Symmetric Exponential Time Requires Near-Maximum Circuit Size

Lijie Chen
University of California
Berkeley, USA
lijiechen@berkeley.edu

Shuichi Hirahara
National Institute of Informatics
Tokyo, Japan
s_hirahara@nii.ac.jp

Hanlin Ren
University of Oxford
Oxford, United Kingdom
hanlin.ren@cs.ox.ac.uk

ABSTRACT

We show that there is a language in $\mathsf{S_2E}/_1$ (symmetric exponential time with one bit of advice) with circuit complexity at least $2^n/n$. In particular, the above also implies the same near-maximum circuit lower bounds for the classes $\Sigma_2\mathsf{E}$, $(\Sigma_2\mathsf{E} \cap \Pi_2\mathsf{E})/_1$, and $\mathsf{ZPE}^{\mathsf{NP}}/_1$. Previously, only "half-exponential" circuit lower bounds for these complexity classes were known, and the smallest complexity class known to require exponential circuit complexity was $\Delta_3\mathsf{E} = \mathsf{E}^{\Sigma_2\mathsf{P}}$ (Miltersen, Vinodchandran, and Watanabe COCOON'99).

Our circuit lower bounds are corollaries of an unconditional zero-error pseudodeterministic algorithm with an NP oracle and one bit of advice ($\mathsf{FZPP}^{\mathsf{NP}}/_1$) that solves the range avoidance problem infinitely often. This algorithm also implies unconditional infinitely-often pseudodeterministic $\mathsf{FZPP}^{\mathsf{NP}}/_1$ constructions for Ramsey graphs, rigid matrices, two-source extractors, linear codes, and $\mathsf{K}^{\mathsf{poly}}$-random strings with nearly optimal parameters.

Our proofs relativize. The two main technical ingredients are (1) Korten's P^{NP} reduction from the range avoidance problem to constructing hard truth tables (FOCS'21), which was in turn inspired by a result of Jeřábek on provability in Bounded Arithmetic (Ann. Pure Appl. Log. 2004); and (2) the recent iterative win-win paradigm of Chen, Lu, Oliveira, Ren, and Santhanam (FOCS'23).

CCS CONCEPTS

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

KEYWORDS

circuit lower bounds, explicit constructions, range avoidance

ACM Reference Format:
Lijie Chen, Shuichi Hirahara, and Hanlin Ren. 2024. Symmetric Exponential Time Requires Near-Maximum Circuit Size. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC '24), June 24–28, 2024, Vancouver, BC, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3618260.3649624

1 INTRODUCTION

Proving lower bounds against non-uniform computation (i.e., circuit lower bounds) is one of the most important challenges in theoretical computer science. From Shannon's counting argument [18, 48], we know that almost all n-bit Boolean functions have *near-maximum* $(2^n/n)$ circuit complexity.[1] Therefore, the task of proving circuit lower bounds is simply to *pinpoint* one such hard function. More formally, one fundamental question is:

> What is the smallest complexity class that contains a language of exponential ($2^{\Omega(n)}$) circuit complexity?

Compared with super-polynomial lower bounds, exponential lower bounds are interesting in their own right for the following reasons. First, an exponential lower bound would make Shannon's argument *fully constructive.* Second, exponential lower bounds have more applications than super-polynomial lower bounds: For example, if one can show that E has no $2^{o(n)}$-size circuits, then we would have $\mathsf{prP} = \mathsf{prBPP}$ [28, 43], while super-polynomial lower bounds such as $\mathsf{EXP} \not\subset \mathsf{P}/_{\mathsf{poly}}$ only imply sub-exponential time derandomization of prBPP.[2]

Unfortunately, despite its importance, our knowledge about exponential lower bounds is quite limited. Kannan [31] showed that there is a function in $\Sigma_3\mathsf{E} \cap \Pi_3\mathsf{E}$ that requires maximum circuit complexity; 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. This is **essentially all we know** regarding exponential circuit lower bounds.[3]

We remark that Kannan [31, Theorem 4] claimed that $\Sigma_2\mathsf{E} \cap \Pi_2\mathsf{E}$ requires exponential circuit complexity, but [42] pointed out a gap in Kannan's proof, and suggested that exponential lower bounds for $\Sigma_2\mathsf{E} \cap \Pi_2\mathsf{E}$ were "reopened and considered an open problem." Recently, Vyas and Williams [51] emphasized our lack of knowledge regarding the circuit complexity of $\Sigma_2\mathsf{EXP}$, even with respect to *relativizing* proof techniques. In particular, the following question has been open for at least 20 years (indeed, if we count from [31], it would be at least 40 years):

Open Problem 1.1. *Can we prove that $\Sigma_2\mathsf{EXP} \not\subset \mathsf{SIZE}[2^{\varepsilon n}]$ for some absolute constant $\varepsilon > 0$, or at least show a relativization barrier for proving such a lower bound?*

[1] All n-input Boolean functions can be computed by a circuit of size $(1 + \frac{3\log n}{n} + O(\frac{1}{n}))2^n/n$ [18, 41], while most Boolean functions require circuits of size $(1+\frac{\log n}{n} - O(\frac{1}{n}))2^n/n$ [18]. Hence, in this paper, we say an n-bit Boolean function has *near-maximum* circuit complexity if its circuit complexity is at least $2^n/n$.

[2] $\mathsf{E} = \mathsf{DTIME}[2^{O(n)}]$ denotes *single-exponential* time and $\mathsf{EXP} = \mathsf{DTIME}[2^{n^{O(1)}}]$ denotes *exponential* time; classes such as E^{NP} and $\mathsf{EXP}^{\mathsf{NP}}$ are defined analogously. Exponential time and single-exponential time are basically interchangeable in the context of super-polynomial lower bounds (by a padding argument); the exponential lower bounds proven in this paper will be stated for single-exponential time classes since this makes our results stronger. Below, $\Sigma_3\mathsf{E}$ and $\Pi_3\mathsf{E}$ denote the exponential-time versions of $\Sigma_3\mathsf{P} = \mathsf{NP}^{\mathsf{NP}^{\mathsf{NP}}}$ and $\Pi_3\mathsf{P} = \mathsf{coNP}^{\mathsf{NP}^{\mathsf{NP}}}$, respectively.

[3] We also mention that Hirahara, Lu, and Ren [25] recently proved that for every constant $\varepsilon > 0$, $\mathsf{BPE}^{\mathsf{MCSP}}/_{2^{\varepsilon n}}$ requires near-maximum circuit complexity, where MCSP is the Minimum Circuit Size Problem [30]. However, the hard function they constructed requires subexponentially ($2^{\varepsilon n}$) many advice bits to describe.

STOC '24, June 24–28, 2024, Vancouver, BC, Canada

ACM ISBN 979-8-4007-0383-6/24/06
https://doi.org/10.1145/3618260.3649624

The half-exponential barrier. There is a richer literature regarding super-polynomial lower bounds than exponential lower bounds. Kannan [31] proved that $\Sigma_2\mathsf{E} \cap \Pi_2\mathsf{E}$ does not have polynomial-size circuits. Subsequent works proved super-polynomial circuit lower bounds for exponential-time complexity classes such as $\mathsf{ZPEXP}^{\mathsf{NP}}$ [5, 35], $\mathsf{S}_2\mathsf{EXP}$ [8, 9], PEXP [1, 50], and $\mathsf{MA\text{-}EXP}$ [6, 46].

Unfortunately, all these works fail to prove exponential lower bounds. All of their proofs go through certain *Karp–Lipton* collapses [32]; such a proof strategy runs into a so-called "half-exponential barrier", preventing us from getting exponential lower bounds. See subsection 5.1 for a detailed discussion.

2 OUR RESULTS

2.1 New Near-Maximum Circuit Lower Bounds

In this work, we *overcome* the half-exponential barrier mentioned above and resolve Theorem 1.1 by showing that both $\Sigma_2\mathsf{E}$ and $(\Sigma_2\mathsf{E} \cap \Pi_2\mathsf{E})/_1$ require near-maximum $(2^n/n)$ circuit complexity. Moreover, our proof indeed *relativizes*:

Theorem 2.1.

$$\Sigma_2\mathsf{E} \not\subset \mathsf{SIZE}[2^n/n] \text{ and } (\Sigma_2\mathsf{E} \cap \Pi_2\mathsf{E})/_1 \not\subset \mathsf{SIZE}[2^n/n].$$

Moreover, they hold in every relativized world.

Up to one bit of advice, we finally provide a proof of Kannan's original claim in [31, Theorem 4]. Moreover, with some more work, we extend our lower bounds to the smaller complexity class $\mathsf{S}_2\mathsf{E}/_1$, again with a relativizing proof:

Theorem 2.2.

$$\mathsf{S}_2\mathsf{E}/_1 \not\subset \mathsf{SIZE}[2^n/n].$$

Moreover, this holds in every relativized world.

The symmetric time class $\mathsf{S}_2\mathsf{E}$. $\mathsf{S}_2\mathsf{E}$ can be seen as a "randomized" version of E^{NP} since it is sandwiched between E^{NP} and $\mathsf{ZPE}^{\mathsf{NP}}$: it is easy to show that $\mathsf{E}^{\mathsf{NP}} \subseteq \mathsf{S}_2\mathsf{E}$ [45], and it is also known that $\mathsf{S}_2\mathsf{E} \subseteq \mathsf{ZPE}^{\mathsf{NP}}$ [8]. We also note that under plausible derandomization assumptions (e.g., E^{NP} requires $2^{\Omega(n)}$-size SAT-oracle circuits), all three classes simply collapse to E^{NP} [34].

