit advice for the same problem.
- Range avoidance: Range avoidance problem [38, 39, 47] refers to the search problem that given a multi-output circuit $C : \{0,1\}^n \to \{0,1\}^m$ satisfying $m > n$, outputs a string $y \in \{0,1\}^m$ outside of the range of C, i.e., $C^{-1}(y) = \varnothing$. Deterministic (and pseudodeterministic) algorithms for range avoidance are known to imply circuit lower bounds. Chen, Hirahara, and Ren [18] proved that there is a single-valued search-$\mathsf{S_2P}_{/1}$ algorithm[6] for range avoidance that works on infinitely many input lengths, which was improved by Li [40] to a fully uniform search-$\mathsf{S_2P}$ algorithm (i.e. avoiding the 1-bit advice) that works on all input lengths.[7]

Note that the range avoidance problem is a non-unary explicit construction problem, i.e., the input is not of form 1^n. One can also consider the unary version of it, i.e., the input circuit C is restricted to uniform family $\{C_n\}_{n\in\mathbb{N}}$ of circuits.[8] Solving unary range avoidance is just to hit the dense coNP property Π_{avoid} defined as

$$\Pi_{\mathsf{avoid}} \coloneqq \left\{ y \in \{0,1\}^n \mid n \in \mathbb{N}, C_n^{-1}(y) = \varnothing \right\}.$$

Therefore, our main theorem extends the sequence of works to the explicit construction of properties beyond (unary) range avoidance to arbitrary dense coNP properties.[9]

[4] This is also known as a pseudodeterministic algorithm [26], i.e., a randomized algorithm that outputs canonical solutions with high probability.

[5] Note that primality is a dense property by the prime number theorem, and is decidable in P by the AKS primality test [2].

[6] $\mathsf{S_2P}$ is a subclass of $\mathsf{ZPP}^{\mathsf{NP}} \subseteq \Sigma_2\mathsf{P}$; interested readers are referred to [18] and references therein.

[7] There have also been many works on solving special cases of the range avoidance problem [22, 25, 29, 47], as well matrix rigidity [4, 13] (which is reducible to range avoidance, see [39]).

[8] That is, there is a polynomial-time Turing machine outputting C_n given 1^n. Note that (pseudo-)deterministic algorithms for the unary range avoidance problem suffice to imply circuit lower bounds (see, e.g., [18, 47]).

[9] We also note that the (non-unary) range avoidance problem is unlikely to be solvable by even non-uniform nondeterministic search algorithms [23, 32]; for this reason, it is unlikely to improve the search-$\mathsf{S_2P}$ algorithm of [18, 40] to a single-valued AM algorithm (even with advice) as otherwise one can derandomize AM using non-uniformity to obtain a non-uniform nondeterministic search algorithm for the range avoidance problem.

[10] Recall that the result of [45] (and the subsequent improvement from [20]) can be used to hit any dense property in P, and the construction of primes follows from the AKS primality test [2] and the prime number theorem. For concreteness, we stick to the construction of large primes in the introduction.

2 Technical Overview

In this section, we will first revisit the important conceptual ideas and technical ingredients leading to recent breakthroughs in pseudodeterministic constructions [20] and exponential circuit lower bounds [18, 40]. We will explain the *iterative win-win paradigm* in Section 2.1 introduced in [18, 20], which will be adapted to our setting and sketch the *first* proof of Theorem 1.3 (see Sections 2.2 and 2.3). We will then introduce an alternative technique called the *criticial win-win argument* that sketchs the *second* proof of Theorem 1.3 in Section 2.4. Readers who are already familiar with the iterative win-win paradigm can skip directly to Section 2.2.

The main reason that we include both proofs of Theorem 1.3 is that they are technically incomparable, and they highlight two conceptually different approaches to bypassing the half-exponential barrier (see Section 2.1 and [18, 42]). We believe that these two techniques (or combined in some way) will lead to stronger results in circuit lower bounds and explicit construction problems.

2.1 Bypassing the Half-exponential Barrier: The Iterative Win-win Paradigm

Before delving into our techniques of proving Theorem 1.3, we first review why a vanilla win-win argument is unable to prove exponential circuit lower bounds, and how a recent *iterative win-win paradigm* bypasses this barrier.

Win-win arguments. Win-win arguments are widely used in complexity theory to prove unconditional circuit lower bounds against general circuits. The idea goes as follows. Given an (inefficient) brute-force algorithm BF for finding a hard truth table (e.g. via diagonalization), we find a suitable problem Q and ask whether Q has large circuit complexity. If so, we obtain a circuit lower bound for Q; otherwise, we *speedup* the brute-force algorithm BF using the efficient circuit for Q and thus obtain a non-trivial algorithm for finding hard truth tables, which also leads to a circuit lower bound.

A standard example is Kannan's theorem [35]. The brute-force algorithm BF is a language $L_{\mathsf{diag}} \in \Sigma_3\mathsf{E}$ that requires super-polynomial size circuits via diagonalization, and the problem Q is SAT. If $\mathsf{SAT} \notin \mathsf{P}_{/\mathsf{poly}}$ we already obtain a circuit lower bound for $\mathsf{NP} \subseteq \Sigma_2\mathsf{E}$; otherwise, Karp-Lipton theorem [36] shows that the polynomial hierarchy collapses to its second level (in particular, $\Sigma_2\mathsf{E} = \Sigma_3\mathsf{E}$), and thus $L_{\mathsf{diag}} \in \Sigma_2\mathsf{E}$. In both cases, we obtain a super-polynomial circuit lower bound for $\Sigma_2\mathsf{E}$. The lower bound can be improved to (for example) $\mathsf{S_2E}$ [16] using the same approach by invoking a stronger Karp-Lipton style collapse.

Indeed, a similar argument can be adapted to solve other explicit construction problems beyond circuit lower bounds, with an interesting example of pseudodeterministic constructions of large primes [41, 45].[10] Recall that a pseudodeterministic construction refers to a randomized algorithm that (given 1^n) outputs a *canonical* n-bit primes with high probability.

Instead of the Karp-Lipton collapse, [45] uses a reconstructive pseudorandom generator from the hardness-vs-randomness paradigm [34, 52]. The brute-force algorithm BF enumerates all n-bit strings and outputs the first prime, which can be implemented in PSPACE. It then asks whether a PSPACE-complete problem L_{TV} is in BPP. If so, PSPACE = BPP and thus BF can be implemented by a polynomial-time randomized algorithm. Otherwise, we can obtain, from the hardness of L_{TV}, a pseudorandom generator with seed length n^{ε} that fools uniform algorithms via the uniform hardness-vs-randomness paradigm [34, 52]. Since primes are dense and the primality test is in P [2], the pseudorandom generator must hit an n-bit prime, and thus we can output a canonical prime in sub-exponential time by enumerating all the seeds and outputting the first prime from the outputs of the pseudorandom generator.

The half-exponential barrier. The win-win arguments we mentioned above all run into a "half-exponential barrier", as pointed out in [42] (also see [18]). Intuitively, it means that the two cases in the win-win argument are competing with each other, which prevents us from proving exponential lower bounds (or polynomial-time explicit construction) in both cases.

Take Kannan's theorem as an example. Suppose that we want to prove an exponential circuit lower bound for $\Sigma_2\mathsf{E}$. If we perform a win-win argument on whether $\mathsf{SAT} \notin \mathsf{SIZE}[s(n)]$ for (say) $s(n) = 2^{n^{\varepsilon}}$ rather than $s(n) = n^{\omega(1)}$ to improve the lower bound when $\mathsf{SAT} \notin \mathsf{SIZE}[s(n)]$, we will encounter a sub-exponential overhead for the Karp-Lipton collapse in the case that $\mathsf{SAT} \in \mathsf{SIZE}[s(n)]$, which prevents us from proving an exponential lower bound for $\Sigma_2\mathsf{E}$. By a careful calculation of parameters, it turns out that the best we can hope is to set $s(n)$ such that $s(s(\mathrm{poly}(n))) \le 2^n$, leading to a so-called *half-exponential* lower bound. Similarly, [41, 45] can only construct large primes psueodeterministically in half-exponential time.

