On the Range Avoidance Problem for Circuits

Hanlin Ren
Dept. of Computer Science
University of Oxford
Oxford, UK
hanlin.ren@cs.ox.ac.uk

Rahul Santhanam
Dept. of Computer Science
University of Oxford
Oxford, UK
rahul.santhanam@cs.ox.ac.uk

Zhikun Wang
School of CS&T
Xi'an Jiaotong University
Xi'an, China
nocrizwang@gmail.com

***Abstract*—We consider the *range avoidance* problem (called Avoid): given the description of a circuit with more output gates than input gates, find a string that is not in the range of the circuit. This problem is complete for the class APEPP that corresponds to explicit constructions of objects whose existence follows from the probabilistic method (Korten, FOCS 2021).**

Motivated by applications in explicit constructions and complexity theory, we initiate the study of the range avoidance problem for weak circuit classes, and obtain the following results:

1) **Generalising Williams's connections between circuit-analysis algorithms and circuit lower bounds (J. ACM 2014), we present a framework for solving $\mathscr{C}$-Avoid in $\mathrm{FP}^{\mathrm{NP}}$ using circuit-analysis *data structures* for $\mathscr{C}$, for "typical" multi-output circuit classes $\mathscr{C}$. As an application, we present a non-trivial $\mathrm{FP}^{\mathrm{NP}}$ range avoidance algorithm for De Morgan formulas.**
   An important technical ingredient is a construction of *rectangular* PCPs of proximity, building on the rectangular PCPs by Bhangale, Harsha, Paradise, and Tal (FOCS 2020).
2) **Using the above framework, we show that circuit lower bounds for E^{NP} are equivalent to circuit-analysis algorithms *with E^{NP} preprocessing*. This is the first equivalence result regarding circuit lower bounds for E^{NP}. Our equivalences have the additional advantages that they work in both infinitely-often and almost-everywhere settings, and that they also hold for larger (e.g., subexponential) size bounds.**
3) **Complementing the above results, we show that in some settings, solving $\mathscr{C}$-Avoid would imply breakthrough lower bounds, even for very weak circuit classes $\mathscr{C}$. In particular, an algorithm for AC^0-Avoid with polynomial stretch implies lower bounds against NC^1, and an algorithm for NC^0_4-Avoid with very small stretch implies lower bounds against NC^1 and branching programs.**
4) **We show that Avoid is in FNP if and only if there is a propositional proof system that breaks every non-uniform proof complexity generator. This result connects the study of range avoidance with fundamental questions in proof complexity.**

***Index Terms*—computational complexity; circuit complexity; pseudorandomness**

Note: Various proofs are either abridged or omitted in this extended abstract; proofs of all stated results can be found in the full version: https://eccc.weizmann.ac.il/report/2022/048/

I. Introduction

In this paper, we consider the *range avoidance* problem, denoted as Avoid:

Problem I.1 (Range Avoidance Problem, Avoid)**.** Given the description of a circuit $C : \{0,1\}^n \to \{0,1\}^\ell$, where $\ell > n$, output any string $y \in \{0,1\}^\ell$ that is not in the range of C. That is, for every $x \in \{0,1\}^n$, $C(x) \neq y$.

The *dual weak pigeonhole principle* [33], [37] states that if N pigeons are placed into M holes where $M \geq 2N$, then there is an empty hole. This principle implies that Avoid is a *total* problem, i.e., it always has a valid solution. But what is the computational complexity of finding this solution?

This problem was studied in [34] under the name 1-Empty.[1] In their paper, the motivation was to identify natural total search problems in the polynomial hierarchy, in particular $\mathsf{TF}\Sigma_2$. Indeed, it is easy to see that Avoid belongs to (the function version of) Σ_2, but it is unknown whether it is in FNP. We may try to solve Avoid by guessing a string $y \in \{0,1\}^\ell$ as an answer, but it seems unclear how to verify that y is not in the range of C without using a universal quantifier. Then, [34] defines a natural subclass of $\mathsf{TF}\Sigma_2$ called APEPP (Abundant Polynomial Empty Pigeonhole Principle), which is the class of total search problems polynomial-time reducible to Avoid.

The avoidance problem is also motivated by proof complexity and bounded arithmetic, in particular, the proof complexity of the dual weak pigeonhole principle. Jeřábek [33] defined a theory of bounded arithmetic called APC^1 for formalising probabilistic reasoning by incorporating the dual weak pigeonhole principle as an axiom. Krajíček [37], [38] connects the dual weak pigeonhole principle to hard candidates for Extended Frege and stronger proof systems, as well as to the provability of circuit lower bounds.

A. Explicit Constructions

Another motivation for studying Avoid, which is more relevant to the current paper, is its connection to *explicit construction* problems. The existence of many combinatorial objects, such as Ramsey graphs, expander graphs, and rigid matrices, are proved by the *probabilistic method* [21], [46], [54]. These probabilistic arguments are able to show that a random object has the desired property with non-zero probability, but are usually unable to explicitly construct such an object. Indeed, many explicit construction problems, such as deterministically constructing Ramsey graphs and rigid matrices, are long-standing open questions.

[1] "Empty" stands for "empty pigeonhole principle"; the constant 1 means that the input circuit has stretch at least one bit, i.e., $\ell \geq n + 1$.

Arguably, for complexity theorists, the most interesting explicit construction problems are *circuit lower bounds*. It is well-known that almost every Boolean function on n input bits requires circuits of size $\Omega(2^n/n)$ to compute [52], but so far the best lower bound for any explicit function against general circuits is only $5n$ [32] or $3.1n$ [22], [41], depending on the circuit model.

It was pointed out by Korten [36] that the range avoidance problem nicely captures the complexity of explicit constructions. Most explicit construction problems fall into the category where a "non-random" object of length n can be compressed into strictly less than n bits, and there is an efficient decompression algorithm; see [36, Section 3]. In this case, the problem of constructing a "random" object lies in APEPP, as it suffices to find an object outside the range of the decompression algorithm.

> **Example I.2.** Consider, for example, the problem of proving circuit lower bounds. We want to find a function $f : \{0,1\}^n \to \{0,1\}$ that cannot be computed by circuits of size $s := 2^{n/2}$ (say).
>
> Let $\mathsf{TT} : \{0,1\}^{O(s \log s)} \to \{0,1\}^{2^n}$ be the circuit that takes as input the description of a size-s circuit, and outputs the truth table of this circuit. (Here TT denotes *truth table*.) If we could solve Avoid on the particular instance TT, then we could find a truth table $tt \in \{0,1\}^{2^n}$ without size-s circuits, therefore proving a circuit lower bound. More precisely, solving Avoid for TT in polynomial time is equivalent to proving a circuit lower bound for E, while solving Avoid for TT in $\mathsf{FP}^{\mathsf{NP}}$ is equivalent to proving a circuit lower bound for E^{NP}.