Hence, our results also imply a near-maximum circuit lower bound for the class $\mathsf{ZPE}^{\mathsf{NP}}/_1 \subseteq (\Sigma_2\mathsf{E}\cap\Pi_2\mathsf{E})/_1$. This vastly improves the previous lower bound for $\Delta_3\mathsf{E} = \mathsf{E}^{\Sigma_2\mathsf{P}}$.

Corollary 2.3.

$$\mathsf{ZPE}^{\mathsf{NP}}/_1 \not\subset \mathsf{SIZE}[2^n/n].$$

Moreover, this holds in every relativized world.

2.2 New Algorithms for the Range Avoidance Problem

Background on Avoid. Actually, our circuit lower bounds are implied by our new algorithms for solving the range avoidance problem (Avoid) [33, 36, 44], which is defined as follows: given a circuit $C\colon \{0,1\}^n \to \{0,1\}^{n+1}$ as input, find a string outside the range of C (we define $\mathrm{Range}(C) := \{C(z) : z \in \{0,1\}^n\}$). That is, output any string $y \in \{0,1\}^{n+1}$ such that for every $x \in \{0,1\}^n$, $C(x) \neq y$.

There is a trivial $\mathsf{FZPP}^{\mathsf{NP}}$ algorithm solving Avoid: randomly generate strings $y \in \{0,1\}^{n+1}$ and output the first y that is outside the range of C (note that we need an NP oracle to verify if $y \notin \mathrm{Range}(C)$). The class APEPP (Abundant Polynomial Empty Pigeonhole Principle) [33] is the class of total search problems reducible to Avoid.

As demonstrated by Korten [36, Section 3], APEPP captures the complexity of explicit construction problems whose solutions are guaranteed to exist by the probabilistic method (more precisely, the dual weak pigeonhole principle [29, 37]), in the sense that constructing such objects reduces to the range avoidance problem. This includes many important objects in mathematics and theoretical computer science, including Ramsey graphs [16], rigid matrices [19, 22, 49], two-source extractors [11, 38], linear codes [22], hard truth tables [36], and strings with maximum time-bounded Kolmogorov complexity (i.e., $\mathrm{K}^{\mathsf{poly}}$-random strings) [44]. Hence, derandomizing the trivial $\mathsf{FZPP}^{\mathsf{NP}}$ algorithm for Avoid would imply explicit constructions for all these important objects.

Our results: new pseudodeterministic algorithms for Avoid. We show that, *unconditionally*, the trivial $\mathsf{FZPP}^{\mathsf{NP}}$ algorithm for Avoid can be made *pseudodeterministic* on infinitely many input lengths. A *pseudodeterministic* algorithm [20] is a randomized algorithm that outputs the same *canonical* answer on most computational paths. In particular, we have:

Theorem 2.4. *For every constant $d \geq 1$, there is a randomized algorithm $\mathcal{A}$ with an* NP *oracle such that the following holds for infinitely many integers n. For every circuit $C\colon \{0,1\}^n \to \{0,1\}^{n+1}$ of size at most n^d, there is a string $y_C \in \{0,1\}^n \setminus \mathrm{Range}(C)$ such that $\mathcal{A}(C)$ either outputs y_C or $\perp$, and the probability (over the internal randomness of $\mathcal{A}$) that $\mathcal{A}(C)$ outputs y_C is at least $2/3$. Moreover, this theorem holds in every relativized world.*

As a corollary, for every problem in APEPP, we obtain zero-error pseudodeterministic constructions with an NP oracle and one bit of advice ($\mathsf{FZPP}^{\mathsf{NP}}/_1$) that works infinitely often[4]:

Corollary 2.5 (Informal). *There are infinitely-often zero-error pseudodeterministic constructions for the following objects with an* NP *oracle and one-bit of advice: Ramsey graphs, rigid matrices, two-source extractors, linear codes, hard truth tables, and* $\mathrm{K}^{\mathsf{poly}}$*-random strings.*

Actually, we obtain single-valued $\mathsf{FS}_2\mathsf{P}/_1$ algorithms for the explicit construction problems above, and the pseudodeterministic $\mathsf{FZPP}^{\mathsf{NP}}/_1$ algorithms follow from Cai's theorem that $\mathsf{S}_2\mathsf{P} \subseteq \mathsf{ZPP}^{\mathsf{NP}}$ [8]. We stated them as pseudodeterministic $\mathsf{FZPP}^{\mathsf{NP}}/_1$ algorithms since this notion is better known than the notion of single-valued $\mathsf{FS}_2\mathsf{P}/_1$ algorithms.

Theorem 2.4 is tantalizingly close to an infinitely-often $\mathsf{FP}^{\mathsf{NP}}$ algorithm for Avoid (with the only caveat of being *zero-error* instead of being completely *deterministic*). However, since an $\mathsf{FP}^{\mathsf{NP}}$ algorithm for range avoidance would imply near-maximum circuit lower bounds for E^{NP}, we expect that it would require fundamentally new

[4]The one-bit advice encodes whether our algorithm succeeds on a given input length; it is needed since on bad input lengths, our algorithm might not be pseudodeterministic (i.e., there may not be a canonical answer that is outputted with high probability).

ideas to completely derandomize our algorithm. Previously, Hirahara, Lu, and Ren [25, Theorem 36] presented an infinitely-often pseudodeterministic $\mathsf{FZPP}^{\mathsf{NP}}$ algorithm for the range avoidance problem using n^ε bits of advice, for any small constant $\varepsilon > 0$. Our result improves the above in two aspects: first, we reduce the number of advice bits to 1; second, our techniques relativize but their techniques do not.

Lower bounds against non-uniform computation with maximum advice length. Finally, our results also imply lower bounds against non-uniform computation with maximum advice length. We mention this corollary because it is a stronger statement than circuit lower bounds, and similar lower bounds appeared recently in the literature of super-fast derandomization [15].

Corollary 2.6. *For every $\alpha(n) \geq \omega(1)$ and any constant $k \geq 1$, $\mathsf{S_2E}/_1 \not\subset \mathsf{TIME}[2^{kn}]/_{2^n-\alpha(n)}$. The same holds for $\Sigma_2\mathsf{E}$, $(\Sigma_2\mathsf{E} \cap \Pi_2\mathsf{E})/_1$, and $\mathsf{ZPE}^{\mathsf{NP}}/_1$ in place of $\mathsf{S_2E}/_1$. Moreover, this holds in every relativized world.*

3 INTUITIONS

In the following, we present some high-level intuitions for our new circuit lower bounds.

3.1 Perspective: Single-Valued Constructions

A key perspective in this paper is to view circuit lower bounds (for exponential-time classes) as *single-valued* constructions of hard truth tables. This perspective is folklore; it was also emphasized in recent papers on the range avoidance problem [36, 44].

Let $\Pi \subseteq \{0,1\}^*$ be an *ε-dense* property, i.e., for every integer $N \in \mathbb{N}$, $|\Pi_N| \geq \varepsilon \cdot 2^N$. (In what follows, we use $\Pi_N := \Pi \cap \{0,1\}^N$ to denote the length-N slice of Π.) As a concrete example, let Π_{hard} be the set of hard truth tables, i.e., a string $tt \in \Pi_{\mathrm{hard}}$ if and only if it is the truth table of a function $f: \{0,1\}^n \to \{0,1\}$ whose circuit complexity is at least $2^n/n$, where $n := \log N$. (We assume that $n := \log N$ is an integer.) Shannon's argument [18, 48] shows that Π_{hard} is a 1/2-dense property. We are interested in the following question:

> What is the complexity of *single-valued* constructions for any string in Π_{hard}?

Here, informally speaking, a computation is *single-valued* if each of its computational paths either fails or outputs the *same* value. For example, an NP machine M is a single-valued construction for Π if there is a "canonical" string $y \in \Pi$ such that (1) M outputs y on every accepting computational path; (2) M has at least one accepting computational path. (That is, it is an NPSV construction in the sense of [4, 17, 23, 47].) Similarly, a BPP machine M is a single-valued construction for Π if there is a "canonical" string $y \in \Pi$ such that M outputs y on most (say $\geq 2/3$ fraction of) computational paths. (In other words, single-valued ZPP and BPP constructions are another name for *pseudodeterministic constructions* [20].)[5]

Hence, the task of proving circuit lower bounds is equivalent to the task of *defining*, i.e., single-value constructing, a hard function, in the smallest possible complexity class. For example, a single-valued BPP construction (i.e., pseudodeterministic construction) for Π_{hard} is equivalent to the circuit lower bound $\mathsf{BPE} \not\subset$ i.o.-$\mathsf{SIZE}[2^n/n]$.[6] In this regard, the previous near-maximum circuit lower bound for $\Delta_3\mathsf{E} := \mathsf{E}^{\Sigma_2\mathsf{P}}$ [42] can be summarized in one sentence: The lexicographically first string in Π_{hard} can be constructed in $\Delta_3\mathsf{P} := \mathsf{P}^{\Sigma_2\mathsf{P}}$ (which is necessarily single-valued).

Reduction to Avoid. It was observed in [33, 36] that explicit construction of elements from Π_{hard} is a special case of range avoidance: Let $\mathsf{TT}: \{0,1\}^{N-1} \to \{0,1\}^N$ (here $N = 2^n$) be a circuit that maps the description of a $2^n/n$-size circuit into its 2^n-length truth table (by [18], this circuit can be encoded by $N-1$ bits). Hence, a single-valued algorithm solving Avoid for TT is equivalent to a single-valued construction for Π_{hard}. This explains how our new range avoidance algorithms imply our new circuit lower bounds (as mentioned in subsection 2.2).