Perspective: Construction of dense property and an input-length-pair-wise win-win argument. It turns out that a new interpretation of the win-win argument in [41, 45] serves as the key idea for bypassing the half-exponential barrier [18, 20, 40]. Concretely:

- Both tasks above (i.e. proving circuit lower bounds and generating large primes) can be viewed as designing an efficient single-valued algorithm to hit a uniform dense property P. For the construction of primes, P is the set of primes and is decidable in P [2]; for exponential circuit lower bounds, P is the set of strings that are not the truth tables of $2^n/n$-size circuits, which is known to be in coNP. This view is highlighted in recent works on the range avoidance problem, see, e.g., [22, 39, 47].[11]
- Instead of identifying a problem Q and designing two algorithms for two possible outcomes of whether Q is hard or easy (the win-win argument), we can indeed interpret the standard win-win argument as designing *one* algorithm unifying the two algorithms so that it always "wins". That is, the new algorithm always correctly outputs a canonical string in P on infinitely many input lengths.

  In more detail, let $A_{\mathsf{n}}, A_{\mathsf{m}}$ be two different algorithms, the unified algorithm considers two disjoint infinite sets of input lengths $\{n_0, n_1, \dots\}$ and $\{m_0, m_1, \dots\}$. It simulates $A_{\mathsf{n}}(1^{n_i})$ on each input length n_i ($i \in \mathbb{N}$), and simulates $A_{\mathsf{m}}(1^{m_i})$ on each input length m_i.[12] These two algorithms are designed so that for each $i \in \mathbb{N}$, either it (simulating A_{n}) is correct on the input length n_i, or it (simulating A_{m}) is correct on the input length m_i. In other words, it performs an "input-length-pair-wise" win-win argument between each pair of input lengths n_i and m_i. Indeed, this view has been found useful in proving circuit lower bounds against ACC^0 [17] (following [43]).

For concreteness, consider $n_{i+1} = 2^{n_i}$ and $m_i = 2^{n_i^{0.1}}$. Let BF be a brute-force algorithm for hitting the property P that runs in (say) exponential time.[13] Intuitively, the unified algorithm performs a win-win argument on each pair (n_i, m_i) of input lengths by considering whether the "computation history" of $\mathsf{BF}(1^{n_i})$ is "hard". It works differently according to the input length:

- On the input length m_i ($i \in \mathbb{N}$), it assumes that the "computation history" of $\mathsf{BF}(1^{n_i})$ is "hard", and utilizes this hardness to hit the dense property P (say in time $2^{O(n_i)} = 2^{m_i^{o(1)}}$) with a suitable hardness-vs-randomness framework (e.g. [34, 44, 52]).
- On the input length n_i ($i \in \mathbb{N}$), it assumes that the "computation history" of $\mathsf{BF}(1^{n_i})$ is not "hard", and tries to speed up the brute-force algorithm $\mathsf{BF}(1^{n_i})$.

If we can figure out suitable definitions of "computation history" and "hardness", and apply a hardness-vs-randomness framework accordingly, this algorithm will work on either the input length n_i or m_i for each $i \in \mathbb{N}$, unifying the two cases of the win-win argument. Indeed, one can verify that [45] can be interpreted as such an algorithm using the PRG in [52].

Insight: amortizing the cost of win-win. Once adapted to the input-length-pair-wise view on the win-win argument, it is natural to ask whether it is beneficial to perform a "win-win-win argument" on 3-tuples of input lengths (rather than pairs), or even play with more "wins". The rationale is that the half-exponential barrier comes with the "self-competing" nature of two possible outcomes of the win-win argument, and if we introduce more outcomes and amortize the overhead among different input lengths, we may achieve better bounds.

Iterative win-win paradigm. Indeed, the answer is positive. A framework called the *iterative win-win paradigm* was introduced in [20], which improved the half-exponential time pseudodeterministic algorithm in [41, 45] to a *polynomial-time* algorithm. Subsequently, [18] proved maximum circuit lower bounds for $\Sigma_2\mathsf{E}$ and $\mathsf{S}_2\mathsf{E}_{/1}$ following the same paradigm, improving the half-exponential lower bounds of Kannan [35].

[11] In particular, hitting the set of strings that are not the truth tables of small circuits is reducible to the range avoidance problem [39], which serves as the main technical ingredient leading to [18, 40].

[12] Recall that the explicit construction problem has a unary input. Also, note that we should define these two sets so that the algorithm can decide uniformly whether the input length is one of n_i or one of m_i.

[13] The complexity measure may not be time complexity, but (for instance) alternation in Kannan's theorem [35]. We use time complexity only for illustrative purposes.

As the name indicates, the iterative win-win approach performs a win-win argument iteratively with *super-constantly* many cases instead of only two cases.[14] The basic idea is as follows. Let $n_0, n_1, \ldots, n_\ell$ be an increasing sequence of input lengths where n_0 is sufficiently large. We start with the brute-force algorithm BF that runs in (say) exponential-time, and ask whether its "computation history" on input length n_0 is "moderately hard".

- (*Win*). If the computation history on input length n_0 is not even "moderately hard", we can improve the brute-force algorithm BF (on input length n_0) to polynomial time.
- (*Improve*). Otherwise, we utilize the moderately hard computation history to obtain a *moderately better* algorithm on input length n_1 following a hardness-vs-randomness framework[15], treat it as the new brute-force algorithm, and proceeds the same win-win argument on input length n_1.

Note that the exact meaning of a "moderately hard" "computation history" will be clear when we instantiate the framework with [18, 20].

The key observation is that the improvements in the case (*Improve*) could accumulate over iterations, and thus if the sequence of input lengths $n_0, n_1, \ldots, n_\ell$ grows sufficiently fast (say $n_{i+1} = n_i^\beta$ for a large constant β) and ℓ is sufficiently large (say $\ell = \log n_0$), the final "brute-force" algorithm on input length n_ℓ will be very efficient. By splitting the input lengths into infinitely many disjoint sequences $\langle n_0, n_1, \ldots, n_\ell \rangle$, we can obtain a *polynomial-time algorithm* that is guaranteed to be correct on at least one n_i for each of such sequence.

Instantiations of iterative win-win. The biggest technical challenge is to identify the exact meaning of being a "moderately hard" computation history and find the hardness-vs-randomness framework allowing us to gain improvement with the hardness of the computation history.

Construction of primes. For the pseudodeterministic construction of primes [20], the "win-or-speedup" is carried out by the uniform non-black-box hardness-vs-randomness framework developed by Chen and Tell [21], which builds on the interactive proof system due to Goldwasser, Kalai, and Rothblum [27]. Intuitively, the Chen-Tell framework allows us to construct a hitting set H_C from an *inefficient parallel computation* C.[16], such that given a dense polynomial-time decidable property that H_C fails to hit, one can simulate the computation C by a randomized polynomial-time algorithm.[17] The win-win argument goes as follows.

- (*Win*). Recall that the brute-force algorithm BF_0 for finding the smallest n_0-bit prime is highly parallel, we instantiate the Chen-Tell framework with the computation $\mathsf{BF}_0(1^{n_0})$ and obtain an HSG (hitting set generator) H_0 with output length n_1. If H_0 fails to hit any n_1-bit prime, we can simulate $\mathsf{BF}_0(1^{n_0})$ by a randomized polynomial-time algorithm.
- (*Improve*). If H_0 hits an n_1-bit prime, we obtain a slightly more efficient algorithm $\mathsf{BF}_1(1^{n_1})$ by enumerating H_0 and returning the first n_1-bit prime.

Notice that the slightly more efficient algorithm BF_1 in (*Improve*) case is still highly parallel, one can iteratively perform the win-win argument as discussed above to obtain $\mathsf{BF}_2, \mathsf{BF}_3, \ldots, \mathsf{BF}_\ell$ on input lengths $n_2, n_3, \ldots, n_\ell$, where (according to the time complexity of the hitting set generator) the running time of BF_{i+1} is polynomially bounded by the running time of BF_i. This means that BF_i runs in time $\exp\{\mathrm{poly}(n_0) \cdot \exp(O(i))\}$. By setting $n_{i+1} \coloneqq n_i^\beta$ for a large constant β and $\ell \coloneqq \lceil \log n_0 \rceil$, then

$$n_\ell = n_0^{\beta^{\lceil \log n_0 \rceil}} \geq 2^{\beta^{\log n_0}} = 2^{n_0^{\log \beta}}$$

the running time of BF_ℓ would be

$$\exp\{\mathrm{poly}(n_0) \cdot \exp(O(\log n_0))\} = \exp(\mathrm{poly}(n_0)) \leq \mathrm{poly}(n_\ell).$$

Therefore, we can obtain a polynomial-time (pseudodeterministic) algorithm that correctly finds primes on infinitely many input lengths: Either BF_ℓ is correct on the input 1^{n_ℓ}, or for some $i < \ell$ we can "win" by simulating $\mathsf{BF}_i(1^{n_i})$ using a randomized polynomial-time algorithm.