Korten was interested in the *structure* of APEPP. Indeed, one of the main results in [36] was that constructing a hard truth table is complete for APEPP under P^{NP} reductions. In other words, if we could construct a hard truth table in $\mathsf{FP}^{\mathsf{NP}}$, then $\mathsf{APEPP} \subseteq \mathsf{FP}^{\mathsf{NP}}$, which means that any "typical" explicit construction problem can be solved in $\mathsf{FP}^{\mathsf{NP}}$.

We are especially interested in the "easier" regime of APEPP. Let $\mathscr{C}$ be a (multi-output) circuit class, we assume $\mathscr{C}$ is also associated with a stretch function $\ell(n) > n$ such that every circuit in $\mathscr{C}$ maps n bits to $\ell(n)$ bits. Let $\mathscr{C}$-Avoid be the range avoidance problem for $\mathscr{C}$ circuits, we ask:

Question I.3. *For which circuit classes $\mathscr{C}$ are the $\mathscr{C}$-*Avoid *problems easy?*

To the best of our knowledge, we are the first to consider the $\mathscr{C}$-Avoid problem for restricted circuit classes $\mathscr{C}$. We think this is an interesting research direction for the following reasons:

- For an explicit construction problem Π, one could identify the weakest circuit class $\mathscr{C}$ such that Π reduces to $\mathscr{C}$-Avoid. Therefore, progress on Question I.3 implies progress on explicit constructions.
- This consideration reveals some new phenomena. For example, even for very weak circuit classes such as $\mathscr{C} = \mathsf{NC}^0$, solving the $\mathscr{C}$-Avoid problem (with stretch $\ell(n) := n + n^{o(1)}$) implies strong lower bounds that are currently out of reach! (See Theorem II.11.)

There are many interpretations of the word "easy", but for now, let us think of it as "*provably* in $\mathsf{FP}^{\mathsf{NP}}$". Note that if strong enough circuit lower bounds hold, then by [36], every explicit construction problem is in $\mathsf{FP}^{\mathsf{NP}}$. Therefore we insist that the correctness of the avoidance algorithm *does not rely on unproven assumptions*.

Why $\mathsf{FP}^{\mathsf{NP}}$? There are at least two reasons to study $\mathsf{FP}^{\mathsf{NP}}$ algorithms for Avoid.

- First, $\mathsf{FP}^{\mathsf{NP}}$ is one of the most powerful notions of algorithms that we do not know how to solve Avoid. This is similar to the situation that E^{NP} is one of the biggest complexity classes for which we have no super-polynomial size circuit lower bounds.[2]

  Actually, Avoid can be solved in $\mathsf{FP}^{\mathsf{NP}}$ if and only if E^{NP} cannot be computed by $2^{o(n)}$-size circuits [36]. Therefore, if we could solve $\mathscr{C}$-Avoid in $\mathsf{FP}^{\mathsf{NP}}$ unconditionally for more and more powerful classes $\mathscr{C}$, we could make progress towards the notorious open problem of proving circuit lower bounds against the class E^{NP}.
- Second, APEPP has a very nice structure under $\mathsf{FP}^{\mathsf{NP}}$ reductions. Many reductions among problems in APEPP are only known to be computable in P^{NP}, such as the reductions among Avoid for different stretch functions [34], [36] and the APEPP-completeness of finding a hard truth table [36]. In this paper, we will see more examples where we can make progress when considering $\mathsf{FP}^{\mathsf{NP}}$ algorithms.

  For comparison, the structure of APEPP under (say) polynomial-time reductions is less clear. For example, it is not known if finding a hard truth table is APEPP-complete under polynomial-time reductions. It is also open whether Avoid $\in \mathsf{FP}$ or its negation is implied by any "plausible" assumption in complexity theory (or cryptography).

Note that Avoid can be solved in $\mathsf{FZPP}^{\mathsf{NP}}$: simply guess a random string $y \in \{0,1\}^\ell$ and use the NP oracle to verify if y is not in the range of the input circuit. Therefore, whether Avoid is in $\mathsf{FP}^{\mathsf{NP}}$ is essentially a derandomisation question.

B. The Algorithmic Method

Building on his previous work [57], Williams [58] famously proved that $\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0$, the first non-uniform lower bound against the notorious circuit class ACC^0. Interestingly, the lower bound is proved by an *algorithmic* method: Williams designed a "non-trivial" satisfiability algorithm for ACC^0 circuits, and then showed that such algorithms imply lower bounds against ACC^0. The only property of ACC^0 that Williams uses is that ACC^0-SAT has a non-trivial algorithm; the algorithm-to-lower-bound connection works for any circuit class satisfying some mild technical conditions.

There have been a long line of subsequent developments of the Algorithmic Method [9], [13]–[19], [42], [50], [55], [56],

[2] Slightly higher classes, such as $\mathsf{ZPE}^{\mathsf{NP}}$ and MA-E (the exponential-time analogue of MA), are known to require super-polynomial size circuits [11], [35].

[59]–[61]. A recent highlight is the following result proved in [16]:

Theorem I.4 ([16], Informal)**.** *There is a language in* E^{NP} *that does not have sub-exponential size* ACC^0 *circuits on almost every input length.*

Thinking of circuit lower bounds as explicit construction problems, [16] gave an $\mathsf{FP}^{\mathsf{NP}}$-explicit construction of hard truth tables against sub-exponential size ACC^0 circuits. We can even formulate Theorem I.4 in the language of circuit range avoidance:

Theorem I.5 (Theorem I.4, Reformulated)**.** *Let* $s(n) := 2^{n^{o(1)}}$*,* $\mathsf{TT}_{\mathsf{ACC}^0} : \{0,1\}^{O(s(n)\log s(n))} \to \{0,1\}^{2^n}$ *be the circuit that takes as input the description of a size-*$s(n)$ ACC^0 *circuit, and outputs its truth table. Then, there is an* $\mathsf{FP}^{\mathsf{NP}}$ *algorithm for solving* Avoid *on the instance* $\mathsf{TT}_{\mathsf{ACC}^0}$*.*

Recently, the Algorithmic Method has found applications to another problem: constructing rigid matrices! Alman and Chen [2] showed how to construct rigid matrices in $\mathsf{FP}^{\mathsf{NP}}$ with parameters much better than previously known constructions; their results were later improved by [10], [15], [16], [27]. The key insight in [2] is to treat low-rank matrices as a special type of *circuit class*, and the task of constructing rigid matrices reduces to proving average-case circuit lower bounds against this class.