In the rest of section 3, we will only consider the special case of Avoid where the input circuit for range avoidance is a P-uniform circuit family. Specifically, let $\{C_n: \{0,1\}^n \to \{0,1\}^{2n}\}_{n\in\mathbb{N}}$ be a P-uniform family of circuits, where $|C_n| \leq \mathrm{poly}(n)$.[7] Our goal is to find an algorithm A such that for infinitely many n, $A(1^n) \in \{0,1\}^{2n} \setminus \mathrm{Range}(C_n)$; see Sections 5.3 and 5.4 of the full version for how to turn this into an algorithm that works for arbitrary input circuit with a single bit of stretch. Also, since from now on we will not talk about truth tables anymore, we will use n instead of N to denote the input length of Avoid instances.

3.2 The Iterative Win-Win Paradigm of [12]

In a recent work, Chen, Lu, Oliveira, Ren, and Santhanam [12] introduced the *iterative win-win* paradigm for explicit constructions, and used that to obtain a polynomial-time pseudodeterministic construction of primes that works infinitely often. Since our construction algorithm closely follows their paradigm, it is instructive to take a detour and give a high-level overview of how the construction from [12] works.[8]

In this paradigm, for a (starting) input length n_0 and some $t = O(\log n_0)$, we will consider an increasing sequence of input lengths $n_0, n_1, \ldots, n_t$ (jumping ahead, we will set $n_{i+1} = n_i^\beta$ for a large constant β), and show that our construction algorithm succeeds on at least one of the input lengths. By varying n_0, we can construct infinitely many such sequences of input lengths that are pairwise

[5]Note that the trivial construction algorithms are not single-valued in general. For example, a trivial $\Sigma_2\mathsf{P} = \mathsf{NP}^{\mathsf{NP}}$ construction algorithm for Π_{hard} is to guess a hard truth table tt and use the NP oracle to verify that tt does not have size-$N/\log N$ circuits; however, different accepting computational paths of this computation would output different hard truth tables. Similarly, a trivial BPP construction algorithm for every dense property Π is to output a random string, but there is no *canonical* answer that is outputted with high probability. In other words, these construction algorithms do not *define* anything; instead, a single-valued construction algorithm should *define* some particular string in Π.

[6]To see this, note that (1) $\mathsf{BPE} \not\subset$ i.o.-$\mathsf{SIZE}[2^n/n]$ implies a simple single-valued BPP construction for Π_{hard}: given $N = 2^n$, output the truth table of L_n (L restricted to n-bit inputs), where $L \in \mathsf{BPE}$ is the hard language not in $\mathsf{SIZE}[2^n/n]$; and (2) assuming a single-valued BPP construction A for Π_{hard}, one can define a hard language L such that the truth table of L_n is the output of $A(1^{2^n})$, and observe that $L \in \mathsf{BPE}$.

[7]We assume that C_n stretches n bits to $2n$ bits instead of $n+1$ bits for simplicity; Korten [36] showed that there is a P^{NP} reduction from the range avoidance problem with stretch $n+1$ to the range avoidance problem with stretch $2n$.

[8]Indeed, for every $1/\mathrm{poly}(n)$-dense property $\Pi \in \mathsf{P}$, they obtained a polynomial-time algorithm A such that for infinitely many $n \in \mathbb{N}$, there exists $y_n \in \Pi_n$ such that $A(1^n)$ outputs y_n with probability at least 2/3. By [2] and the prime number theorem, the set of n-bit primes is such a property.

disjoint, and therefore our algorithm succeeds on infinitely many input lengths.

In more detail, fixing a sequence of input lengths $n_0, n_1, \dots, n_t$ and letting Π be an ε-dense property, for each $i \in \{0, 1, \dots, t\}$, we specify a (deterministic) algorithm ALG_i that takes 1^{n_i} as input and aims to construct an explicit element from Π_{n_i}. We let ALG_0 be the simple brute-force algorithm that enumerates all length-n_0 strings and finds the lexicographically first string in Π_{n_0}; it is easy to see that ALG_0 runs in $T_0 := 2^{O(n_0)}$ time.

The win-or-improve mechanism. The core of [12] is a novel *win-or-improve mechanism*, which is described by a (randomized) algorithm R. Roughly speaking, for input lengths n_i and n_{i+1}, $R(1^{n_i})$ attempts to *simulate* ALG_i *faster by using the oracle* $\Pi_{n_{i+1}}$ (hence it runs in $\mathrm{poly}(n_{i+1})$ time). The crucial property is the following win-win argument:

- **(Win)** Either $R(1^{n_i})$ outputs $\mathsf{ALG}_i(1^{n_i})$ with probability at least 2/3 over its internal randomness,
- **(Improve)** or, from the failure of $R(1^{n_i})$, we can construct an algorithm ALG_{i+1} that outputs an explicit element from $\Pi_{n_{i+1}}$ and runs in $T_{i+1} = \mathrm{poly}(T_i)$ time.

We call the above (Win-or-Improve), since either we have a pseudodeterministic algorithm $R(1^{n_i})$ that constructs an explicit element from Π_{n_i} in $\mathrm{poly}(n_{i+1}) \le \mathrm{poly}(n_i)$ time (since it simulates ALG_i), or we have an *improved* algorithm ALG_{i+1} at the input length n_{i+1} (for example, on input length n_1, the running time of ALG_1 is $2^{O\left(n_1^{1/\beta}\right)} \ll 2^{O(n_1)}$). The (Win-or-Improve) part in [12] is implemented via the Chen–Tell targeted hitting set generator [14] (we omit the details here). Jumping ahead, in this paper, we will implement a similar mechanism using Korten's P^{NP} reduction from the range avoidance problem to constructing hard truth tables [36].

Getting polynomial time. Now we briefly explain why (Win-or-Improve) implies a *polynomial-time* construction algorithm. Let α be an absolute constant such that we always have $T_{i+1} \le T_i^{\alpha}$; we now set $\beta := 2\alpha$. Recall that $n_i = n_{i-1}^{\beta}$ for every i. The crucial observation is the following:

> Although T_0 is much larger than n_0, the sequence $\{T_i\}$ grows slower than $\{n_i\}$.

Indeed, a simple calculation shows that when $t = O(\log n_0)$, we will have $T_t \le \mathrm{poly}(n_t)$; see [12, Section 1.3.1].

For each $0 \le i < t$, if $R(1^{n_i})$ successfully simulates ALG_i, then we obtain an algorithm for input length n_i running in $\mathrm{poly}(n_{i+1}) \le \mathrm{poly}(n_i)$ time. Otherwise, we have an algorithm ALG_{i+1} running in T_{i+1} time on input length n_{i+1}. Eventually, we will hit t such that $T_t \le \mathrm{poly}(n_t)$, in which case ALG_t itself gives a polynomial-time construction on input length n_t. Therefore, we obtain a polynomial-time algorithm on at least one of the input lengths $n_0, n_1, \dots, n_t$.

3.3 Algorithms for Range-Avoidance via Korten's Reduction

Now we describe our new algorithms for Avoid. Roughly speaking, our new algorithm makes use of the iterative win-win argument introduced above, together with an easy-witness style argument [27] and Korten's reduction [36].[9] In the following, we introduce the latter two ingredients and show how to chain them together via the iterative win-win argument.

An easy-witness style argument. Let BF be the $2^{O(n)}$-time brute-force algorithm outputting the lexicographically first non-output of C_n. Our first idea is to consider its *computational history*, a unique $2^{O(n)}$-length string h_{BF} (that can be computed in $2^{O(n)}$ time), and *branch on whether* h_{BF} *has a small circuit or not.* Suppose h_{BF} admits a, say, n^{α}-size circuit for some large α, then we apply an *easy-witness-style* argument [27] to simulate BF by a single-valued $\mathsf{F\Sigma_2P}$ algorithm running in $\mathrm{poly}(n^{\alpha}) = \mathrm{poly}(n)$ time (see subsection 4.2). Hence, we obtained the desired algorithm when h_{BF} is easy.

However, it is less clear how to deal with the other case (when h_{BF} is hard) directly. The crucial observation is that we have gained the following ability: we can generate a string $h_{\mathsf{BF}} \in \{0,1\}^{2^{O(n)}}$ that has circuit complexity at least n^{α}, in only $2^{O(n)}$ time.

Korten's reduction. We will apply Korten's recent work [36] to make use of the "gain" above. So it is worth taking a detour to review the main result of [36]. Roughly speaking, [36] gives **an algorithm that uses a hard truth table f to solve a derandomization task: finding a non-output of the given circuit (that has more output bits than input bits).**[10]

Formally, [36] gives a P^{NP}-computable algorithm $\mathsf{Korten}(C, f)$ that takes as inputs a circuit $C\colon \{0,1\}^n \to \{0,1\}^{2n}$ and a string $f \in \{0,1\}^T$ (think of $n \ll T$), and outputs a string $y \in \{0,1\}^{2n}$. The guarantee is that if the circuit complexity of f is sufficiently larger than the size of C, then the output y is not in the range of C.

This fits perfectly with our "gain" above: for $\beta \ll \alpha$ and $m = n^{\beta}$, $\mathsf{Korten}(C_m, h_{\mathsf{BF}})$ solves Avoid for C_m since the circuit complexity of h_{BF}, n^{α}, is sufficiently larger than the size of C_m. Moreover, $\mathsf{Korten}(C_m, h_{\mathsf{BF}})$ runs in only $2^{O(n)}$ time, which is much less than the brute-force running time $2^{O(m)}$. Therefore, we obtain an improved algorithm for Avoid on input length m.