Circuit lower bounds. Recall that proving circuit lower bounds is equivalent to an explicit construction of canonical hard truth tables (and the property consisting of hard truth tables is a dense property). For the maximum circuit lower bound for $\Sigma_2\mathsf{E}$ (and $\mathsf{S}_2\mathsf{E}$) [18], the win-win argument utilizes a reduction called *hardness condensation* (see, e.g., [15]), which takes a truth table of length $T = 2^n$ hard against size S and constructs a truth table of length $T' = 2^{n'}$ hard against size S', where n' could be much smaller than n but $S' \approx S$.[18]

Let $n_0, n_1, \ldots, n_\ell$ be a sequence of input lengths. We also assume all n_i's are powers of 2 and let m_i be such that $n_i = 2^{m_i}$. On input length $n = 2^m$, we want to find a canonical truth table of length n that is hard against $2^m/m$ size circuits. Starting with a brute-force algorithm BF_0 that enumerates and returns the first hard truth table, we perform the following win-win argument:

- (*Win*). If the computation history of BF_0 on input length $n_0 = 2^{m_0}$ (which is of length exponential in n_0) is the truth table of a $\mathrm{poly}(n_1)$-size circuit, we can simulate the brute-force algorithm in $\Sigma_2\mathsf{P}$ (and even $\mathsf{S}_2\mathsf{P}_{/1}$, with a more careful argument) by guessing the circuit and verifying the computation ($\star$).
- (*Improve*). Otherwise, the computation history of BF_0 has circuit complexity $\mathrm{poly}(n_1) \gg 2^{m_1}/m_1$. By hardness condensation ($\diamond$), we can obtain a maximally hard truth table of length $n_1 = 2^{m_1}$, which is moderately more efficient than the brute-force algorithm.

A careful inspection of a hardness condensation algorithm implicit in [39] shows that it fits perfectly into the win-win argument: both

[14] Note that the new perspective is crucial as it is unclear how to come up with super-constantly many cases and specify super-constantly many different algorithms in the standard win-win arguments.

[15] The intuition is that the hardness-vs-randomness framework will provide a pseudorandom generator (or hitting set generator, HSG for short) over $\{0,1\}^{n_1}$ with a non-trivial seed length from the computation history of $\mathsf{BF}(1^{n_0})$ so that the "moderately better" algorithm on the input length n_1 can enumerate all the seeds and find out a string with the property $P \cap \{0,1\}^{n_1}$.

[16] More formally, the inefficient parallel computation is modeled as a highly uniform, large size (e.g. exponential size), and low-depth layered circuit, see [20, 21] for a formal definition.

[17] The original Chen-Tell hitting set generator only allows a quasi-polynomial-time algorithm for simulating C, which is improved in [20] using a better pseudorandom generator from [49].

[18] To see that this fits into the iterative win-win paradigm we described above, one can also view hardness condenseation as a hardness-vs-randomness framework, as it solves a derandomization task of generating a hard truth table (of length $T' \ll T$) using a hard truth table (of length T).

the verification of its computation ($\star$) and the hardness condensation procedure ($\diamond$) can be implemented into $\Sigma_2\mathsf{P}$ and even $\mathsf{S}_2\mathsf{P}_{/1}$. Moreover, by defining the computation history carefully, we can iteratively perform the win-win argument above to obtain algorithms $\mathsf{BF}_1, \mathsf{BF}_2, \ldots, \mathsf{BF}_\ell$ on input lengths $n_1, n_2, \ldots, n_\ell$, where $\mathsf{BF}_\ell(1^{n_\ell})$ runs in polynomial time, and thus an algorithm that correctly finds hard truth tables on infinitely many input lengths.

A "just-win" proof of $\mathsf{S}_2\mathsf{E}$ *lower bounds.* In a recent paper, Li [40] obtained an almost-everywhere and fully uniform maximum circuit lower bound for $\mathsf{S}_2\mathsf{E}$ using an elegant and elementary proof without relying on the win-win argument. The proof is inspired by the result in [18], with an additional observation that (intuitively) we will just fall into the (*Win*) case if we use a specific brute-force algorithm via the hardness condensation procedure in [39] and define the computation history of it carefully.[19]

2.2 Warmup: A Win-win Argument

Can we directly apply the hardness condensation procedure [39] used in [20, 40] to obtain a circuit lower bound for AMEXP, rather than $\Sigma_2\mathsf{E}$ or $\mathsf{S}_2\mathsf{E}$? Korten's hardness condensation procedure runs in P^{NP} (see, e.g., [39, 47]). A natural idea would be to have a better algorithm for the hardness condensation procedure (say, a single-valued Arthur-Merlin algorithm, which would give circuit lower bounds for AMEXP). Unfortunately, it is unclear how the algorithm should be designed.[20]

Again, viewing the task of proving circuit lower bounds (e.g. Theorem 1.1) as a *derandomization* problem, i.e., pseudodeterministically hitting the dense coNP property consisting of hard truth tables, brought insights on what tools we should look for. Recall that [20] performs an (iterative) win-win argument using the uniform and instance-wise Chen-Tell HSG [21], it is natural to ask whether we should use a similar instance-wise HSG fooling (co-)nondeterministic computation.[21]

Fortunately, a recent work by van Melkebeek and Mocelin Sdroievski [53] (inspired by the Chen-Tell HSG [21]) provides a uniform and instance-wise hardness-vs-randomness connection for AM that is suitable for our application. By combining the PCP theorem (see, e.g., [5]) for nondeterministic computation and a hitting set generator [50] with Arthur-Merlin reconstruction, they proved that:

Theorem 2.1 (informal, see the full version [19])**.** *There is an efficient algorithm* HSG *and an efficient Arthur-Merlin protocol* Rec *such that the following holds. Let $n, m \in \mathbb{N}$, $T \leq 2^{\mathrm{poly}(n)}$, M be a time-T Turing machine, and α be a string. For every* $\mathrm{poly}(m)$*-size* coAM *circuit $D : \{0,1\}^n \to \{0,1\}$ that rejects at most a $1/3$-fraction of its inputs, at least one of the following two conditions holds:*

- *(Hit).* $\mathsf{HSG}(n, m, M, \alpha)$ *runs in time* $\mathrm{poly}(T)$ *and outputs a multset $H \subseteq \{0,1\}^m$ such that $D(z) = 1$ for some $z \in H$.*
- *(Reconstruction). The Arthur-Merlin protocol* $\mathsf{Rec}(n, m, M, \alpha, D, x)$ *runs in $m^{O((\log\log T)^2)} \ll T$ and works as follows: If $M(\alpha)$ halts in time T and outputs x, there is a strategy of Merlin that makes Arthur always accept; otherwise, Arthur rejects with high probability regardless of Merlin's strategy.*

Intuitively, we treat $M(\alpha)$ (the computation of M on input α) as a potential "hard computation", and show that either we can produce a hitting set fooling a coAM circuit D, or it is indeed not a "hard computation" as $M(\alpha)$ can be simulated by a fast Arthur-Merlin protocol Rec that takes D as its input.[22]

Sub-exponential time algorithm from a win-win argument. As a warmup, we first explain how to construct a non-trivial pseudodeterministic algorithm for hitting dense coAM property using Theorem 2.1 and a vanilla win-win argument.

Let BF be the brute-force algorithm that enumerates all n-bit strings $x_1, x_2, \ldots, x_{2^n}$ in lexicographic order, checks whether $x_i \in P$ in exponential time by enumerating all witnesses and outputs the lexicographically first string in the property P. We will use BF as the machine M in Theorem 2.1 to obtain a hitting set for the dense property P decidable in coAM. Let $m = m(n) = n^c$ for a sufficiently large constant c. On input length m, it plugs BF and 1^n (as M and α) into Theorem 2.1 to construct a hitting set $H \subseteq \{0,1\}^m$ in time $2^{\mathrm{poly}(n)}$. Then there will be two cases.