Given the success of the Algorithmic Method, it is natural to ask the following question:

Question I.6. *Under which conditions can the Algorithmic Method be used to solve general explicit construction problems, such as* Avoid*?*

II. Our Results

In this work, we present new algorithmic and structural results for the range avoidance problem. Along the way, we obtain a version of the Algorithmic Method that *characterises* circuit lower bounds for E^{NP}. We begin by presenting a high-level overview of our results, and then we discuss each result in detail.

a) An algorithmic method for range avoidance: In our first main result, we present a version of the Algorithmic Method for solving Avoid. Let $\mathscr{C}$ be a multi-output circuit class under some closure properties. We show that if the *Hamming weight estimation* problem for $\mathscr{C}$ has a non-trivial data structure, then the range avoidance problem for $\mathscr{C}$ circuits can be solved in $\mathsf{FP}^{\mathsf{NP}}$. Here, a data structure for Hamming weight estimation for $\mathscr{C}$ has a preprocessing phase and a query phase: During the preprocessing phase, it is given the description of a $\mathscr{C}$ circuit C and produces a data structure DS in $\mathsf{FP}^{\mathsf{NP}}$, i.e., polynomial time with access to an NP oracle. Each query consists of an input x, and we want to estimate deterministically, with the aid of DS, the Hamming weight of $C(x)$ in time faster than brute force.

For comparison, the Algorithmic Method for proving circuit lower bounds works as follows. Let $\mathscr{C}$ be a (single-output) circuit class under some closure properties. If there is a non-trivial CAPP algorithm for $\mathscr{C}$ circuits,[3] then $\mathsf{E}^{\mathsf{NP}} \not\subseteq \mathscr{C}$. A CAPP algorithm for $\mathscr{C}$ can be seen as a Hamming weight estimation algorithm (without preprocessing) for $\mathsf{TT}_{\mathscr{C}}$, where $\mathsf{TT}_{\mathscr{C}}$ is the multi-output circuit which takes the description of a $\mathscr{C}$ circuit as input and outputs its truth table. Thus, our result is a generalisation of the Algorithmic Method.

As an application of this result, we show *unconditional* $\mathsf{FP}^{\mathsf{NP}}$ algorithms for the range avoidance problem for De Morgan formulas. In particular, let $s = s(n)$ be a polynomial, C be a function that maps n input bits to $n^{\omega(\sqrt{s}\log s)}$ output bits where each output bit is computed by a De Morgan formula of size $s(n)$, then our algorithm finds a non-output of C in $\mathsf{FP}^{\mathsf{NP}}$.

b) Circuit lower bounds for E^{NP} *are data structures:* An easy corollary of our main result is that in the Algorithmic Method, CAPP data structures for $\mathscr{C}$, *even with* E^{NP} *preprocessing*, implies $\mathsf{E}^{\mathsf{NP}} \not\subseteq \mathscr{C}$.[4] In our second main result, we show that this is the "complete" Algorithmic Method for proving lower bounds for E^{NP}! For strong enough circuit classes $\mathscr{C}$ (such as TC^0, NC^1, or $\mathsf{P}_{/\mathrm{poly}}$), $\mathsf{E}^{\mathsf{NP}} \not\subseteq \mathscr{C}$ if and only if it can be proved by a non-trivial CAPP algorithm with E^{NP} preprocessing.

Of course, it would be nicer to obtain a similar equivalence for weaker circuits classes $\mathscr{C}$, namely those that are unable to compute $\mathsf{MAJORITY}$ and perform hardness amplification [25], [26], [51]. We achieve this by considering *strong average-case* circuit lower bounds and CAPP algorithms with inverse-circuit-size error[5]: E^{NP} cannot be $(1/2 + 1/\mathrm{poly}(n))$-approximated by $\mathscr{C}$ if and only if there is a non-trivial CAPP algorithm for $\mathscr{C}$ with inverse-circuit-size error and E^{NP} preprocessing.

c) Range avoidance for smaller stretch implies breakthrough lower bounds: Note that we ignored the role of the stretch function in the above discussion. Actually, even with "perfect" Hamming weight estimation data structures, our main result only solves the range avoidance problem for circuits that maps n input bits to $n^{1+\epsilon}$ output bits, for every constant $\epsilon > 0$.

How does the stretch function affect the difficulty of (provably) solving Avoid? We show that solving the range avoidance problem for circuits with small enough stretch implies breakthrough lower bounds:

- Solving AC^0-Avoid with quasi-polynomial stretch

[3] CAPP stands for "circuit acceptance probability problem". A CAPP algorithm for $\mathscr{C}$ is a deterministic algorithm that takes as input a $\mathscr{C}$ circuit C, and outputs an estimation of $\Pr_{x \leftarrow \{0,1\}^n}[C(x) = 1]$ within some additive error.

[4] Here, a CAPP data structure for $\mathscr{C}$ with E^{NP} preprocessing is the following data structure: In the preprocessing phase, we are given the input 1^n, and need to produce a data structure DS in $2^{O(n)}$ time with an NP oracle. In the query phase, we are given a $\mathscr{C}$ circuit C on n input bits, and need to estimate $\Pr_{x \leftarrow \{0,1\}^n}[C(x) = 1]$ in non-trivial time, with the aid of DS. Equivalently, it is a Hamming weight estimation data structure for $\mathsf{TT}_{\mathscr{C}}$.

[5] A CAPP algorithm with inverse-circuit-size error approximates the probability that C accepts a random input within additive error $1/|C|$, where $|C|$ is the size of C.

(for a small enough quasi-polynomial) implies super-polynomial lower bounds against NC^1.
- Even for the very simple circuit class NC^0_4 (where every output bit only depends on *four* input bits), solving NC^0_4-Avoid with stretch $\ell(n) := n+n^{o(1)}$ implies exponential lower bounds against NC^1 and $\oplus\mathsf{L}_{/\mathrm{poly}}$ (parity branching programs).

We interpret these results as an indication that it may be difficult to generalise Korten's results [36] to restricted circuit classes such as ACC^0 or formulas. Korten showed that Avoid reduces to finding a truth table with large circuit complexity (in P^{NP}); if Formula-Avoid (actually, NC^0_4-Avoid) reduces to finding a truth table with large formula complexity, then formula lower bounds would imply lower bounds for an even stronger model, namely parity branching programs.

d) Complexity of range avoidance below $\mathsf{FP}^{\mathsf{NP}}$*:* We also study the complexity of range avoidance w.r.t. algorithms less powerful than $\mathsf{FP}^{\mathsf{NP}}$. As mentioned before, it is unknown whether Avoid $\in \mathsf{FNP}$, Avoid $\in \mathsf{FP}$, or their negations are implied by any plausible assumptions. As far as we know, we do not even have a good idea of what the "ground truth" should be! (For comparison, by Korten's result [36], if one believes circuit lower bounds, one should also believe Avoid $\in \mathsf{FP}^{\mathsf{NP}}$.)