The iterative win-win argument. What we described above is essentially the first stage of an *win-or-improve mechanism* similar to that from subsection 3.2. Therefore, we only need to iterate the argument above to obtain a polynomial-time algorithm.

For this purpose, we need to consider the computational history of not only BF, but also algorithms of the form $\mathsf{Korten}(C, f)$.[11] For any circuit C and "hard" truth table f, there is a *unique* "computational history" h of $\mathsf{Korten}(C, f)$, and the length of h is upper bounded by $\mathrm{poly}(|f|)$. We are able to prove the following statement akin to the *easy witness lemma* [27]: if h admits a size-s circuit (think of $s \ll T$), then $\mathsf{Korten}(C, f)$ can be simulated by a single-valued

[9] Korten's result was inspired by [29], which proved that the dual weak pigeonhole principle is equivalent to the statement asserting the existence of Boolean functions with exponential circuit complexity in a certain fragment of Bounded Arithmetic.

[10] This is very similar to the classical hardness-vs-randomness connection [28, 43], which can be understood as an algorithm that uses a hard truth table f (i.e., a truth table without small circuits) to solve another derandomization task: estimating the acceptance probability of the given circuit. This explains why one may want to use Korten's algorithm to replace the Chen–Tell targeted generator construction [14] from [12], as they are both hardness-vs-randomness connections.

[11] Actually, we need to consider all algorithms ALG_i defined below and prove the properties of computational history for these algorithms. It turns out that all of ALG_i are of the form $\mathsf{Korten}(C, f)$ (including ALG_0), so in what follows we only consider the computational history of $\mathsf{Korten}(C, f)$.

$\mathsf{F}\Sigma_2\mathsf{P}$ algorithm in time $\mathrm{poly}(s)$; see subsection 4.2 for details on this argument.[12]

Now, following the iterative win-win paradigm of [12], for a (starting) input length n_0 and some $t = O(\log n_0)$, we consider an increasing sequence of input lengths $n_0, n_1, \ldots, n_t$, and show that our algorithm A succeeds on at least one of the input lengths (i.e., $A(1^{n_i}) \in \{0,1\}^{2n_i} \setminus \mathrm{Range}(C_{n_i})$ for some $i \in \{0,1,\ldots,t\}$). For each $i \in \{0,1,\ldots,t\}$, we specify an algorithm ALG_i of the form $\mathsf{Korten}(C_{n_i}, -)$ that aims to solve Avoid for C_{n_i}; in other words, we specify a string $f_i \in \{0,1\}^{T_i}$ for some T_i and let $\mathsf{ALG}_i := \mathsf{Korten}(C_{n_i}, f_i)$.

The algorithm ALG_0 is simply the brute force algorithm BF at input length n_0. (A convenient observation is that we can specify an exponentially long string $f_0 \in \{0,1\}^{2^{O(n_0)}}$ so that $\mathsf{Korten}(C_{n_0}, f_0)$ is equivalent to $\mathsf{BF} = \mathsf{ALG}_0$; see Fact 3.4 in the full version.) For each $0 \le i < t$, to specify ALG_{i+1}, let f_{i+1} denote the history of the algorithm ALG_i, and consider the following win-or-improve mechanism.

(Win) If f_{i+1} admits an n_i^{α}-size circuit (for some large constant α), by our easy-witness argument, we can simulate ALG_i by a $\mathrm{poly}(n_i)$-time single-valued $\mathsf{F}\Sigma_2\mathsf{P}$ algorithm.

(Improve) Otherwise f_{i+1} has circuit complexity at least n_i^{α}, we plug it into Korten's reduction to solve Avoid for $C_{n_{i+1}}$. That is, we take $\mathsf{ALG}_{i+1} := \mathsf{Korten}(C_{n_{i+1}}, f_{i+1})$ as our new algorithm on input length n_{i+1}.

Let $T_i = |f_i|$, then $T_{i+1} \le \mathrm{poly}(T_i)$. By setting $n_{i+1} = n_i^{\beta}$ for a sufficiently large β, a similar analysis as [12] shows that for some $t = O(\log n_0)$ we would have $T_t \le \mathrm{poly}(n_t)$, meaning that ALG_t would be a $\mathrm{poly}(n_t)$-time $\mathsf{FP}^{\mathsf{NP}}$ algorithm (thus also a single-valued $\mathsf{F}\Sigma_2\mathsf{P}$ algorithm) solving Avoid for C_{n_t}. Putting everything together, we obtain a polynomial-time single-valued $\mathsf{F}\Sigma_2\mathsf{P}$ algorithm that solves Avoid for at least one of the C_{n_i}.

The hardness condenser perspective. Below we present another perspective on the construction above which may help the reader understand it better. In the following, we fix $C_n\colon \{0,1\}^n \to \{0,1\}^{2n}$ to be the truth table generator $\mathsf{TT}_{n,2n}$ that maps an n-bit description of a $\log(2n)$-input circuit into its length-$2n$ truth table. Hence, instead of solving Avoid in general, our goal here is simply *constructing hard truth tables* (or equivalently, proving circuit lower bounds).

We note that $\mathsf{Korten}(\mathsf{TT}_{n,2n}, f)$ can then be interpreted as a *hardness condenser* [7]:[13] Given a truth table $f \in \{0,1\}^T$ whose circuit complexity is sufficiently larger than n, it outputs a length-$2n$ truth table that is maximally hard (i.e., without $n/\log n$-size circuits). The win-or-improve mechanism can be interpreted as an iterative application of this hardness condenser.

At the stage i, we consider the algorithm

$$\mathsf{ALG}_i := \mathsf{Korten}(\mathsf{TT}_{n_i,2n_i}, f_i),$$

which runs in $T_i \approx |f_i|$ time and creates (roughly) n_i bits of hardness. (That is, the circuit complexity of the output of ALG_i is roughly n_i.) In the **(Win)** case above, ALG_i admits an n_i^{α}-size history f_{i+1} (with length approximately $|f_i|$) and can therefore be simulated in $\mathsf{F}\Sigma_2\mathsf{P}$. The magic is that in the **(Improve)** case, we actually have access to *much more hardness than* n_i: the history string f_{i+1} has $n_i^{\alpha} \gg n_i$ bits of hardness. So we can *distill* these hardness by applying the condenser to f_{i+1} to obtain a maximally hard truth tables of length $2n_{i+1} = 2n_i^{\beta}$, establish the next algorithm $\mathsf{ALG}_{i+1} := \mathsf{Korten}(\mathsf{TT}_{n_{i+1},2n_{i+1}}, f_{i+1})$, and keep iterating.

Observe that the string f_{i+1} above has $n_i^{\alpha} > n_i^{\beta} = n_{i+1}$ bits of hardness. Since $|f_{i+1}| \approx |f_i|$ and $n_{i+1} = n_i^{\beta}$, the process above creates *harder and harder* strings, until $|f_{i+1}| \le n_{i+1} \le n_i^{\alpha}$, so the **(Win)** case must happen at some point.

4 PROOF OVERVIEW

In this section, we elaborate on the computational history of Korten and how the easy-witness-style argument gives us $\mathsf{F}\Sigma_2\mathsf{P}$ and $\mathsf{FS}_2\mathsf{P}$ algorithms.

4.1 Korten's Reduction

We first review the key concepts and results from [36] that are needed for us. Given a circuit $C\colon \{0,1\}^n \to \{0,1\}^{2n}$ and a parameter $T \ge 2n$, Korten builds another circuit $\mathsf{GGM}_T[C]$ stretching n bits to T bits as follows:[14]

- On input $x \in \{0,1\}^n$, we set $v_{0,0} = x$. For simplicity, we assume that $T/n = 2^k$ for some $k \in \mathbb{N}$. We build a full binary tree with $k+1$ layers; see Figure 1 for an example with $k = 3$.
- For every $i \in \{0,1,\ldots,k-1\}$ and $j \in \{0,1,\ldots,2^i - 1\}$, we set $v_{i+1,2j}$ and $v_{i+1,2j+1}$ to be the first n bits and the last n bits of $C(v_{i,j})$, respectively.
- The output of $\mathsf{GGM}_T[C](x)$ is defined to be the concatenation of $v_{k,0}, v_{k,1}, \ldots, v_{k,2^k-1}$.

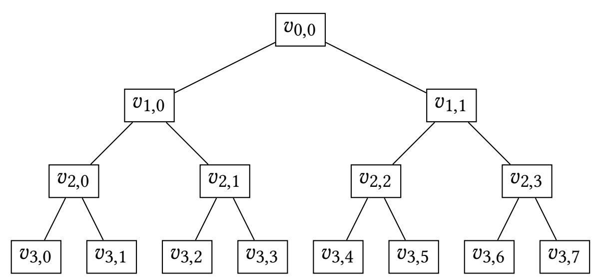

Figure 1: An illustration of the GGM Tree, in which, for instance, it holds that $(v_{3,4}, v_{3,5}) = C(v_{2,2})$.

The following properties of $\mathsf{GGM}_T[C]$ are established in [36], which will be useful for us:

(1) Given $i \in [T]$, C and $x \in \{0,1\}^n$, by traversing the tree from the root towards the leaf with the i-th bit, one can compute the i-th bit of $\mathsf{GGM}_T[C](x)$ in $\mathrm{poly}(\mathsf{SIZE}(C), \log T)$ time. Consequently, for every x, $\mathsf{GGM}_T[C](x)$ has circuit complexity at most $\mathrm{poly}(\mathsf{SIZE}(C), \log T)$.

[12] With an "encoded" version of history and more effort, we are able to simulate $\mathsf{Korten}(C, f)$ by a single-valued $\mathsf{FS}_2\mathsf{P}$ algorithm in time $\mathrm{poly}(s)$, and that is how our $\mathsf{S}_2\mathsf{E}$ lower bound is proved; see subsection 4.3 for details.