- *Case 1: Hitting*. Suppose that H hits the property P for infinitely many input lengths m, i.e., there is an index r such that the r-th string in H is also in P. Let r^* be the lexicographically first such r. Then the following deterministic algorithm in time $2^{\mathrm{poly}(n)}$ with $\mathrm{poly}(n)$ bits of advice r^* will hit the property P infinitely often: On input length m, it simulates $\mathsf{BF}(1^n)$, constructs the hitting set H, and outputs the r^*-th string in H. This deterministic algorithm runs in $2^{m^{0.1}}$ time and takes $m^{0.1}$ bits of advice if c is chosen to be sufficiently large.
- *Case 2: Reconstruction.* Suppose that the hitting set generator fails to hit the property P on all but finitely many input lengths m. Fix any n and $m = m(n)$. By Theorem 2.1, we can verify whether $\mathsf{BF}(1^n)$ outputs x for any $x \in \{0,1\}^n$ by the Arthur-Merlin protocol Rec, where the distinguisher D is the coAM property P. This leads to an efficient single-valued AM protocol that simulates $\mathsf{BF}(1^n)$ on input length n by letting Merlin send the correct output x of $\mathsf{BF}(1^n)$ that is the lexicographically first n-bit string in P by the definition of BF.

We can also view it in the "input-length-pair-wise" perspective: Let $n_1, n_2, \ldots$ and $m_1, m_2, \ldots$ be two disjoint sequences of input

[19] Note that both [18] and [40] indeed prove stronger results: they designed a single-valued algorithm (in the functional version of $\mathsf{S}_2\mathsf{P}$ or $\mathsf{S}_2\mathsf{P}_{/1}$) solving the range avoidance problem (see, e.g., [39, 47]), which is known to imply circuit lower bounds (for $\mathsf{S}_2\mathsf{E}$ or $\mathsf{S}_2\mathsf{E}_{/1}$).

[20] It is worth noting that the range avoidance problem [39] (see also Section 1.2) cannot be solved by non-uniform nondeterministic search algorithms under plausible cryptographic assumptions [23]. Since a single-valued Arthur-Merlin algorithm can be derandomized using non-uniformity by a standard argument (see, e.g., [5]), the range avoidance problem is also unlikely to be solvable by a single-valued Arthur-Merlin algorithm. This might hint that it is hard to improve Korten's hardness condensation procedure, which is derived from a P^{NP} algorithm assuming circuit lower bounds, to a single-valued Arthur-Merlin algorithm.

[21] Our Theorem 1.3 can indeed hit any dense coAM property, where $\mathsf{coNP} \subseteq \mathsf{coAM}$. This is crucial for proving Theorem 1.2 as the dense property for proving it will be a coAM property rather than a coNP property.

[22] There are two caveats. The reconstruction protocol Rec runs in slightly super-polynomial time due to overhead in the hitting set generator [50]. Moreover, the version of hardness-vs-randomness connection we will need is slightly different from the one in [53]; concretely, we will need HSG and Rec to work not only for a fixed time bounded $T = T(n)$, but also when T (encoded in binary) is given in their inputs.

lengths defined as $m_i \coloneqq n_i^c$, $n_{i+1} \coloneqq m_i^c$, for each $i \in \mathbb{N}$, our unified algorithm simulates the algorithm in the former case on input lengths m_i, and simulates the algorithm in the latter case on input lengths n_i. By a win-win argument on each pair (n_i, m_i) of input lengths, our unified algorithm is correct on infinitely many input lengths.

A technical challenge: hardness of deciding the property. Recall that a vanilla win-win argument (including the result above) is subject to the half-exponential barrier (see Section 2.1). Can we bypass the barrier using the iterative win-win paradigm in [20]?

A key difference between our task of hitting dense coAM properties and the task of hitting dense P properties considered in [20] is that we cannot decide the coAM property we need to hit efficiently by a deterministic algorithm (unless AM = P). Recall that in the (Improve) case of the iterative win-win argument in [20], we can construct a moderately more efficient *deterministic* algorithm constructing primes by enumerating over the hitting set, *testing* their primality, and outputting the first prime in the hitting set (see Section 2.1); this ensures that the improvement in the case (Improve) could accumulate over iterations. However, in our setting, it is unclear how to construct this "moderately more efficient algorithm" without the ability to decide the property.

In this paper, we provide two different approaches to partially resolve the issue by allowing the algorithm to take a short advice (see Theorem 1.3). The first proof follows from adapting the *iterative win-win paradigm* [20] to algorithms with short advice; see Section 2.3. The second proof follows from a novel *critical win-win argument* that bypasses the half-exponential barrier without performing a win-win argument on super-constantly many cases; see Section 2.4. As far as we can tell, these two proofs are technically incomparable and interesting as they provide two conceptually different approaches to bypass the half-exponential barrier.

2.3 Proof via Iterative Win-win with Advice

Recall that in the vanilla win-win argument (see Section 2.2), the algorithm in the (Hitting) case takes short advice and runs in subexponential time. In this subsection, we will briefly explain how to improve the running time to quasi-polynomial by adapting the iterative win-win paradigm [20] to work with algorithms that take short advice.

Let $n_0, n_1, \ldots, n_\ell$ be a sequence of input lengths. Let $\mathsf{BF}_0(1^n)$ be the brute-force algorithm that finds the lexicographically first length-n string in the property P we want to hit. Similar to the win-win argument in Section 2.2, we plug BF_0 and 1^{n_0} (as M and α) into Theorem 2.1 to construct a candidate hitting set $H_0 \subseteq \{0,1\}^{n_1}$ in time $2^{\mathrm{poly}(n_0)}$. There are two cases:

- (Win). If H_0 fails to hit the property P, i.e., H_0 is not a hitting set fooling $P \in \mathsf{coAM}$, we know by Theorem 2.1 that the reconstruction AM protocol Rec will simulate $\mathsf{BF}_0(1^{n_0})$ efficiently. (Indeed, Rec runs in quasi-polynomial time, see the full version [19] for the formal statement.)
- (Improve). Otherwise, we obtain a $2^{\mathrm{poly}(n_0)}$-time deterministic algorithm BF_1 that outputs a string in $P \cap H_0$ that takes a $\mathrm{poly}(n_0)$-bit advice α_1. If we set $n_1 \gg n_0$ (e.g., $n_1 = n_0^\beta$ for a large constant β), $\mathsf{BF}_1(1^{n_1})_{\alpha_1}$ runs moderately faster than the brute-force algorithm.

The crucial observation is that in the (Improve) case, we can keep performing the win-win argument as Theorem 2.1 is "instance-wise", i.e., it allows the machine M to take an input α rather than only 1^n. For simplicity of presentation, we assume that BF_1 takes both 1^{n_1} and α_1 as its input (rather than advice). We plug BF_1 and $(1^{n_1}, \alpha_1)$ (as M and α) into Theorem 2.1 to construct a candidate hitting set $H_1 \subseteq \{0,1\}^{n_2}$ in time $(2^{\mathrm{poly}(n_0)})^{O(1)}$, and consider the two cases:

- (Win). If H_1 fails to hit the property P, i.e., H_1 is not a hitting set fooling $P \in \mathsf{coAM}$ (on the input length n_2), we know by Theorem 2.1 that the AM protocol Rec will simulate $\mathsf{BF}_1(1^{n_1}, \alpha_1)$ efficiently, where $(1^{n_1}, \alpha_1)$ is given to the AM protocol as a part of its input. Compared to the (Win) case above, we will only obtain a single-valued AM protocol hitting P given α_1 as its advice instead of a fully uniform AM protocol; nevertheless, this suffices to prove Theorem 1.3.
- (Improve). Otherwise, we obtain a $(2^{\mathrm{poly}(n_0)})^{O(1)}$-time deterministic algorithm BF_2 that outputs a string in $P \cap H_1$ that takes two advice strings: the advice α_1 for BF_1 to compute the hitting set generator H_1, and an advice string α_2 of length $O(\mathrm{poly}(n_0))$ that identifies a string in $P \cap H_1$.

This win-win argument can be iteratively performed over super-constantly many input lengths. That is, for each $i \geq 1$:

- BF_i takes $\alpha_1, \ldots, \alpha_{i-1}, \alpha_i$ as advice, where $\alpha_1, \ldots, \alpha_{i-1}$ are used to simulate BF_{i-1},[23]
- $H_i \subseteq \{0,1\}^{n_{i+1}}$ is the hitting set obtained by plugging BF_i and $(1^{n_i}, \alpha_1, \ldots, \alpha_i)$ (as M and α) into Theorem 2.1.
- BF_i first obtains the hitting set H_{i-1} using BF_{i-1} and $(1^{n-1}, \alpha_1, \ldots, \alpha_{i-1})$, and uses α_i to identify a length-n_i string in $H_{i-1} \cap P$.