It turns out that the statements "Avoid $\in \mathsf{FNP}$" and "Avoid $\in \mathsf{FP}$" can be characterised by classical notions in complexity theory. In particular, we connect the existence of FNP algorithms for Avoid with the security of *proof complexity generators*, and connect the existence of FP algorithms for Avoid with a version of *time hierarchy theorem* with advice.

We now discuss our results in more detail.

A. An Algorithmic Method for Range Avoidance

Our first main result is a version of the Algorithmic Method for solving the range avoidance problem. Here, instead of non-trivial circuit-analysis algorithms, we consider *data structures with* P^{NP} *preprocessing and non-trivial query time*.

For a binary string s, let $\delta(s)$ denote the *relative Hamming weight* of s, i.e., the fraction of bits in s that is equal to 1. For a multi-output circuit class $\mathscr{C}$, let $\mathscr{C}$-HammingHit be the following data structure problem:

(Preprocessing) Given the description of a $\mathscr{C}$ circuit $C : \{0,1\}^n \to \{0,1\}^\ell$, preprocess C in polynomial time with access to an NP oracle (that is, in P^{NP}), and produce a data structure $\mathsf{DS} \in \{0,1\}^{\mathrm{poly}(\ell)}$.

(Query) Given a string $x \in \{0,1\}^n$, distinguish between the case that $\delta(C(x)) = 1$ and the case that $\delta(C(x)) < 0.01$ in deterministic $\ell/\log^{\omega(1)}\ell$ time with oracle access to DS.

We show that if we want to solve $\mathscr{C}$-Avoid in $\mathsf{FP}^{\mathsf{NP}}$, it suffices to design a data structure for the $\mathscr{C}'$-HammingHit problem, where $\mathscr{C}' := \mathsf{NC}^0 \circ \mathscr{C}$. Here, $\mathscr{C}' = \mathsf{NC}^0 \circ \mathscr{C}$ means that each output gate of a $\mathscr{C}'$ circuit is a function over a constant number of $\mathscr{C}$ circuits.

Theorem II.1 (Main Result 1, Informal)**.** *Let $\mathscr{C}$ be a (multi-output) circuit class, and $\mathscr{C}' := \mathsf{NC}^0 \circ \mathscr{C}$. Suppose there is a data structure for the $\mathscr{C}'$-HammingHit problem with P^{NP} preprocessing and non-trivial (i.e., $\ell/\log^{\omega(1)}\ell$) query time. Then $\mathscr{C}$-Avoid is in $\mathsf{FP}^{\mathsf{NP}}$.*

Remark II.2. The power of P^{NP} preprocessing for data structures remains to be investigated. Some examples in the literature where P^{NP} preprocessing seems helpful are:

- The currently fastest data structure for the Online Matrix-Vector Multiplication problem achieves *amortised* $n^2/2^{\Omega(\sqrt{\log n})}$ query time [40]. It is implicit in their paper that if we allow P^{NP} preprocessing, then the query algorithm can be made worst-case.
- The optimal expander decomposition can be computed in P^{NP} (see e.g., [47]). However, in this case, there are also very good expander decomposition algorithms in deterministic polynomial time [24].

a) A note on the stretch functions: It is clear that a non-trivial data structure for HammingHit is possible only when $n < \ell/\log^{\omega(1)}\ell$. In the above informal statements, we omitted the stretch of $\mathscr{C}$ and $\mathscr{C}'$ circuits for simplicity. Actually, even assuming the best possible HammingHit data structures, Theorem II.1 could only solve the range avoidance problem for circuits with stretch $\ell(n) = n^{1+\epsilon}$. We refer to the full version for the precise statement of Theorem II.1.

b) Application: Range avoidance for De Morgan formulas: We apply our connection to show a non-trivial $\mathsf{FP}^{\mathsf{NP}}$ algorithm that solves the range avoidance problem for De Morgan formulas.

Theorem II.3. *Let $s(n)$ be a polynomial, $\mathscr{C}$ be the class of multi-output functions where each output bit is computed by a size-$s(n)$ De Morgan formula, and the number of output bits is at least $\ell := n^{\omega(\sqrt{s}\log s)}$. Then there is an $\mathsf{FP}^{\mathsf{NP}}$ algorithm for $\mathscr{C}$-Avoid.*

Roughly speaking, Theorem II.3 is proved by the "quantum method" [53] for De Morgan formulas: every function computed by a De Morgan formula of size s has approximate degree $O(\sqrt{s}\log s)$. By Theorem II.1, it suffices to solve the HammingHit problem for De Morgan formulas. In the preprocessing phase, we compute the low-degree polynomials that approximate each output gate and add them into a single polynomial p of degree $O(\sqrt{s}\log s)$. In the query phase, we use $n^{O(\sqrt{s}\log s)} \ll \ell$ time to evaluate p, which gives a good estimation of the relative Hamming weight.

B. Equivalences between E^{NP} Circuit Lower Bounds and Non-trivial Derandomisation with Preprocessing

In our second set of results, we show that circuit lower bounds for E^{NP} and CAPP algorithms with E^{NP} preprocessing are equivalent. From Theorem II.1 we know that a non-trivial $\mathsf{GapUNSAT}$ algorithm for $\mathscr{C}$, even *with* E^{NP} *preprocessing*, would imply $\mathsf{E}^{\mathsf{NP}} \not\subseteq \mathscr{C}$. We show that for powerful enough circuit classes $\mathscr{C}$ (e.g., TC^0, NC^1, or $\mathsf{P}_{/\mathrm{poly}}$), the converse is also true:

Theorem II.4 (Main Result 2.1, Informal)**.** *Let $\mathscr{C} \in \{\mathsf{TC}^0, \mathsf{NC}^1, \mathsf{P}_{/\mathrm{poly}}\}$. The following are equivalent:*

- E^{NP} *cannot be computed by polynomial-size $\mathscr{C}$ circuits on almost every input length.*
- *There is a non-trivial* $\mathsf{GapUNSAT}$ *algorithm for $\mathscr{C}$ circuits with* E^{NP} *preprocessing.*

For circuit classes $\mathscr{C}$ that are less powerful (i.e., that might not be able to efficiently compute $\mathsf{MAJORITY}$), we show that *strong average-case* circuit lower bounds against $\mathscr{C}$ and CAPP algorithms for $\mathscr{C}$ with inverse-circuit-size error are equivalent:

Theorem II.5 (Main Results 2.2, Informal)**.** *Let $\mathscr{C}$ be a "weak" circuit class under some mild closure properties. The following are equivalent:*

- E^{NP} *cannot be* $(1/2 + 1/\mathrm{poly}(n))$*-approximated by $\mathscr{C}$ circuits on almost every input length.*
- *There is a non-trivial* CAPP *algorithm for $\mathscr{C}$ circuits with* E^{NP} *preprocessing and inverse-circuit-size error.*

Actually, we can show equivalences among a lot of notions, including strong average-case lower bounds for E^{NP}, non-trivial CAPP algorithms with E^{NP} preprocessing, subexponential-time CAPP algorithms with E^{NP} preprocessing, and E^{NP}-computable PRGs. For details please refer to the full version.