[13] A hardness condenser takes a long truth table f with certain hardness and outputs a shorter truth table with similar hardness.

[14] We use the name GGM because the construction is similar to the pseudorandom function generator of Goldreich, Goldwasser, and Micali [21].

(2) There is a P^{NP} algorithm $\mathsf{Korten}(C, f)$ that takes an input $f \in \{0,1\}^T \setminus \mathrm{Range}(\mathsf{GGM}_T[C])$ and outputs a string $u \in \{0,1\}^{2n} \setminus \mathrm{Range}(C)$. Note that this is a reduction from solving Avoid for C to solving Avoid for $\mathsf{GGM}_T[C]$.

In particular, letting f be a truth table whose circuit complexity is sufficiently larger than $\mathsf{SIZE}(C)$, by the first property above, it is not in $\mathrm{Range}(\mathsf{GGM}_T[C])$, and therefore $\mathsf{Korten}(C, f)$ solves Avoid for C. This confirms our description of Korten in subsection 2.2.

4.2 Computational History of Korten and an Easy-Witness Argument for $\mathsf{F\Sigma_2P}$ Algorithms

The algorithm $\mathsf{Korten}(C, f)$ works as follows: we first view f as the labels of the last layer of the binary tree, and try to reconstruct the whole binary tree, layer by layer (start from the bottom layer to the top layer, within each layer, start from the rightmost node to the leftmost one), by filling the labels of the intermediate nodes. To fill $v_{i,j}$, we use an NP oracle to find the lexicographically first string $u \in \{0,1\}^n$ such that $C(u) = v_{i+1,2j} \circ v_{i+1,2j+1}$, and set $v_{i,j} = u$. If no such u exists, the algorithm stops and report $v_{i+1,2j} \circ v_{i+1,2j+1}$ as the solution to Avoid for C. Observe that this reconstruction procedure must stop somewhere, since if it successfully reproduces all the labels in the binary tree, we would have $f = \mathsf{GGM}_T[C](v_{0,0}) \in \mathrm{Range}(\mathsf{GGM}_T[C])$, contradicting the assumption. For details, see [36, Theorem 7] or Lemma 3.3 of the full version.

The computational history of Korten. The algorithm described above induces a natural description of the computational history of Korten, denoted as $\mathsf{History}(C, f)$, as follows: the index $(i_\star, j_\star)$ when the algorithm stops (i.e., the algorithm fails to fill in $v_{i_\star, j_\star}$) concatenated with the labels of all the nodes generated by $\mathsf{Korten}(C, f)$ (for the intermediate nodes with no label assigned, we set their labels to a special symbol $\perp$); see Figure 2 for an illustration. This history has length at most $5T$, and for convenience, we pad additional zeros at the end of it so that its length is exactly $5T$.

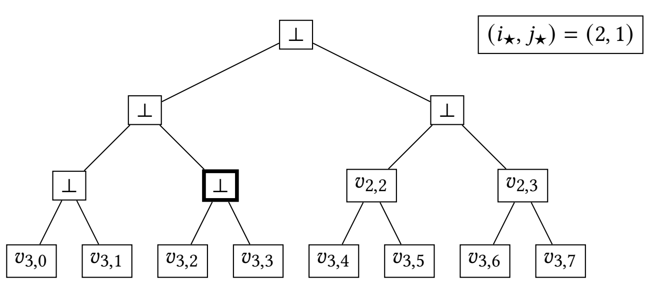

Figure 2: An illustration of the history of $\mathsf{Korten}(C, f)$**. Here we have** $\mathsf{History}(C, f) = (2,1) \circ \perp\perp\perp\perp\perp \circ v_{2,2} \circ v_{2,3} \circ v_{3,0} \circ \ldots \circ v_{3,7}$ **and** $\mathsf{Korten}(C, f) = v_{3,2} \circ v_{3,3}$**.**

A local characterization of $\mathsf{History}(C, f)$. The crucial observation we make on $\mathsf{History}(C, f)$ is that it admits a local characterization in the following sense: there is a family of local constraints $\{\psi_x\}_{x \in \{0,1\}^{\mathrm{poly}(n)}}$, where each $\psi_x \colon \{0,1\}^{5T} \times \{0,1\}^T \to \{0,1\}$ reads only $\mathrm{poly}(n)$ many bits of its input (we think about it as a local constraint since usually $n \ll T$), such that for fixed f, $\mathsf{History}(C, f) \circ f$ is the unique string making all the ψ_x outputting 1.

The constraints are follows: (1) for every leaf node $v_{k,i}$, its content is consistent with the corresponding block in f; (2) all labels at or before node $(i_\star, j_\star)$ are $\perp$;[15] (3) for every $z \in \{0,1\}^n$, $C(z) \neq v_{i_\star+1, 2j_\star} \circ v_{i_\star+1, 2j_\star+1}$ (meaning the algorithm fails at $v_{i_\star, j_\star}$); (4) for every (i, j) after $(i_\star, j_\star)$, $C(v_{i,j}) = v_{i+1,2j} \circ v_{i+1,2j+1}$ ($v_{i,j}$ is the correct label); (5) for every (i, j) after $(i_\star, j_\star)$ and for every $v' < v_{i,j}$, $C(v') \neq v_{i+1,2j} \circ v_{i+1,2j+1}$ ($v_{i,j}$ is the lexicographically first correct label). It is clear that each of these constraints above only reads $\mathrm{poly}(n)$ many bits from the input and a careful examination shows they precisely **define** the string $\mathsf{History}(C, f)$.

A more intuitive way to look at these local constraints is to treat them as a $\mathrm{poly}(n)$-time oracle algorithm V_{History} that takes a string $x \in \mathrm{poly}(n)$ as input and two strings $h \in \{0,1\}^{5T}$ and $f \in \{0,1\}^T$ as oracles, and we simply let $V_{\mathsf{History}}^{h,f}(x) = \psi_x(h \circ f)$. Since the constraints above are all very simple and only read $\mathrm{poly}(n)$ bits of $h \circ f$, V_{History} runs in $\mathrm{poly}(n)$ time. In some sense, V_{History} is a local Π_1 verifier: it is local in the sense that it only queries $\mathrm{poly}(n)$ bits from its oracles, and it is Π_1 since it needs a universal quantifier over $x \in \{0,1\}^{\mathrm{poly}(n)}$ to perform all the checks.

$\mathsf{F\Sigma_2P}$ *algorithms.* Before we proceed, we give a formal definition of a single-valued $\mathsf{F\Sigma_2P}$ algorithm A. Here A is implemented by an algorithm V_A taking an input x and two $\mathrm{poly}(|x|)$-length witnesses π_1 and π_2. We say $A(x)$ outputs a string $z \in \{0,1\}^\ell$ (we assume $\ell = \ell(x)$ can be computed in polynomial time from x) if z is the *unique* length-ℓ string such that the following hold:

- there exists π_1 such that for every π_2, $V_{\mathsf{History}}(x, \pi_1, \pi_2, z) = 1$.[16]

We can view V_{History} as a verifier that checks whether z is the desired output using another universal quantifier: given a proof π_1 and a string $z \in \{0,1\}^\ell$. A accepts z if and only if *for every* π_2, $V_{\mathsf{History}}(x, \pi_1, \pi_2, z) = 1$. That is, A can perform exponentially many checks on π_1 and z, each taking $\mathrm{poly}(|x|)$ time.

The easy-witness argument. Now we are ready to elaborate on the easy-witness argument mentioned in subsection 2.2. Recall that at stage i, we have $\mathsf{ALG}_i = \mathsf{Korten}(C_{n_i}, f_i)$ and $f_{i+1} = \mathsf{History}(C_{n_i}, f_i)$ (the history of ALG_i). Assuming that f_{i+1} admits a $\mathrm{poly}(n_i)$-size circuit, we want to show that $\mathsf{Korten}(C_{n_i}, f_i)$ can be simulated by a $\mathrm{poly}(n_i)$-time single-valued $\mathsf{F\Sigma_2P}$ algorithm.

Observe that for every $t \in [i+1]$, f_{t-1} is simply a substring of f_t since $f_t = \mathsf{History}(C_{n_{t-1}}, f_{t-1})$. Therefore, f_{i+1} admitting a $\mathrm{poly}(n_i)$-size circuit implies that all f_t admit $\mathrm{poly}(n_i)$-size circuits for $t \in [i]$. We can then implement A as follows: the proof π_1 is a $\mathrm{poly}(n_i)$-size circuit C_{i+1} supposed to compute f_{i+1}, from which one can obtain in polynomial time a sequence of circuits $C_1, \ldots, C_i$ that are supposed to compute $f_1, \ldots, f_i$, respectively. (Also, one can easily construct a $\mathrm{poly}(n_0)$-size circuit C_0 computing f_0.) Next, for every $t \in \{0, 1, \ldots, i\}$, A checks whether $(\mathtt{C}_{t+1}) \circ (\mathtt{C}_t)$ satisfies all the

[15] We say that (i, j) is before (after) $(i_\star, j_\star)$ if the pair (i, j) is lexicographically smaller (greater) than $(i_\star, j_\star)$.

[16] Note that our definition here is different from the formal definition we used in the full version of this paper. But from this definition, it is easier to see why $\mathsf{F\Sigma_2P}$ algorithms for constructing hard truth tables imply circuit lower bounds for $\Sigma_2\mathsf{E}$.

local constraints ψ_x's from the characterization of $\mathsf{History}(C_{n_t}, f_t)$. In other words, A checks whether $V_{\mathsf{History}}^{C_{t+1},C_t}(x) = 1$ for all $x \in \{0,1\}^{\mathsf{poly}(n_t)}$.