This will lead to Theorem 1.3 by carefully tracking the time and advice complexity of BF_i and setting the sequence $n_0, \ldots, n_\ell$ according.

2.4 Proof via Critical Win-Win

We now introduce an alternative approach to speed up the derandomization algorithm in Section 2.2, which we call a *critical win-win argument*. Rather than using the hardness-vs-randomness connection in Theorem 2.1 in a black-box manner, it exploits special properties of the specific hitting set generator [50] underlying Theorem 2.1, and combines this generator with a *strong, Reed-Muller-based* PCP [46].

Insight: the amount of hardness. Instead of introducing more cases in win-win arguments to amortize the cost as in the iterative win-win paradigm, the critical win-win argument is inspired by the observation that we can *reduce* the cost by considering the exact "amount of hardness" we need for derandomization. This observation has been used by Lu, Oliveira, and Santhanam [41] to improve the explicit construction algorithm in [45] and subsequently by

[23] Note that in [20], the sequence of brute-force algorithms are represented by different Turing machines, and they need to track the growth of the description lengths. We introduce a trick that allows us to represent $\mathsf{BF}_0, \ldots, \mathsf{BF}_\ell$ as a single Turing machine using the recursion theorem.

Hirahara, Ren, and Lu [31] to prove circuit lower bounds, while the idea can be dated back to the circuit lower bound for MA [48].

We explain the idea using the result of [41] as an example. Recall that in [45] (see Section 2.1) a sub-exponential time algorithm for generating canonical primes is constructed by performing a win-win argument on whether PSPACE = BPP. If PSPACE ≠ BPP, we can construct a PRG $G^{L^{\mathsf{TV}}_m}:\{0,1\}^{\mathrm{poly}(m)}\to\{0,1\}^n$ that is guaranteed to hit the set of primes by plugging in a PSPACE-complete language L^{TV} into the framework in [52]; specifically, the PRG will utilize the truth table of L^{TV} on the input length $m=n^{\varepsilon}$ for some constant $\varepsilon\in(0,1)$. It will take $2^{\mathrm{poly}(m)}$ time to produce the truth table via brute force, which leads to a sub-exponential time overhead in the final algorithm to generate a canonical prime number in [45].

The first observation in [41] is that by providing the lexicographically first seed w such that $G^{L^{\mathsf{TV}}_m}(w)$ is prime as advice, it suffices to evaluate $G^{L^{\mathsf{TV}}_m}$ on a single seed for generating a canonical prime number. Moreover, the PRG used in [52] (which is essentially the Nisan-Wigderson PRG [44]) is *local* in the sense that it allows us to output $G^{L^{\mathsf{TV}}_m}(w)$ given the seed w with $\mathrm{poly}(m)$ queries to $L^{\mathsf{TV}}_m:\{0,1\}^m\to\{0,1\}$ (rather than reading the entire truth table). Therefore, we can get rid of the $2^{\mathrm{poly}(m)}$ overhead if we can compute L^{TV}_m efficiently.

Crucially, it is observed in [41] that it benefits to consider the *exact amount of hardness* of L^{TV}_m. Intuitively, it is proved in [52] that if L^{TV}_m is hard for T^c-time probabilistic algorithms for some constant $c>1$, then $G^{L^{\mathsf{TV}}_m}:\{0,1\}^{\mathrm{poly}(m)}\to\{0,1\}^n$ fools any T-time algorithm for *every* function T. The flexibility in the hardness-vs-randomness connection allows us to consider what is the "minimum"[24] $T^*(n)$ such that $L^{\mathsf{TV}}\in\mathsf{BPPTIME}[T^*(n)]$, which characterizes "the exact amount of hardness" of L_{TV},[25] and use both the (probabilistic time) upper and lower bound for L^{TV}. Concretely, we will set m so that $T^*(m)=\mathrm{poly}(n)$ such that:

(1) The $T^*(m)$-time upper bound allows us to evaluate L^{TV} efficiently, and thus by the locality of the PRG in [52], we can output a canonical prime $G^{L^{\mathsf{TV}}_m}(w)$ with high probability given w as advice in probabilistic time $\mathrm{poly}(m)\cdot T^*(m)=\mathrm{poly}(n)$.
(2) The (roughly $T^*(m)$ time) lower bound for L^{TV} makes sure that L^{TV}_m will hit the set of primes on the input length n (as long as $T^*(m)\geq n^{\delta}$ for a sufficiently large constant δ) by the hardness-vs-randomness framework in [52].

Therefore, by considering the exact amount of hardness and using both the upper and lower bounds, [41] improved the algorithm generating a canonical prime to polynomial time.

Locality of the HSG.. Recall that the time bottleneck of the algorithm in Section 2.2 appears in the "hitting" case. Suppose that the hitting set H constructed $\mathsf{BF}(1^n)$ using Theorem 2.1 hits an m-bit string in P, we need to compute the entire hitting set H within time $2^{\mathrm{poly}(n)}$ to output a canonical element in $H\cap P$, resulting in an unaffordable exponential running time in m.

[24] More formally, we need to find $T^*(n)$ such that $L^{\mathsf{TV}}\in\mathsf{BPPTIME}[T^*(n)]\setminus\mathsf{BPPTIME}[n^b\cdot(T^*(n))^{\delta}]_{/\delta\log T^*(n)}$ in [41] for some constants b and δ. We will ignore the formality and say "minimum" $T^*(n)$ informally here.
[25] Note that $T^*(n)=n^{\omega(1)}$ as PSPACE ≠ BPP and L^{TV} is PSPACE-complete.

Based on the first observation in [41], it is natural to ask whether it is overkill to generate the entire hitting set. Fortunately, similar to the PRG used in [41], the hitting set construction [50] underlying Theorem 2.1 is *local* in the sense that it allows us to output a *single element* in H more efficiently. Opening up the proof of Theorem 2.1, we can see that H is constructed by plugging the *computation history* of $\mathsf{BF}(1^n)$ (as a certain PCP proof) into the hitting set generator in [50]. Given oracle access to the computation history of $\mathsf{BF}(1^n)$ and a seed i, we can compute the i-th string in H within time $\mathrm{poly}(m)$. As we can store the index i in the advice, the "hitting" case can be improved to $\mathrm{poly}(m)$ time if we can implement the oracle access to the computation history of $\mathsf{BF}(1^n)$ efficiently.

Critical win-win argument. The crucial idea of our critical win-win argument is to define a suitable notion of the "amount of hardness".

Instead of identifying a language as in [41], our result is based on the "input-length-pair-wise" perspective of win-win arguments (see Section 2.1). Let $H_{n,m}$ be a hitting set over m-bit strings constructed from the computation history of $\mathsf{BF}(1^n)$. We will play with three input lengths $n,m,m+1$ such that

- $H_{n,m}$ hits an m-bit string in P, and
- $H_{n,m+1}$ fails to hit any $(m+1)$-bit string in P,

We will call (n,m) a *critical pair*. To gain some intuition, one may think about the case where $H_{n,t}$ hits a t-bit string in P for every $t\leq m$, and fails to hit any t-bit string in P for every $t>m$; in this simplified setting, the critical pair (n,m) captures the exact amount of "hardness" of the computation history of $\mathsf{BF}(1^n)$ that can be used for derandomization (on the task of hitting P).

The crucial observation leading to the critical win-win argument is that for each critical pair (n,m), we can efficiently construct (with short advice) a canonical m-bit string in $H_{n,m}\cap P$ with a suitable hardness-vs-randomness framework. Intuitively, the algorithm can "reconstruct" the computation history of $\mathsf{BF}(1^n)$ from the failure of hitting P using $H_{n,m+1}$, and output a string in $H_{n,m}\cap P$ according to an index encoded in the advice using the *locality* of the hitting set generator.[26] Formally, we will design a single-valued Arthur-Merlin algorithm $\mathrm{Critical}_{\alpha}$ such that for each critical pair (n,m), $\mathrm{Critical}(1^n,1^m)_{\alpha_m}$ constructs a canonical m-bit string in $H_{n,m}\cap P$, runs in time $2^{\mathrm{polylog}(m)}$, and takes a short advice string α_m.[27]

Assume that such an algorithm Critical exists. We consider the following three cases:

- *Case 1: Easy Reconstruction.* If there are infinite many pairs (n,m) such that $m\leq 2^{\log^k n}$ and $H_{n,m}$ fails to hit P, we can instantiate the original algorithm for "the reconstruction case" in Section 2.2. Namely, we directly run the reconstruction protocol in Theorem 2.1 to simulate $\mathsf{BF}(1^n)$ and output the lexicographically first n-bit string in P.