It is remarkable that although these equivalences do not refer to AVOID, the most natural way of deriving them seems to go through it. In particular, in the range avoidance problem, it is impossible to estimate the Hamming weight of an ℓ-output circuit in $o(\ell)$ time without preprocessing it, so the preprocessing phase appears naturally. It turns out that adding this preprocessing phase to the (standard) Algorithmic Method makes it an equivalence![6] Our results are also a rare instance where a data structure problem ($\mathsf{HammingHit}$ or CAPP with preprocessing) plays a crucial role in a fundamental problem in complexity theory.

One advantage of our equivalence is that it also holds for larger size bounds and the case of infinitely-often lower bounds:

Theorem II.6 (Informal)**.** *Let $\mathscr{C}$ be a "weak" circuit class under some mild closure properties. The following are equivalent:*

- E^{NP} *cannot be* $(1/2 + 1/2^{n^{o(1)}})$*-approximated by $\mathscr{C}$ circuits of size* $2^{n^{o(1)}}$.
- *There is a* CAPP *algorithm for $\mathscr{C}$ circuits of size* $2^{n^{o(1)}}$ *with* $2^{n-n^{\Omega(1)}}$ *query time,* E^{NP} *preprocessing, and inverse-circuit-size error, that works for infinitely many* n.

Remark II.7 (Equivalences between Derandomisation and Lower Bounds).

Equivalences between derandomisation and lower bounds are known in many settings.

- Impagliazzo, Kabanets, and Wigderson [28] showed that $\mathsf{NEXP} \not\subseteq \mathsf{P}_{/\mathrm{poly}}$ if and only if there is a non-deterministic subexponential-time algorithm for CAPP with $n^{o(1)}$ bits of advice and error $1/6$ that works infinitely often.
- Korten's result [36] can also be interpreted as an equivalence between derandomisation and lower bounds: A full derandomisation of the trivial $\mathsf{FZPP}^{\mathsf{NP}}$ algorithm for AVOID is equivalent to both $\mathsf{E}^{\mathsf{NP}} \not\subseteq \mathsf{SIZE}[2^{0.1n}]$ and $\mathsf{E}^{\mathsf{NP}} \not\subseteq \mathsf{SIZE}[2^n/3n]$.
- Equivalences between derandomisation and *uniform* lower bounds are also known. Impagliazzo and Wigderson [29] showed that $\mathsf{EXP} \neq \mathsf{BPP}$ is equivalent to an infinitely-often, subexponential time derandomisation of BPP on average ($\mathsf{BPP} \subseteq \text{i.o.-}\mathsf{heurDTIME}[2^{n^{o(1)}}]$). Williams [59] showed that $\mathsf{NEXP} \neq \mathsf{BPP}$ is equivalent to an infinitely-often, subexponential time nondeterministic derandomisation of BPP on average, with $n^{o(1)}$ bits of advice ($\mathsf{BPP} \subseteq \text{i.o.-}\mathsf{heurNTIME}[2^{n^{o(1)}}]_{/n^{o(1)}}$).

In our opinion, compared to the above equivalences, our results have the following features that make them particularly attractive:

- First, they work in both infinitely-often and almost-everywhere settings; in contrast, [29] and [59] only hold for infinitely-often lower bounds.
- Second, they scale better with large circuit size bounds (such as $2^{n^{o(1)}}$); no similar equivalences to [28] for $\mathsf{NEXP} \not\subseteq \mathsf{SIZE}[2^{n^{o(1)}}]$ or to [29] for $\mathsf{EXP} \not\subseteq \mathsf{BPTIME}[2^{n^{o(1)}}]$ are known.
- Third, they are also true for weaker circuit classes such as formulas or ACC^0 circuits; in contrast, the arguments in [36] does not seem to yield any characterisation of the lower bound $\mathsf{E}^{\mathsf{NP}} \not\subseteq \mathsf{Formula}[2^{0.1n}]$.
- Finally, our equivalences include both subexponential-time derandomisation and non-trivial derandomisation; none of the equivalences above are known to include non-trivial derandomisation.

An interesting corollary of Theorems II.4 and II.5 is the following "speed-up" result for derandomisation with E^{NP} preprocessing:

Corollary II.8 (Informal)**.** *The following are true:*

- *If there is a non-trivial* $\mathsf{GapUNSAT}$ *algorithm for* TC^0 *circuits with* E^{NP} *preprocessing, then there is a subexponential-time* CAPP *algorithm for* TC^0 *circuits with* E^{NP} *preprocessing.*
- *Let $\mathscr{C}$ be a "weak" circuit class under some mild closure properties. If there is a non-trivial* CAPP *algorithm for $\mathscr{C}$ circuits with* E^{NP} *preprocessing and inverse-circuit-size error, then there is a subexponential-time* CAPP *algorithm for $\mathscr{C}$ circuits with* E^{NP} *preprocessing and inverse-circuit-size error.*

Remark II.9 (Comparison with Other Speed-Ups in Complexity Theory).

Williams [57] showed that if CAPP has a nondeterministic algorithm with non-trivial running time, then CAPP also has a nondeterministic subexponential time algorithm. One caveat of this result is that the speed-up algorithm is only infinitely-often correct, and requires n^{ϵ} bits of advice. Therefore, the speed-up algorithm does not imply the non-trivial algorithm. In contrast, in Corollary II.8, the speed-up algorithms always imply the non-trivial algorithms.

Oliveira and Santhanam [44] showed a similar speed-up

[6] Note that the *proof* that adding the E^{NP} preprocessing phase still gives us lower bounds is highly non-trivial and requires some new ideas. See Section III for an overview.

result in learning theory: a typical circuit class is "non-trivially learnable" if and only if it is learnable in sub-exponential time. Their result is proved using the connection between natural proofs and learning [12], [49], while our result is a strengthening of the Algorithmic Method.

C. The Role of the Stretch Function

The input to AVOID is a circuit $C : \{0,1\}^n \to \{0,1\}^{\ell(n)}$. How does the stretch function $\ell(n)$ affect the complexity of AVOID? Clearly, the larger ℓ is, the easier it is to solve AVOID. But is there any *qualitative* difference between (say) $\ell(n) = n+1$, $\ell(n) = 2n$, and $\ell(n) = n^{100}$?