The crucial observation is that since all the C_t have size $\mathsf{poly}(n_i)$, each check above can be implemented in $\mathsf{poly}(n_i)$ time as they only read at most $\mathsf{poly}(n_i)$ bits from their input, despite that $(\mathtt{C_{t+1}}) \circ (\mathtt{C_t})$ itself can be much longer than $\mathsf{poly}(n_i)$. Assuming that all the checks of A above are passed, by induction we know that $f_{t+1} = \mathsf{History}(C_{n_t}, f_t)$ for every $t \in \{0,1,\dots,i\}$. Finally, A checks whether z corresponds to the answer described in $(\mathtt{C_{i+1}}) = \mathtt{f_{i+1}}$.

4.3 Selectors and an Easy-Witness Argument for $\mathsf{FS_2P}$ Algorithms

Finally, we discuss how to implement the easy-witness argument above with a single-valued $\mathsf{FS_2P}$ algorithm. It is known that any single-valued $\mathsf{FS_2BPP}$ algorithm can be converted into an equivalent single-valued $\mathsf{FS_2P}$ algorithm outputting the same string [10, 45]. Therefore, in the following we aim to give a single-valued $\mathsf{FS_2BPP}$ algorithm for solving range avoidance, which is easier to achieve.

$\mathsf{FS_2BPP}$ *algorithms and randomized selectors.* Before we proceed, we give a formal definition of a single-valued $\mathsf{FS_2BPP}$ algorithm A. We implement A by a randomized algorithm V_A that takes an input x and two $\mathsf{poly}(|x|)$-length witnesses π_1 and π_2.[17] We say that $A(x)$ outputs a string $z \in \{0,1\}^\ell$ (we assume $\ell = \ell(x)$ can be computed in polynomial time from x) if the following hold:

- there exists a string h such that for every π, both $V_A(x,h,\pi)$ and $V_A(x,\pi,h)$ output z with probability at least $2/3$. (Note that such z must be unique if it exists.)

Actually, our algorithm A will be implemented as a randomized *selector*: given two potential proofs π_1 and π_2, it first selects the correct one and then outputs the string z induced by the correct proof.[18]

Recap. Revising the algorithm in subsection 3.3, our goal now is to give an $\mathsf{FS_2BPP}$ simulation of $\mathsf{Korten}(C_{n_i}, f_i)$, assuming that $\mathsf{History}(C_{n_i}, f_i)$ admits a small circuit. Similar to the local Π_1 verifier used in the case of $\mathsf{F\Sigma_2P}$ algorithms, now we consider a local randomized selector V_{select} which takes oracles $\pi_1, \pi_2 \in \{0,1\}^{5T}$ and $f \in \{0,1\}^T$ such that if exactly one of the π_1 and π_2 is $\mathsf{History}(C,f)$, V_{select} outputs its index with high probability.

Assuming that $f_{i+1} = \mathsf{History}(C_{n_i}, f_i)$ admits a small circuit, one can similarly turn V_{select} into a single-valued $\mathsf{FS_2BPP}$ algorithms A computing $\mathsf{Korten}(C_{n_i}, f_i)$: treat two proofs π_1 and π_2 as two small circuits C and D both supposed to compute f_{i+1}, from C and D we can obtain a sequence of circuits $\{C_t\}$ and $\{D_t\}$ supposed to compute the f_t for $t \in [i]$. Then we can use the selector V_{select} to decide for each $t \in [i+1]$ which of the C_t and D_t is the correct circuit for f_t. Finally, we output the answer encoded in the selected circuit for f_{i+1}.[19]

Observation: it suffices to find the first differing node label. Ignore the $(i_\star, j_\star)$ part of the history for now. Let $\{v^1_{i,j}\}$ and $\{v^2_{i,j}\}$ be the node labels encoded in π_1 and π_2, respectively. We also assume that exactly one of them corresponds to the correct node labels in $\mathsf{History}(C,f)$. The crucial observation here is that, since the correct node labels are generated by a deterministic procedure *node by node* (from bottom to top and from rightmost to leftmost), it is possible to tell which of the $\{v^1_{i,j}\}$ and $\{v^2_{i,j}\}$ is correct given the largest (i',j') such that $v^1_{i',j'} \neq v^2_{i',j'}$. (Note that since all (i,j) are processed by $\mathsf{Korten}(C,f)$ in reverse lexicographic order, this (i',j') corresponds to the first node label that the wrong process differs from the correct process, so we call this the first differing point.)

In more detail, assuming we know this (i',j'), we proceed by discussing several cases. First of all, if (i',j') corresponds to a leaf, then one can query f to figure out which of $v^1_{i',j'}$ and $v^2_{i',j'}$ is consistent with the corresponding block in f. Now we can assume (i',j') corresponds to an intermediate node. Since (i',j') is the first differing point, we know that $v^1_{i'+1,2j'} \circ v^1_{i'+1,2j'+1} = v^2_{i'+1,2j'} \circ v^2_{i'+1,2j'+1}$ (we let this string to be α for convenience). By the definition of $\mathsf{History}(C,f)$, it follows that the correct $v_{i',j'}$ should be uniquely determined by α, which means the selector only needs to read α, $v^1_{i',j'}$, and $v^2_{i',j'}$, and can then be implemented by a somewhat tedious case analysis (so it is local). We refer readers to the proof of Lemma 5.5 in the full version for the details and only highlight the most illuminating case here: if both of $v^1_{i',j'}$ and $v^2_{i',j'}$ are good (we say a string γ is good, if $\gamma \neq \perp$ and $C(\gamma) = \alpha$), we select the lexicographically smaller one. To handle the $(i_\star, j_\star)$ part, one needs some additional case analysis. We omit the details here and refer the reader to the proof in the full version.

The takeaway here is that if we can find the first differing label (i',j'), then we can construct the selector V_{select} and hence the desired single-valued $\mathsf{FS_2BPP}$ algorithm.

Encoded history. However, the above assumes the knowledge of (i',j'). In general, if one is only given oracle access to $\{v^1_{i,j}\}$ and $\{v^2_{i,j}\}$, there is no $\mathsf{poly}(n)$-time oracle algorithm computing (i',j') because there might be exponentially many nodes. To resolve this issue, we will encode $\{v^1_{i,j}\}$ and $\{v^2_{i,j}\}$ via Reed–Muller codes.

Formally, recall that $\mathsf{History}(C,f)$ is the concatenation of $(i_\star, j_\star)$ and the string S, where S is the concatenation of all the labels on the binary tree. We now define the encoded history, denoted as $\widetilde{\mathsf{History}}(C,f)$, as the concatenation of $(i_\star, j_\star)$ and *a Reed–Muller encoding* of S. The new selector is given oracle access to two candidate encoded histories together with f. By applying low-degree tests and self-correction of polynomials, we can assume that the Reed–Muller parts of the two candidates are indeed low-degree polynomials. Then we can use a reduction to polynomial identity testing to compute the first differing point between $\{v^1_{i,j}\}$ and $\{v^2_{i,j}\}$ in randomized polynomial time. See the proof of Lemma 5.3

[17] $\mathsf{FS_2P}$ algorithms are the special case of $\mathsf{FS_2BPP}$ algorithms where the algorithm V_A is *deterministic*.

[18] If both proofs are correct or neither proofs are correct, it can select an arbitrary one. The condition only applies when exactly one of the proofs is correct.

[19] However, for the reasons to be explained below, we will actually work with the encoded history instead of the history, which entails a lot of technical challenges in the actual proof.

in the full version for the details. This part is similar to the selector construction from [24].

5 DISCUSSIONS

We conclude the introduction by discussing some related works.

5.1 Previous Approach: Karp–Lipton Collapses and the Half-Exponential Barrier

In the following, we elaborate on the half-exponential barrier mentioned earlier in the introduction.[20] Let C be a "typical" uniform complexity class containing P, a *Karp–Lipton collapse* to C states that if a large class (say EXP) has polynomial-size circuits, then this class collapses to C. For example, there is a Karp–Lipton collapse to $C = \Sigma_2\mathsf{P}$:

> Suppose $\mathsf{EXP} \subseteq \mathsf{P}/_{\mathsf{poly}}$, then $\mathsf{EXP} = \Sigma_2\mathsf{P}$. ([32], attributed to Albert Meyer)

Now, assuming that $\mathsf{EXP} \subseteq \mathsf{P}/_{\mathsf{poly}} \implies \mathsf{EXP} = C$, the following win-win analysis implies that C-EXP, the exponential-time version of C, is not in $\mathsf{P}/_{\mathsf{poly}}$: (1) if $\mathsf{EXP} \not\subseteq \mathsf{P}/_{\mathsf{poly}}$, then of course C-$\mathsf{EXP} \supseteq \mathsf{EXP}$ does not have polynomial-size circuits; (2) otherwise $\mathsf{EXP} \subseteq \mathsf{P}/_{\mathsf{poly}}$. We have $\mathsf{EXP} = C$ and by padding $\mathsf{EEXP} = C$-EXP. Since EEXP contains a function of maximum circuit complexity by direct diagonalization, it follows that C-EXP does not have polynomial-size circuits.