[26] Here, the specific mean of "reconstruction" will be explained below when we formally describe the theorem.
[27] In the previous algorithm in Section 2.2 and the iterative win-win framework, the parameter m should be bounded by $2^{\mathrm{polylog}(n)}$ to make sure the protocol in the "reconstruction" case is efficient enough (with running time $2^{\mathrm{polylog}(m)}=2^{\mathrm{polylog}(n)}$) for input length n. This is no longer required as we will run the reconstruction protocol on input length m instead of n in the new algorithm, which is the key to avoiding iterative win-win.

- *Case 2: Easy Hitting.* If for some large constant β, there are infinitely many pairs (n, m) such that $m \geq 2^{n^\beta}$ and $H_{n,m}$ hits m, we can instantiate the original algorithm for "the hitting case" in Section 2.2. It naively simulates $\mathsf{BF}(1^n)$, computes the entire hitting set $H_{n,m}$, and outputs the lexicographically first string in P (according to a short advice string). The simulation of $\mathsf{BF}(1^n)$ requires $2^{n^{O(1)}} = 2^{\mathrm{polylog}(m)}$ time.
- *Case 3: Critical Hitting.* Otherwise, for all but finite many n, $H_{n,m}$ hits P for all $m \leq 2^{\log^k n}$, and $H_{n,m}$ fails to hit P for some $m \leq 2^{n^\beta}$. Therefore for any sufficiently large n, there is an $m \in [2^{\log^k n}, 2^{n^\beta}]$ such that $H_{n,m}$ hits P and $H_{n,m+1}$ fails to hit P, or equivalently, (n, m) forms a critical pair. This means that for infinitely many m, there is an n such that
  – $2^{\log^k n} \leq m \leq 2^{n^\beta}$,
  – (n, m) is a critical pair.
  On each such input length m, if we take as advice the corresponding input length n and the advice α_m for Critical on the critical pair (n, m), we can simulate $\mathrm{Critical}(1^n, 1^m)_{\alpha_m}$ and output a canonical m-bit string in $H_{n,m} \cap P$ in time $2^{\mathrm{polylog}(m+1)} = 2^{\mathrm{polylog}\, m}$.

A closer look at Theorem 2.1: How can locality help? To explain how this algorithm σ_{crit} works, we need to take a closer look at how the uniform hardness-vs-randomness connection for AM [53] (also see Theorem 2.1) is proved. The key technical ingredient is the hitting set generator in [50]. Intuitively, it works as follows: Let p be an m-variate low-degree polynomial over $\mathbb{F} \coloneqq \mathbb{F}_q$, it either generates a valid hitting set fooling coAM based on p, or (given oracle access to an coAM circuit that it fails to fool) *reconstructs* p. Here, the reconstruction of p is achieved by an AM *commit-and-evaluate* protocol which consists of two AM protocols σ_{c} and σ_{e} that informally works as follows:

- In σ_{c}, Merlin will commit to a low-degree polynomial $g_\alpha : \mathbb{F}^m \to \mathbb{F}$, and Arthur will output a string α that is supposed to be a commitment made by Merlin;
- In σ_{e}, Arthur (given the commitment α and an $x \in \mathbb{F}^m$) will ask Merlin to send some $y \in \mathbb{F}$, which is supposed to the evaluation $g_\alpha(x)$ of the committed polynomial.

It is ensured that if the HSG constructed from p fails, then: There is a strategy of Merlin that commits to p in σ_{c} and makes Arthur output $y = p(x)$ in σ_{e}, and for any strategy of Merlin, once it commits, the evaluation protocol will be single-valued. That is, once Merlin commits a polynomial g_α, it will have to faithfully reveal $y = g_\alpha(x)$ in $\sigma_{\mathrm{e}}(\alpha, x)$ since attempting to reveal any value other than $g_\alpha(x)$ will be rejected by Arthur with high probability.[28]

The commit-and-evaluate reconstruction protocol makes it possible to verify any $\mathsf{NTIME}[T]$ language L by an AM protocol in time $\mathrm{polylog}(T)$: We plug the (low-degree extension of) the PCP proof for an inefficient deterministic computation as the polynomial p into the HSG so that if the HSG fails, we can achieve random access to the PCP proof by a commit-and-evaluate protocol, which (together with the PCP verifier in time $\mathrm{polylog}(T)$) implies fast simulation $\mathsf{NTIME}[T]$ by an Arthur-Merlin protocol. More concretely, Arthur will ask Merlin to commit to a polynomial encoding a PCP proof, and then simulate the PCP verifier (where queries to the proof are implemented by the evaluation protocol σ_{e}).[29] In particular, we can prove Theorem 2.1 by letting L be the verification of the deterministic computation "$M(\alpha) = x$".

How can we make use of the locality property of the HSG in [50] to design the algorithm σ_{crit}? Suppose that (n, m) is a critical pair, i.e., $H_{n,m}$ hits P but $H_{n,m+1}$ fails to hit P, our goal is to construct a canonical string in $H_{n,m} \cap P$. A natural idea is to mimic the proof of Theorem 2.1. Recall that $H_{n,m}$ is defined as the HSG in [50] where p is the computation history of $\mathsf{BF}(1^n)$ in the form of a (Reed-Muller-encoded) PCP proof. Since $H_{n,m+1}$ fails to hit P, the reconstruction protocol provides oracle accesses to a polynomial that is supposed to be the (Reed-Muller-encoded) computation history of $\mathsf{BF}(1^n)$. By running the PCP verifier, Arthur can make sure that Merlin commits to a correct computation history, and thus σ_{e} provides efficient oracle access to the committed computation history. By the locality of the HSG, we can then output the i-th string in $H_{n,m}$ given i efficiently. This implies that we can output a canonical string in $H_{n,m} \cap P$ if we take an advice string i such that the i-th string in $H_{n,m}$ is in P.

A flaw, and the solution using certain PCP systems. However, there is a subtle flaw in the argument. Although Merlin cannot change the polynomial once it is committed, it does not prevent Merlin from committing to another polynomial other than the polynomial p used to generate the HSG. Therefore, if there are multiple valid computation histories (in the form of PCP proofs), Arthur will accept when Merlin commits to a polynomial encoding any of the computation histories, which makes the protocol *not* single-valued. We stress that even though in Theorem 2.1 we only want to simulate *deterministic* Turing machines in AM whose computation pattern is unique, there could still be multiple valid PCP proofs for verifying the computation, which may make the protocol being not single-valued.

To resolve this issue, we need to look for a suitable definition of the computation history (in the form of a PCP proof) such that it is in some sense *unique*; that is, if the prover deviates from a *canonical PCP proof*, the verifier will reject with noticeable probability. If this is possible, Merlin can only commit to the canonical PCP proof (for verifying the computation of $\mathsf{BF}(1^n)$) while running the reconstruction protocol (utilizing the failure of hitting $H_{n,m}$), and thus the final protocol will be single-valued by the correctness of the commit-and-evaluate protocol.

Fortunately, it turns out that a strong and canonical PCP with Reed-Muller-encoded proofs suffices, and it can be constructed from the standard algebraic proof of the PCP theorem [6–8, 12, 46] with certain technical modifications. A PCP system for an NP relation R is said to be strong and canonical if for each input x and each witness w such that $R(x, w)$ is true, there is a canonical PCP proof $\Pi = \Pi(x, w)$ satisfying that

(1) the verifier accepts with probability 1 if the proof is Π, and
(2) for some constant $\alpha \in (0, 1)$, the verifier rejects with probability at least $\alpha \cdot \delta$ if the proof is δ-far from Π.

[28] For readers familiar with cryptography, this is functionally similar to a commitment scheme with local opening (e.g. the commitment scheme in Kilian's protocol [37]).

[29] Again, this is very similar to the construction of Kilian's *succinct argument* scheme [37].

Compared to a standard PCP system, a strong and canonical PCP ensures that if a proof oracle is accepted, it is likely to be close to the unique canonical proof oracle.