Korten [36] suggests that the answer might be *no*, at least w.r.t. $\mathsf{FP}^{\mathsf{NP}}$ algorithms. The range avoidance problem for stretch $n+1$ reduces to the problem of finding a truth table without $2^{0.001n}$-size circuits in P^{NP}; the latter problem trivially reduces to the range avoidance problem for some circuit of stretch n^{100}.

However, when we consider $\mathscr{C}$-AVOID for restricted circuit classes $\mathscr{C}$, we show that the answer is *yes*! We think this is an interesting phenomenon revealed by the investigation of $\mathscr{C}$-AVOID for very weak classes $\mathscr{C}$.

Theorem II.10 (Informal)**.** *Suppose that for a small enough quasi-polynomial $\ell(\cdot)$, AC^0-AVOID with stretch ℓ is solvable in $\mathsf{FP}^{\mathsf{NP}}$. Then $\mathsf{E}^{\mathsf{NP}} \not\subseteq \mathsf{NC}^1$.*

Theorem II.11 (Informal)**.** *Suppose that NC^0_4-AVOID with stretch $\ell := n + n^{o(1)}$ is solvable in $\mathsf{FP}^{\mathsf{NP}}$. Then E^{NP} does not have subexponential-size formulas and parity branching programs.*

Roughly speaking, Theorem II.10 is proved by the subexponential-size simulation of NC^1 circuits by AC^0 circuits [43], and Theorem II.11 is proved by the randomised encoding techniques of [3], [30], [31].

Our results imply that if Korten's result can be generalised to formula complexity (i.e., Formula-AVOID reduces to finding a hard truth table w.r.t. formula complexity in P^{NP}), then formula lower bounds imply parity branching program lower bounds! This raises several interesting open questions: Is there any fundamental reason that Korten's techniques do not work for formula complexity? What is the difference between circuits and formulas that plays a crucial role here? Are these results connected to the open question of constructing randomised encodings for polynomial-size circuits [3]? We leave these questions as interesting research directions.

D. Is AVOID *in* FNP *or* FP*?*

Finally, we turn to the problem of whether AVOID is in FNP or FP.[7] We note that the results in this section involve some technicalities in the stretch functions (see the full version for details), but we ignore this issue in the informal overview. We show that:

Theorem II.12 (Informal)**.** *The following are true:*

- *There is an FNP algorithm for AVOID if and only if there is a propositional proof system that breaks every non-uniform proof complexity generator.*
- *There is an FP algorithm for (a sparse version of) AVOID if and only if a version of time hierarchy for E holds with near-maximum advice, i.e.,*

$$\mathsf{E} \not\subseteq \text{i.o.-}\mathsf{DTIME}[2^{n+1}]_{/(2^n - \omega(1))}.$$

a) Background: proof complexity generators: Let $\ell > n$, a circuit $C : \{0,1\}^n \to \{0,1\}^\ell$ is a *proof complexity generator* secure against a propositional proof system, if the proof system cannot efficiently prove *any* string not in the range of C [1]. More formally, for every $y \in \{0,1\}^\ell$, the (properly encoded version of the) statement "$\forall x \in \{0,1\}^n, C(x) \neq y$" does not have proofs of polynomial length, even though the statement is true for most y.

The study of proof complexity generators is partly motivated by the search for explicit tautologies that are hard for strong propositional proof systems such as Extended Frege [38]. It is also motivated by meta-mathematical questions about circuit lower bounds: are strong circuit lower bounds efficiently provable in propositional proof systems such as Frege and Extended Frege? Razborov has made several conjectures supporting the possibility that the truth table generator[8] is secure against Frege [48], while Krajíček has examined the evidence in favour of the truth table generator being hard even for Extended Frege [38]. In other words, it has been hypothesised that EF cannot efficiently prove super-polynomial circuit lower bounds for *any* truth table.

Krajíček [39] has also studied the possibility that some proof complexity generator is secure against *every* proof system. Conversely, we could ask: Is there a proof system that can break *every* proof complexity generator? This question turns out to be characterised by the statement AVOID $\in \mathsf{FNP}$.

b) Discussion: generating K^t-random strings: Along the way, we show that the problem of generating strings with near-maximum time-bounded Kolmogorov complexity (K^t) is complete for (the sparse version of) AVOID under polynomial-time reductions.[9]

Consider the following hypothesis:

Hypothesis II.13. *There is a deterministic polynomial-time algorithm that given $(1^t, 1^n)$, finds a string $x \in \{0,1\}^n$ such that $\mathrm{K}^t(x)$ is large.*

We show that this hypothesis is equivalent to the aforementioned "time-hierarchy" hypothesis. The plausibility of Hypothesis II.13 remains to be investigated.

[7] If the following is true, then we say AVOID $\in \mathsf{FNP}$. There is a nondeterministic algorithm that takes a circuit $C : \{0,1\}^n \to \{0,1\}^\ell$ as input, where $\ell > n$, such that the following holds. (1) The algorithm accepts at least one nondeterministic branch. (2) On every accepting branch, the algorithm outputs a string $y \in \{0,1\}^\ell$ that is not in the range of C successfully.

[8] The truth table generator is the function TT introduced in Example I.2. If TT is hard for a propositional proof system, then this proof system cannot efficiently prove circuit lower bounds of *any* truth table.

[9] Again, we emphasize that this result has some technicalities in the stretch.

Hypothesis II.13 is a natural generalisation of circuit lower bounds. If we replace K^t with circuit complexity, we obtain the statement that truth tables with high circuit complexity can be generated in deterministic polynomial time, which is equivalent to circuit lower bounds for E. Is there any formal connection between Hypothesis II.13 and other hardness assumptions (such as circuit lower bounds or cryptographic assumptions)? We leave this question for future research.

We also define a proof complexity generator based on K^t, and show that the K^t generator is the "hardest" proof complexity generator. A proof system breaks every proof complexity generator if and only if it breaks the K^t generator.

E. A Rectangular PCP of Proximity

A crucial technical ingredient for proving Theorem II.1 is a rectangular PCP of proximity (or rectangular PCPP). Here we introduce the rectangular PCPP and discuss its features in the context of the Algorithmic Method. We refer the reader to Section III for the reason that this notion of PCPP is needed.

PCPs (probabilistically checkable proofs) provide a surprisingly efficient way to verify NP proofs. The PCP theorem [4], [5] states that any NP proof can be converted into a polynomially-longer PCP such that a verifier can check its validity by only reading a constant number of bits (at randomly selected locations).