Karp–Lipton collapses are known for the classes $\Sigma_2\mathsf{P}$ [32], $\mathsf{ZPP}^{\mathsf{NP}}$ [5], $\mathsf{S}_2\mathsf{P}$ [8] (attributed to Samik Sengupta), PP, MA [3, 40], and $\mathsf{ZPP}^{\mathsf{MCSP}}$ [26]. All the aforementioned super-polynomial circuit lower bounds for $\Sigma_2\mathsf{EXP}$, $\mathsf{ZPEXP}^{\mathsf{NP}}$, $\mathsf{S}_2\mathsf{EXP}$, PEXP, MA-EXP, and $\mathsf{ZPEXP}^{\mathsf{MCSP}}$ are proven in this way.[21]

The half-exponential barrier. The above argument is very successful at proving various super-polynomial lower bounds. However, a closer look shows that it is only capable of proving *sub-half-exponential* circuit lower bounds. Indeed, suppose we want to show that C-EXP does not have circuits of size $f(n)$. We will have to perform the following win-win analysis:

- if $\mathsf{EXP} \not\subseteq \mathsf{SIZE}[f(n)]$, then of course C-$\mathsf{EXP} \supseteq \mathsf{EXP}$ does not have circuits of size $f(n)$;
- if $\mathsf{EXP} \subseteq \mathsf{SIZE}[f(n)]$, then (a scaled-up version of) the Karp–Lipton collapse implies that EXP can be computed by a C machine of $\mathrm{poly}(f(n))$ time. Note that $\mathsf{TIME}[2^{\mathrm{poly}(f(n))}]$ does not have circuits of size $f(n)$ by direct diagonalization. By padding, $\mathsf{TIME}[2^{\mathrm{poly}(f(n))}]$ can be computed by a C machine of $\mathrm{poly}(f(\mathrm{poly}(f(n))))$ time. Therefore, if f is sub-half-exponential (meaning $f(\mathrm{poly}(f(n))) = 2^{o(n)}$), then C-EXP does not have circuits of size $f(n)$.

Intuitively speaking, the two cases above are *competing with each other*: we cannot get exponential lower bounds in both cases.

[20] A function $f\colon \mathbb{N} \to \mathbb{N}$ is *sub-half-exponential* if $f(f(n)^c) = 2^{o(n)}$ for every constant $c \geq 1$, i.e., composing f twice yields a sub-exponential function. For example, for constants $c \geq 1$ and $\varepsilon > 0$, the functions $f(n) = n^c$ and $f(n) = 2^{\log^c n}$ are sub-half-exponential, but the functions $f(n) = 2^{n^\varepsilon}$ and $f(n) = 2^{\varepsilon n}$ are not.

[21] There are some evidences that Karp–Lipton collapses are essential for proving circuit lower bounds [13].

5.2 Implications for the Missing-String Problem?

In the Missing-String problem, we are given a list of m strings $x_1, x_2, \dots, x_m \in \{0,1\}^n$ where $m < 2^n$, and the goal is to output any length-n string y that does not appear in $\{x_1, x_2, \dots, x_m\}$. Vyas and Williams [51] connected the circuit complexity of Missing-String with the (relativized) circuit complexity of $\Sigma_2\mathsf{E}$:

Theorem 5.1 ([51, Theorem 32], Informal). *The following are equivalent:*

- $\Sigma_2\mathsf{E}^A \not\subseteq$ i.o.-$\mathsf{SIZE}^A[2^{\Omega(n)}]$ *for every oracle A;*
- *for $M = 2^{N^{\Omega(1)}}$, the Missing-String problem can be solved by a "good" circuit family (roughly speaking, a uniform family of depth-3 AC^0 circuits of size $2^{N^{O(1)}}$ and bottom fan-in* $\mathrm{poly}(N)$).

The intuition behind Theorem 5.1 is roughly as follows. For every oracle A, the set of truth tables with low A-oracle circuit complexity induces an instance for Missing-String, and solving this instance gives us a hard truth table relative to A. If the algorithm for Missing-String is a uniform AC^0 circuit of depth 3, then the hard function is inside $\Sigma_2\mathsf{E}^A$.

However, despite our Theorem 2.1 being completely relativizing, it does not seem to imply any non-trivial depth-3 AC^0 circuit for Missing-String. The reason is the heavy win-win analysis *across multiple input lengths*: for each $0 \leq i < t$, we have a single-valued $\mathsf{F}\Sigma_2\mathsf{P}$ construction algorithm for hard truth tables relative to oracle A on input length n_i, but this algorithm needs access to $A_{n_{i+1}}$, *a higher input length of* A. Translating this into the language of Missing-String, we obtain a weird-looking depth-3 AC^0 circuit that takes as input a *sequence* of Missing-String instances $\mathcal{I}_{n_0}, \mathcal{I}_{n_1}, \dots, \mathcal{I}_{n_t}$ (where *each* $\mathcal{I}_{n_i} \subseteq \{0,1\}^{n_i}$ is a set of strings), looks at all of the instances (or, at least $\mathcal{I}_{n_i}$ and $\mathcal{I}_{n_{i+1}}$), and outputs a purportedly missing string of $\mathcal{I}_{n_i}$. It is guaranteed that for at least one input length i, the output string is indeed a missing string of $\mathcal{I}_{n_i}$. However, if our algorithm is only given one instance $\mathcal{I} \subseteq \{0,1\}^n$, without assistance from a larger input length, it does not know how to find any missing string of $\mathcal{I}$.

6 SUBSEQUENT DEVELOPMENTS

Just one month after our paper was posted online, Li [39] strengthened our results and removed the need of the iterative win-win argument. This allows [39] to prove that:

Theorem 6.1 ([39]). *The following are true:*

- $\mathsf{S}_2\mathsf{E} \not\subseteq$ i.o.-$\mathsf{SIZE}[2^n/n]$. *Consequently, the classes $\Sigma_2\mathsf{E} \cap \Pi_2\mathsf{E}$ and $\mathsf{ZPE}^{\mathsf{NP}}$ also admit the same* almost-everywhere *near-maximum circuit lower bounds. Moreover, this holds in every relativized world.*
- *There is a single-valued $\mathsf{FS}_2\mathsf{P}$ algorithm for the range avoidance problem that works on* every *input length. Consequently, there are zero-error pseudodeterministic polynomial-time constructions for Ramsey graphs, rigid matrices, two-source extractors, linear codes, hard truth tables, and $\mathrm{K}^{\mathsf{poly}}$-random strings, with an* NP *oracle.*
- *There is a uniform family of quasi-polynomial-size depth-3 AC^0 circuit solving the Missing-String problem.*

Compared to our results, Theorem 6.1 holds on almost every input length and does not require the advice bit.

Following our work, the proof of [39] also utilizes the history of Korten's reduction. The crucial insight of [39] is that a variant of "history" (called Histree in [39, Definition 3.5]) *always* have succinct descriptions. Instead, our proof needs to branch on whether our History has succinct descriptions and perform a win-win analysis.

ACKNOWLEDGMENTS

Part of the work was done when all authors were participating in the Meta-Complexity program at the Simons Institute. Lijie Chen is supported by a Miller Research Fellowship. Shuichi Hirahara is supported by JST, PRESTO Grant Number JPMJPR2024, Japan. Hanlin Ren received support from DIMACS through grant number CCF-1836666 from the National Science Foundation. We thank Oliver Korten, Zhenjian Lu, Igor C. Oliveira, Rahul Santhanam, Roei Tell, and Ryan Williams for helpful discussions. We also want to thank Jiatu Li, Igor C. Oliveira, and Roei Tell for comments on an early draft of the paper.

REFERENCES

[1] Scott Aaronson. 2006. Oracles Are Subtle But Not Malicious. In *CCC*. IEEE Computer Society, 340–354. https://doi.org/10.1109/CCC.2006.32

[2] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. 2004. PRIMES Is in P. *Annals of Mathematics* 160, 2 (2004), 781–793. https://doi.org/10.4007/annals.2004.160.781

[3] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. 1993. BPP Has Subexponential Time Simulations Unless EXPTIME has Publishable Proofs. *Computational Complexity* 3 (1993), 307–318. https://doi.org/10.1007/BF01275486

[4] Ronald V. Book, Timothy J. Long, and Alan L. Selman. 1985. Qualitative Relativizations of Complexity Classes. *J. Comput. Syst. Sci.* 30, 3 (1985), 395–413. https://doi.org/10.1016/0022-0000(85)90053-4

[5] Nader H. Bshouty, Richard Cleve, Ricard Gavaldà, Sampath Kannan, and Christino Tamon. 1996. Oracles and Queries That Are Sufficient for Exact Learning. *J. Comput. Syst. Sci.* 52, 3 (1996), 421–433. https://doi.org/10.1006/jcss.1996.0032

[6] Harry Buhrman, Lance Fortnow, and Thomas Thierauf. 1998. Nonrelativizing Separations. In *CCC*. 8–12. https://doi.org/10.1109/CCC.1998.694585

[7] Joshua Buresh-Oppenheim and Rahul Santhanam. 2006. Making Hard Problems Harder. In *CCC*. IEEE Computer Society, 73–87. https://doi.org/10.1109/CCC.2006.26

[8] Jin-yi Cai. 2007. $\mathsf{S}_2^P \subseteq \mathsf{ZPP}^{\mathsf{NP}}$. *J. Comput. Syst. Sci.* 73, 1 (2007), 25–35. https://doi.org/10.1016/j.jcss.2003.07.015

[9] Jin-yi Cai, Venkatesan T. Chakaravarthy, Lane A. Hemaspaandra, and Mitsunori Ogihara. 2005. Competing provers yield improved Karp–Lipton collapse results. *Inf. Comput.* 198, 1 (2005), 1–23. https://doi.org/10.1016/j.ic.2005.01.002

[10] Ran Canetti. 1996. More on BPP and the Polynomial-Time Hierarchy. *Inf. Process. Lett.* 57, 5 (1996), 237–241. https://doi.org/10.1016/0020-0190(96)00016-6