We can now sketch why this will resolve the aforementioned issue. Since the proof oracle of the PCP verifier is a Reed-Muller code word[30], if we define the computation history as the *canonical* proof oracle and plug it (as the polynomial p) into the HSG in [50], the reconstruction protocol (in case that the HSG fails) ensures that if Merlin commits to a polynomial g_α different from p (i.e. the polynomial encoding the *canonical* PCP proof), either g is noticeably far from any low-degree polynomial or (by Schwartz-Zippel lemma) g is very far from p. In the former case, Arthur can detect it using standard low-degree testing; in the latter case, the PCP verifier will reject it with noticeable probability. Therefore, by performing these two tests, Arthur can force Merlin to commit to the desired polynomial p, and thus make the protocol single-valued.[31]

Additional technical issues. It is worth noting that to implement the idea above, we need to ensure that the (Reed-Muller encoded) proof oracle of the PCP system and the low-degree polynomial p in hitting set generator [50] are using the same parameters (i.e. field size, degree, and the number of variables). This requires a careful inspection of both proofs and several changes to them; in particular, we need to work with a Reed-Muller-based PCP with fields of size that is *exponentially larger* than that of the usual setting. We provide a formal description of our PCP system and a self-contained proof in the full version [19] (with an overview of all changes we made), as well as an exposition of the hitting set generator [50] highlighting the properties to be checked and the changes to be made.

2.5 Open Problems

An immediate open question stemming from our work is whether we can reduce the length of the non-uniform advice from Theorem 1.1, or even remove it, to obtain a uniform exponential circuit lower bound for AMEXP. Note that [14] showed that MAE requires half-exponential-size circuits, but it is unclear how to adapt our techniques to improve this lower bound.

Another open problem is whether we can obtain exponential lower bounds for $\mathsf{AME} = \mathsf{AMTIME}[2^{O(n)}]$ instead of AMEXP (here we allow the sub-exponential amount of advice). The main technical issue that prevents us from proving such a lower bound is that the reconstruction procedure for the hitting set generator in [50] has a quasi-polynomial overhead from collapsing a logarithmic-round protocol to a constant-round protocol [9]; see Section 2 for details.

Finally, it is interesting to understand the strength of our techniques: iterative win-win argument and critical win-win argument. Can we adapt these techniques to prove new results in complexity theory? Is there a barrier (e.g. relativization [10] or algebrization [1]) that prevents us from proving (say) $\mathsf{EXP}_{/2^{n^\varepsilon}} \not\subset \mathsf{SIZE}[2^n/n]$ or $\mathsf{AMEXP} \not\subset \mathsf{SIZE}^{\mathsf{AM}\cap\mathsf{coAM}}[2^n/n]$ using these techniques? Note that our proof does not relativize due to the usage of the PCP theorem, but it may algebrize under a suitable definition.

[30] Indeed, the PCP proof in [11, 46] is not a single Reed-Muller code word but the concatenation of several Reed-Muller code words. This is a minor technical issue and we refer readers to the full version [19] for more details.

[31] One may suspect that a strong and canonical PCP (i.e. not necessarily Reed-Muller encoded) should suffice, as we can either define the PCP system as the composition of it and the low-degree extension, or view the hitting set generator as the composition of it and the low-degree extension. Unfortunately, neither of the approaches works: the composition of a strong and canonical PCP and the low-degree extension may not necessarily be strong and canonical, and the composition of a local hitting set generator and the low-degree extension may not necessarily be local.

Acknowledgments

Lijie Chen is supported by a Miller Research Fellowship. Jiatu Li is supported by the MIT Akamai Presidential Fellowship and the National Science Foundation under Grant CCF-2127597. We thank Hanlin Ren and Ryan Williams for helpful discussion, and an anonymous STOC referee for pointing out that there are multiple formal definitions of AM ∩ coAM classes with advice.

References

[1] Scott Aaronson and Avi Wigderson. 2009. Algebrization: A new barrier in complexity theory. *ACM Transactions on Computation Theory* 1, 1 (Feb. 2009), 2:1–2:54. doi:10.1145/1490270.1490272

[2] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. 2004. PRIMES Is in P. *Annals of Mathematics* 160, 2 (2004), 781–793. jstor:3597229

[3] M. Ajtai. 1983. Σ^1_1-Formulae on finite structures. *Annals of Pure and Applied Logic* 24, 1 (July 1983), 1–48. doi:10.1016/0168-0072(83)90038-6

[4] Josh Alman and Lijie Chen. 2022. Efficient Construction of Rigid Matrices Using an NP Oracle. *SIAM J. Comput.* (March 2022). doi:10.1137/20M1322297

[5] Sanjeev Arora and Boaz Barak. 2009. *Computational complexity: a modern approach*. Cambridge University Press.

[6] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. 1998. Proof verification and the hardness of approximation problems. *J. ACM* 45, 3 (May 1998), 501–555. doi:10.1145/278298.278306

[7] Sanjeev Arora and Shmuel Safra. 1998. Probabilistic checking of proofs: A new characterization of NP. *J. ACM* 45, 1 (Jan. 1998), 70–122. doi:10.1145/273865.273901

[8] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. 1991. Checking computations in polylogarithmic time. In *Proc. 23rd ACM Symposium on Theory of Computing (STOC)*. 21–32. doi:10.1145/103418.103428

[9] László Babai and Shlomo Moran. 1988. Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity class. *J. Comput. System Sci.* 36, 2 (April 1988), 254–276. doi:10.1016/0022-0000(88)90028-1

[10] Theodore Baker, John Gill, and Robert Solovay. 1975. Relativizations of the P =?NP Question. *SIAM J. Comput.* 4, 4 (Dec. 1975), 431–442. doi:10.1137/0204037

[11] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. 2006. Robust PCPs of Proximity, Shorter PCPs, and Applications to Coding. *SIAM J. Comput.* 36, 4 (Jan. 2006), 889–974. doi:10.1137/S0097539705446810

[12] Eli Ben-Sasson and Madhu Sudan. 2005. Simple PCPs with poly-log rate and query complexity. In *Proc. 37th ACM Symposium on Theory of Computing (STOC)*. 266–275. doi:10.1145/1060590.1060631

[13] Amey Bhangale, Prahladh Harsha, Orr Paradise, and Avishay Tal. 2024. Rigid Matrices from Rectangular PCPs. *SIAM J. Comput.* 53, 2 (April 2024), 480–523. doi:10.1137/22M1495597

[14] H. Buhrman, L. Fortnow, and T. Thierauf. 1998. Nonrelativizing separations. In *Proc. 13th Computational Complexity Conference (CCC)*. 8–12. doi:10.1109/CCC.1998.694585

[15] Joshua Buresh-Oppenheim and Rahul Santhanam. 2006. Making Hard Problems Harder. In *Proc. 21st Computational Complexity Conference (CCC)*. 73–87. doi:10.1109/CCC.2006.26

[16] Jin-Yi Cai. 2007. $S_2^p \subseteq \mathrm{ZPP}^{\mathrm{NP}}$. *J. Comput. System Sci.* 73, 1 (Feb. 2007), 25–35. doi:10.1016/j.jcss.2003.07.015

[17] Lijie Chen. 2024. Nondeterministic Quasi-Polynomial Time is Average-Case Hard for ACC Circuits. *SIAM J. Comput.* (Feb. 2024), 19–332. doi:10.1137/20M1321231

[18] Lijie Chen, Shuichi Hirahara, and Hanlin Ren. 2024. Symmetric Exponential Time Requires Near-Maximum Circuit Size. In *Proc. 56th ACM Symposium on Theory of Computing (STOC)*. 1990–1999. doi:10.1145/3618260.3649624

[19] Lijie Chen, Jiatu Li, and Jingxun Liang. 2024. Maximum Circuit Lower Bounds for Exponential-time Arthur Merlin. electronic colloquium on computational complexity:TR24-182

[20] Lijie Chen, Zhenjian Lu, Igor C. Oliveira, Hanlin Ren, and Rahul Santhanam. 2023. Polynomial-Time Pseudodeterministic Construction of Primes. In *Proc. 64th IEEE Symposium on Foundations of Computer Science (FOCS)*. 1261–1270. doi:10.1109/FOCS57990.2023.00074