It is often desirable that a PCP has additional properties. One property is shortness: suppose the original NP instance has length n and the NP proof has length m, then the PCP has length $\tilde{O}(n+m)$. Short PCPs are constructed in [6], [7], [20]. Another property is proximity [6]: instead of being in the language, we only verify that the input is *close* to being in the language. The benefit of proximity is that a *super efficient* verifier is now possible: instead of seeing the whole input, the verifier only probes a few bits of both the input string and the proof. We also want the PCP to have projection queries [9], which means the circuit that maps the PCP randomness to its query indices is a projection (i.e., has the lowest possible circuit complexity). This is useful in the Algorithmic Method.

The final property we consider is rectangularity, recently introduced in [10]. In a perfectly rectangular PCP, the proof is a $\sqrt{m} \times \sqrt{m}$ matrix Π, and the PCP randomness seed is partitioned into two parts: seed.row and seed.col. Each PCP query to the proof matrix is specified by a coordinate (r, c) (where we want to probe $\Pi[r, c]$). Rectangularity means that r only depends on seed.row and c only depends on seed.col. In other words, the queries of a rectangular PCP is generated as follows:

- First, the verifier reads seed.row and produces $(r_1, r_2, \ldots, r_q)$ without seeing seed.col.
- Then, the verifier forgets about seed.row, reads seed.col and produces $(c_1, c_2, \ldots, c_q)$.
- The query locations are $(r_1, c_1), (r_2, c_2), \ldots, (r_q, c_q)$.

We do not know if perfectly rectangular PCPs exist. We can only construct *almost* rectangular PCPs, where the randomness seed also contains a short shared portion seed.shared. The verifier sees both seed.row and seed.shared while generating $(r_1, r_2, \ldots, r_q)$, and sees both seed.col and seed.shared while generating $(c_1, c_2, \ldots, c_q)$.

The motivation for considering rectangular PCPs in [10] was to construct rigid matrices (and improve [2]). In this paper, we show another application of rectangular PCPs (actually, PCPs of proximity): they are a crucial ingredient in our Algorithmic Method for range avoidance!

In this paper, we construct a short and almost rectangular PCP of proximity with projection queries. Here, the input is also a matrix that is queried in a rectangular fashion; see the full version for a precise definition.

Theorem II.14 (Informal)**.** *For every constant $\tau > 0$ and functions $W_{\mathsf{input}}(n), W_{\mathsf{proof}}(n)$ satisfying some technical conditions,* $\mathsf{NTIME}[T(n)]$ *has a τ-almost rectangular PCP of proximity with proof length $T(n) \cdot \mathrm{polylog}(T(n))$, input matrix width $W_{\mathsf{input}}(n)$, and proof matrix width $W_{\mathsf{proof}}(n)$. Here, τ-almost rectangularity means that* $|\text{seed.shared}| \leq \tau \cdot |\text{seed}|$.

Moreover, for fixed seed.shared*, the maps from* seed.row *or* seed.col *to the query indices (i.e., r_i or c_i) are projections, computable in polynomial time given* seed.shared.

Remark II.15 (Comparison with [10]). Our rectangular PCP of proximity differs from the rectangular PCP in [10] in the following ways.

- The biggest difference is that our construction is a PCP of *proximity*. As a result, the input is also treated as a matrix, and its query pattern also has to be rectangular.
- The rectangular PCP in [10] is *smooth*, i.e., every bit in the proof is queried with equal probability. Smoothness is not required in our application, and our rectangular PCPP has no smoothness guarantee.
- Finally, the input matrix size and the proof matrix size in our rectangular PCPP are flexible, while the proof matrix in [10] is $\sqrt{m} \times \sqrt{m}$. It is easy to make the proof matrix size flexible, but more care needs to be taken for the input matrix. (See the full version where we artificially define a bijection called bin_{H^m}.) This is quite important as in our application, we need the input matrix width to be as small as possible!

Remark II.16 (Comparison with [9]). To reduce the circuit complexity "overhead" of the Algorithmic Method, [9] constructed PCPs where the query indices are computable by a projection over seed. To achieve this property, [9] needed to use the PCP in [8]. Unfortunately, this PCP needs $\mathrm{polylog}(n)$ queries; even worse, this property is broken when we use PCP composition to reduce query complexity to $O(1)$.

However, if we allow the queries to depend arbitrarily on a small portion of seed (namely seed.shared), but has to be a projection over the rest of the bits, then this is also achievable using the PCP in [6]. The [6] PCP has the advantage of being almost rectangular. We are also able to compose PCPs now, by simply adding the (very short) randomness of the inner PCP into seed.shared. Thus, the query complexity can be reduced to $O(1)$. It turns out that having such a small portion (i.e., seed.shared) does not hurt the Algorithmic Method at all.

III. Technical Overview

In this section, we present an overview of the proof of Theorem II.1.

It is helpful to review the Algorithmic Method for proving E^{NP} lower bounds. Let $L^{\mathsf{hard}} \in \mathsf{NTIME}[2^n] \setminus \mathsf{NTIME}[o(2^n)]$ be a hard language constructed by the nondeterministic time hierarchy theorem [62]. Let V be the PCP verifier of [6]; here V is an oracle circuit $V^{(-)} : \{0,1\}^r \to \{0,1\}$. This oracle circuit takes PCP randomness as input (so the input length is $r = n + O(\log n)$), and receives the PCP proof as the oracle.

For a proof oracle $\pi : \{0,1\}^r \to \{0,1\}$, denote $p_{\mathsf{acc}}(\pi) := \Pr_{\mathsf{seed}\leftarrow\{0,1\}^r}[V^\pi(\mathsf{seed}) \text{ accepts}]$. For every input $x \in \{0,1\}^\star$:

- If $x \in L^{\mathsf{hard}}$, then there is a proof oracle π such that $p_{\mathsf{acc}}(\pi) = 1$.
- If $x \notin L^{\mathsf{hard}}$, then for every proof oracle π, we have $p_{\mathsf{acc}}(\pi) \le 0.01$.

Now, suppose that for every input $x \in L^{\mathsf{hard}}$, there is a proof oracle π such that $p_{\mathsf{acc}}(\pi) = 1$, and in addition, π *can be computed by a $\mathscr{C}$ circuit*. (Call this assumption the "easy-witness assumption".) Moreover, suppose that the $\mathsf{GapUNSAT}$ problem for $V^{\mathscr{C}}$ can be solved in $2^r/r^{\omega(1)} < o(2^n)$ time. Then there is a faster nondeterministic algorithm for L^{hard} as follows. Given an input x, we first guess a circuit $\mathscr{C}$ that computes a valid proof oracle π, and use the $\mathsf{GapUNSAT}$ algorithm to distinguish between the case that $p_{\mathsf{acc}}(\pi) = 1$ and that $p_{\mathsf{acc}}(\pi) \le 0.01$.