[11] Eshan Chattopadhyay and David Zuckerman. 2019. Explicit two-source extractors and resilient functions. *Annals of Mathematics* 189, 3 (2019), 653–705. https://doi.org/10.4007/annals.2019.189.3.1

[12] Lijie Chen, Zhenjian Lu, Igor C. Oliveira, Hanlin Ren, and Rahul Santhanam. 2023. Polynomial-Time Pseudodeterministic Construction of Primes. In *FOCS*. IEEE, 1261–1270. https://doi.org/10.1109/FOCS57990.2023.00074

[13] Lijie Chen, Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. 2019. Relations and Equivalences Between Circuit Lower Bounds and Karp–Lipton Theorems. In *CCC (LIPIcs, Vol. 137)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 30:1–30:21. https://doi.org/10.4230/LIPIcs.CCC.2019.30

[14] Lijie Chen and Roei Tell. 2021. Hardness vs Randomness, Revised: Uniform, Non-Black-Box, and Instance-Wise. In *FOCS*. 125–136. https://doi.org/10.1109/FOCS52979.2021.00021

[15] Lijie Chen and Roei Tell. 2021. Simple and fast derandomization from very hard functions: eliminating randomness at almost no cost. In *STOC*. 283–291. https://doi.org/10.1145/3406325.3451059

[16] Paul Erdős. 1959. Graph theory and probability. *Canadian Journal of Mathematics* 11 (1959), 34–38. https://doi.org/10.4153/CJM-1959-003-9

[17] Stephen A. Fenner, Steven Homer, Mitsunori Ogiwara, and Alan L. Selman. 1993. On Using Oracles That Compute Values. In *STACS (Lecture Notes in Computer Science, Vol. 665)*. Springer, 398–407. https://doi.org/10.1007/3-540-56503-5_40

[18] Gudmund Skovbjerg Frandsen and Peter Bro Miltersen. 2005. Reviewing bounds on the circuit size of the hardest functions. *Information Processing Letters* 95, 2 (2005), 354–357. https://doi.org/10.1016/j.ipl.2005.03.009

[19] Karthik Gajulapalli, Alexander Golovnev, Satyajeet Nagargoje, and Sidhant Saraogi. 2023. Range Avoidance for Constant Depth Circuits: Hardness and Algorithms. In *APPROX/RANDOM (LIPIcs, Vol. 275)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 65:1–65:18. https://doi.org/10.4230/LIPIcs.APPROX/RANDOM.2023.65

[20] Eran Gat and Shafi Goldwasser. 2011. Probabilistic Search Algorithms with Unique Answers and Their Cryptographic Applications. *Electron. Colloquium Comput. Complex.* TR11-136 (2011). https://eccc.weizmann.ac.il/report/2011/136

[21] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. 1986. How to construct random functions. *Journal of the ACM* 33, 4 (1986), 792–807. https://doi.org/10.1145/6490.6503

[22] Venkatesan Guruswami, Xin Lyu, and Xiuhan Wang. 2022. Range Avoidance for Low-Depth Circuits and Connections to Pseudorandomness. In *APPROX/RANDOM (LIPIcs, Vol. 245)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 20:1–20:21. https://doi.org/10.4230/LIPIcs.APPROX/RANDOM.2022.20

[23] Lane A. Hemaspaandra, Ashish V. Naik, Mitsunori Ogihara, and Alan L. Selman. 1996. Computing Solutions Uniquely Collapses the Polynomial Hierarchy. *SIAM J. Comput.* 25, 4 (1996), 697–708. https://doi.org/10.1137/S0097539794268315

[24] Shuichi Hirahara. 2015. Identifying an Honest $\mathsf{EXP}^{\mathsf{NP}}$ Oracle Among Many. In *CCC (LIPIcs, Vol. 33)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 244–263. https://doi.org/10.4230/LIPIcs.CCC.2015.244

[25] Shuichi Hirahara, Zhenjian Lu, and Hanlin Ren. 2023. Bounded Relativization. In *CCC (LIPIcs, Vol. 264)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 6:1–6:45. https://doi.org/10.4230/LIPIcs.CCC.2023.6

[26] Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. 2018. The Power of Natural Properties as Oracles. In *CCC (LIPIcs, Vol. 102)*. 7:1–7:20. https://doi.org/10.4230/LIPIcs.CCC.2018.7

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

[28] Russell Impagliazzo and Avi Wigderson. 1997. P = BPP if E Requires Exponential Circuits: Derandomizing the XOR Lemma. In *STOC*. ACM, 220–229. https://doi.org/10.1145/258533.258590

[29] Emil Jeřábek. 2004. Dual weak pigeonhole principle, Boolean complexity, and derandomization. *Ann. Pure Appl. Log.* 129, 1-3 (2004), 1–37. https://doi.org/10.1016/j.apal.2003.12.003

[30] Valentine Kabanets and Jin-Yi Cai. 2000. Circuit minimization problem. In *STOC*. 73–79. https://doi.org/10.1145/335305.335314

[31] Ravi Kannan. 1982. Circuit-Size Lower Bounds and Non-Reducibility to Sparse Sets. *Inf. Control.* 55, 1-3 (1982), 40–56. https://doi.org/10.1016/S0019-9958(82)90382-5

[32] Richard M. Karp and Richard J. Lipton. 1980. Some Connections between Nonuniform and Uniform Complexity Classes. In *STOC*. 302–309. https://doi.org/10.1145/800141.804678

[33] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos Papadimitriou. 2021. Total functions in the polynomial hierarchy. In *ITCS (LIPIcs, Vol. 185)*. 44:1–44:18. https://doi.org/10.4230/LIPIcs.ITCS.2021.44

[34] Adam R. Klivans and Dieter van Melkebeek. 2002. Graph Nonisomorphism Has Subexponential Size Proofs Unless the Polynomial-Time Hierarchy Collapses. *SIAM J. Comput.* 31, 5 (2002), 1501–1526. https://doi.org/10.1137/S0097539700389652

[35] Johannes Köbler and Osamu Watanabe. 1998. New Collapse Consequences of NP Having Small Circuits. *SIAM J. Comput.* 28, 1 (1998), 311–324. https://doi.org/10.1137/S0097539795296206

[36] Oliver Korten. 2021. The Hardest Explicit Construction. In *FOCS*. IEEE, 433–444. https://doi.org/10.1109/FOCS52979.2021.00051

[37] Jan Krajíček. 2001. Tautologies from pseudo-random generators. *Bull. Symb. Log.* 7, 2 (2001), 197–212. https://doi.org/10.2307/2687774

[38] Xin Li. 2023. Two Source Extractors for Asymptotically Optimal Entropy, and (Many) More. In *FOCS*. IEEE, 1271–1281. https://doi.org/10.1109/FOCS57990.2023.00075

[39] Zeyong Li. 2023. Symmetric Exponential Time Requires Near-Maximum Circuit Size: Simplified, Truly Uniform. *CoRR* (2023). https://doi.org/10.48550/arXiv.2310.17762

[40] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. 1992. Algebraic Methods for Interactive Proof Systems. *Journal of the ACM* 39, 4 (1992), 859–868. https://doi.org/10.1145/146585.146605

[41] Oleg B Lupanov. 1958. On the synthesis of switching circuits. *Doklady Akademii Nauk SSSR* 119, 1 (1958), 23–26.

[42] Peter Bro Miltersen, N. V. Vinodchandran, and Osamu Watanabe. 1999. Super-Polynomial Versus Half-Exponential Circuit Size in the Exponential Hierarchy. In *COCOON (Lecture Notes in Computer Science, Vol. 1627)*. Springer, 210–220. https://doi.org/10.1007/3-540-48686-0_21

[43] Noam Nisan and Avi Wigderson. 1994. Hardness vs Randomness. *Journal of Computer and System Sciences* 49, 2 (1994), 149–167. https://doi.org/10.1016/S0022-0000(05)80043-1

[44] Hanlin Ren, Rahul Santhanam, and Zhikun Wang. 2022. On the Range Avoidance Problem for Circuits. In *FOCS*. IEEE, 640–650. https://doi.org/10.1109/FOCS54457.2022.00067

[45] Alexander Russell and Ravi Sundaram. 1998. Symmetric Alternation Captures BPP. *Comput. Complex.* 7, 2 (1998), 152–162. https://doi.org/10.1007/s000370050007

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

[47] Alan L. Selman. 1994. A Taxonomy of Complexity Classes of Functions. *J. Comput. Syst. Sci.* 48, 2 (1994), 357–381. https://doi.org/10.1016/S0022-0000(05)80009-1

[48] Claude E. Shannon. 1949. The synthesis of two-terminal switching circuits. *Bell System technical journal* 28, 1 (1949), 59–98. https://doi.org/10.1002/j.1538-7305.1949.tb03624.x

[49] Leslie G. Valiant. 1977. Graph-Theoretic Arguments in Low-Level Complexity. In *MFCS (Lecture Notes in Computer Science, Vol. 53)*. 162–176. https://doi.org/10.1007/3-540-08353-7_135

[50] N. V. Vinodchandran. 2005. A note on the circuit complexity of PP. *Theor. Comput. Sci.* 347, 1-2 (2005), 415–418. https://doi.org/10.1016/j.tcs.2005.07.032

[51] Nikhil Vyas and Ryan Williams. 2023. On Oracles and Algorithmic Methods for Proving Lower Bounds. In *ITCS (LIPIcs, Vol. 251)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 99:1–99:26. https://doi.org/10.4230/LIPIcs.ITCS.2023.99

Received 06-OCT-2023; accepted 2024-02-11