[21] Lijie Chen and Roei Tell. 2022. Hardness vs Randomness, Revised: Uniform, Non-Black-Box, and Instance-Wise. In *Proc. 63rd IEEE Symposium on Foundations of Computer Science (FOCS)*. 125–136. doi:10.1109/FOCS52979.2021.00021

[22] Yeyuan Chen, Yizhi Huang, Jiatu Li, and Hanlin Ren. 2023. Range Avoidance, Remote Point, and Hard Partial Truth Table via Satisfying-Pairs Algorithms. In

Proc. 55th ACM Symposium on Theory of Computing (STOC). 1058–1066. doi:10.1145/3564246.3585147

[23] Yilei Chen and Jiatu Li. 2024. Hardness of Range Avoidance and Remote Point for Restricted Circuits via Cryptography. In *Proc. 56th ACM Symposium on Theory of Computing (STOC)*. 620–629. doi:10.1145/3618260.3649602

[24] Merrick Furst, James B. Saxe, and Michael Sipser. 1984. Parity, circuits, and the polynomial-time hierarchy. *Mathematical systems theory* 17, 1 (Dec. 1984), 13–27. doi:10.1007/BF01744431

[25] Karthik Gajulapalli, Alexander Golovnev, Satyajeet Nagargoje, and Sidhant Saraogi. 2023. Range Avoidance for Constant Depth Circuits: Hardness and Algorithms. In *Proc. 26th International Conference on Approximation Algorithms for Combinatorial Optimization Problems and 27th International Conference on Randomization and Computation (APPROX/RANDOM)*. doi:10.4230/LIPIcs.APPROX/RANDOM.2023.65

[26] Eran Gat and Shafi Goldwasser. 2011. Probabilistic Search Algorithms with Unique Answers and Their Cryptographic Applications. Preprint ECCC:TR11-136. electronic colloquium on computational complexity:TR11-136

[27] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. 2015. Delegating Computation: Interactive Proofs for Muggles. *J. ACM* 62, 4 (Sept. 2015), 27:1–27:64. doi:10.1145/2699436

[28] Shafi Goldwasser and Michael Sipser. 1989. Private Coins versus Public Coins in Interactive Proof Systems. *Advances in Computing Research* 5 (1989), 73–90.

[29] Venkatesan Guruswami, Xin Lyu, and Xiuhan Wang. 2022. Range Avoidance for Low-Depth Circuits and Connections to Pseudorandomness. In *Proc. 25th International Conference on Approximation Algorithms for Combinatorial Optimization Problems and 26th International Conference on Randomization and Computation (APPROX/RANDOM)*. doi:10.4230/LIPIcs.APPROX/RANDOM.2022.20

[30] J Hastad. 1986. Almost optimal lower bounds for small depth circuits. In *Proc. 18th ACM Symposium on Theory of Computing (STOC)*. 6–20. doi:10.1145/12130.12132

[31] Shuichi Hirahara, Zhenjian Lu, and Hanlin Ren. 2023. Bounded Relativization. In *Proc. 38th Computational Complexity Conference (CCC)*. 1–45. doi:10.4230/LIPIcs.CCC.2023.6

[32] Rahul Ilango, Jiatu Li, and R. Ryan Williams. 2023. Indistinguishability Obfuscation, Range Avoidance, and Bounded Arithmetic. In *Proc. 55th ACM Symposium on Theory of Computing (STOC)*. 1076–1089. doi:10.1145/3564246.3585187

[33] Russell Impagliazzo and Avi Wigderson. 1997. $P = BPP$ if E Requires Exponential Circuits: Derandomizing the XOR Lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*. ACM, 220–229.

[34] Russell Impagliazzo and Avi Wigderson. 2001. Randomness vs Time: Derandomization under a Uniform Assumption. *J. Comput. System Sci.* 63, 4 (Dec. 2001), 672–688. doi:10.1006/jcss.2001.1780

[35] Ravi Kannan. 1982. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control* 55, 1 (Oct. 1982), 40–56. doi:10.1016/S0019-9958(82)90382-5

[36] Richard M. Karp and Richard J. Lipton. 1980. Some connections between nonuniform and uniform complexity classes. In *Proc. 12th ACM Symposium on Theory of Computing (STOC)*. 302–309. doi:10.1145/800141.804678

[37] Joe Kilian. 1992. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proc. 24th ACM Symposium on Theory of Computing (STOC)*. 723–732. doi:10.1145/129712.129782

[38] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos Papadimitriou. 2021. Total Functions in the Polynomial Hierarchy. In *Proc. 12th Innovations in Theoretical Computer Science (ITCS)*. doi:10.4230/LIPIcs.ITCS.2021.44

[39] Oliver Korten. 2022. The Hardest Explicit Construction. In *Proc. 63rd IEEE Symposium on Foundations of Computer Science (FOCS)*. 433–444. doi:10.1109/FOCS52979.2021.00051

[40] Zeyong Li. 2024. Symmetric Exponential Time Requires Near-Maximum Circuit Size: Simplified, Truly Uniform. In *Proc. 56th ACM Symposium on Theory of Computing (STOC)*. 2000–2007. doi:10.1145/3618260.3649615

[41] Zhenjian Lu, Igor C. Oliveira, and Rahul Santhanam. 2021. Pseudodeterministic algorithms and the structure of probabilistic time. In *Proc. 53rd ACM SIGACT Symposium on Theory of Computing (STOC)*. 303–316. doi:10.1145/3406325.3451085

[42] Peter Bro Miltersen, N. V. Vinodchandran, and Osamu Watanabe. 1999. Superpolynomial versus half-exponential circuit size in the exponential hierarchy. In *Proc. 5th International Conference on Computing and Combinatorics (COCOON)*. 210–220.

[43] Cody D. Murray and R. Ryan Williams. 2020. Circuit Lower Bounds for Nondeterministic Quasi-polytime from a New Easy Witness Lemma. *SIAM J. Comput.* 49, 5 (Jan. 2020), STOC18–300. doi:10.1137/18M1195887

[44] Noam Nisan and Avi Wigderson. 1994. Hardness vs randomness. *J. Comput. System Sci.* 49, 2 (Oct. 1994), 149–167. doi:10.1016/S0022-0000(05)80043-1

[45] Igor C. Oliveira and Rahul Santhanam. 2017. Pseudodeterministic constructions in subexponential time. In *Proc. 49th ACM SIGACT Symposium on Theory of Computing (STOC)*. 665–677. doi:10.1145/3055399.3055500

[46] Orr Paradise. 2021. Smooth and Strong PCPs. *Computational Complexity* 30, 1 (Jan. 2021), 1. doi:10.1007/s00037-020-00199-3

[47] Hanlin Ren, Rahul Santhanam, and Zhikun Wang. 2022. On the Range Avoidance Problem for Circuits. In *Proc. 63rd IEEE Symposium on Foundations of Computer Science (FOCS)*. 640–650. doi:10.1109/FOCS54457.2022.00067

[48] Rahul Santhanam. 2009. Circuit Lower Bounds for Merlin–Arthur Classes. *SIAM J. Comput.* 39, 3 (Jan. 2009), 1038–1061. doi:10.1137/070702680

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

[50] Ronen Shaltiel and Christopher Umans. 2007. Low-end uniform hardness vs. randomness tradeoffs for AM. In *Proc. 39th ACM Symposium on Theory of Computing (STOC)*. 430–439. doi:10.1145/1250790.1250854

[51] Claude. E. Shannon. 1949. The synthesis of two-terminal switching circuits. *The Bell System Technical Journal* 28, 1 (Jan. 1949), 59–98. doi:10.1002/j.1538-7305.1949.tb03624.x

[52] Luca Trevisan and Salil Vadhan. 2007. Pseudorandomness and Average-Case Complexity Via Uniform Reductions. *Computational Complexity* 16, 4 (Dec. 2007), 331–364. doi:10.1007/s00037-007-0233-x

[53] Dieter van Melkebeek and Nicollas Mocelin Sdroievski. 2023. Instance-Wise Hardness versus Randomness Tradeoffs for Arthur-Merlin Protocols. In *Proc. 38th Computational Complexity Conference (CCC)*. 1–36. doi:10.4230/LIPIcs.CCC.2023.17

[54] Andrew Chi-Chih Yao. 1985. Separating the polynomial-time hierarchy by oracles. In *Proc. 26th IEEE Symposium on Foundations of Computer Science (FOCS)*. 1–10. doi:10.1109/SFCS.1985.49

Received 2024-11-04; accepted 2025-02-01