By the nondeterministic time hierarchy theorem, the above speed-up algorithm has to be incorrect. Therefore, our "easy-witness assumption" has to be false, i.e., there is an input $x \in L^{\mathsf{hard}}$ which does not have valid PCP proofs computable by a small $\mathscr{C}$ circuit.

a) A naïve attempt: Given a circuit $C : \{0,1\}^n \to \{0,1\}^\ell$, our goal is to find a non-output of C in $\mathsf{FP}^{\mathsf{NP}}$. Again, let $L^{\mathsf{hard}} \in \mathsf{NTIME}[\ell] \setminus \mathsf{NTIME}[o(\ell)]$ be the hard language constructed by the nondeterministic time hierarchy.[10] Our "easy-witness" assumption now becomes:

Assumption III.1. For every $x \in L^{\mathsf{hard}}$, there is a PCP proof for x that is in the range of C.

Now we design a faster nondeterministic algorithm M_{fast} that tries to solve L^{hard}. Let $Q_{\mathsf{PCP}} : \{0,1\}^\ell \to \{0,1\}^{2^{|\mathsf{seed}|}}$ be the circuit such that for every PCP proof $\pi \in \{0,1\}^\ell$ and every seed, the seed-th output of $Q_{\mathsf{PCP}}(\pi)$ is the verifier's output when given π as the PCP proof and seed as the randomness. (Here we interpret seed as both a random string of length $|\mathsf{seed}|$ and an integer in $[2^{|\mathsf{seed}|}]$.) Note that if the PCP is efficient enough, then Q_{PCP} is an NC^0 circuit. Let $C'(x) := Q_{\mathsf{PCP}}(C(x))$. To solve L^{hard}, it suffices to solve the Hamming weight estimation problem for C', i.e., distinguish between $\delta(C'(x)) = 1$ and $\delta(C'(x)) \le 0.01$.

There is a serious problem with this approach: The description length of C is already $\Omega(\ell)$, therefore it is impossible to solve the Hamming weight estimation problem in $o(\ell)$ time.

b) Idea 1: Make copies: Our first idea is simple but crucial: we pick a large enough number $H = \mathrm{poly}(\ell)$ and make H copies of C. That is, instead of the avoidance problem for C, we consider the avoidance problem for the circuit

$$C^H(x_1, x_2, \ldots, x_H) = (C(x_1), C(x_2), \ldots, C(x_H)).$$

There is a simple $\mathsf{FP}^{\mathsf{NP}}$ reduction from the avoidance problem of C to the avoidance problem of C^H. Suppose $y = (y_1, y_2, \ldots, y_H)$ is not in the range of C^H, then we can use the NP oracle to check whether each y_i is in the range of C, and pick the first y_i that is not. Hence, it suffices to solve the avoidance problem for C^H.

Now, let $L^{\mathsf{hard}} \in \mathsf{NTIME}[H \cdot \ell] \setminus \mathsf{NTIME}[o(H \cdot \ell)]$. Let $Q_{\mathsf{PCP}} : \{0,1\}^{H\cdot\ell} \to \{0,1\}^{2^{|\mathsf{seed}|}}$ be the NC^0 circuit mapping the PCP proof to the verifier's outputs on each seed. It suffices to design a $\mathsf{HammingHit}$ data structure for the circuit $C'(x) := Q_{\mathsf{PCP}}(C^H(x))$. Note that we only need $O(\ell)$ bits to describe C': Q_{PCP} is completely determined by the PCP verifier, and we can use $O(\ell)$ bits to describe C. As $O(\ell) \ll H \cdot \ell$, at least in principle, it could be possible to solve the $\mathsf{HammingHit}$ problem for C' in less than $H \cdot \ell$ time.

But how do we *actually* solve the $\mathsf{HammingHit}$ problem? We need to exploit the structures of the circuit Q_{PCP} (if any)! What property should the PCP have?

c) Idea 2: Rectangular PCP: Our second idea is to use *rectangular* PCPs. In this overview, let us assume the PCP is *perfectly* rectangular.

We recall the definition of rectangular PCPs. Here, the PCP proof π is an $H \times \ell$ matrix, and our easy-witness assumption becomes that every row of π is in the range of C. The PCP randomness seed is divided into two parts: $\mathsf{seed.row}$ and $\mathsf{seed.col}$. The row index of each query only depends on $\mathsf{seed.row}$, and the column index of each query only depends on $\mathsf{seed.col}$.

We enumerate $\mathsf{seed.row}$. Denote the PCP verifier as V, we want to estimate

$$\Pr_{\mathsf{seed.col}}[V^\pi(\mathsf{seed.row}, \mathsf{seed.col}) \text{ accepts}]. \tag{1}$$

As $\mathsf{seed.row}$ is fixed, we now know q rows $r_1, r_2, \ldots, r_q$ such that $V^\pi(\mathsf{seed.row}, -)$ will only access these rows of π. Call these rows $\pi_{r_1}, \pi_{r_2}, \ldots, \pi_{r_q} \in \{0,1\}^\ell$. Let $Q_{\mathsf{PCP}} : (\{0,1\}^\ell)^q \to \{0,1\}^{2^{|\mathsf{seed.col}|}}$ be the NC^0 circuit such that for every $\mathsf{seed.col}$, the $\mathsf{seed.col}$-th output of $Q_{\mathsf{PCP}}(\pi_{r_1}, \pi_{r_2}, \ldots, \pi_{r_q})$ is 1 if and only if $V^\pi(\mathsf{seed.row}, \mathsf{seed.col})$ accepts. Let $x_1, x_2, \ldots, x_q \in \{0,1\}^n$, define the following $\mathsf{NC}^0 \circ \mathscr{C}$ circuit:

$$C'(x_1, x_2, \ldots, x_q) = Q_{\mathsf{PCP}}(C(x_1), C(x_2), \ldots, C(x_q)).$$

We guess the strings $w_1, w_2, \ldots, w_H \in \{0,1\}^n$ such that the i-th row of π is equal to $C(w_i)$. It is easy to see that

$$(1) = \delta(C'(w_{r_1}, w_{r_2}, \ldots, w_{r_q})).$$

Therefore, we can use a $\mathsf{HammingHit}$ data structure for C' to estimate Eq. (1). Note that q is a constant, and $|\mathsf{seed.col}| \approx \log \ell$, so $C' : \{0,1\}^{qn} \to \{0,1\}^{2^{|\mathsf{seed.col}|}}$ is *indeed* small (instead of only having a short description).

To summarise, our speed-up algorithm M_{fast} proceeds as follows. First, we guess the inputs $w_1, w_2, \ldots, w_H$, (implicitly) construct an $H \times \ell$ proof matrix π whose i-th row is

[10] Note that we have not specified the input length for L^{hard}. We only know that L^{hard} is in non-deterministic ℓ time on this input length. This important issue will be discussed later.

equal to $C(w_i)$, and hope that π is a valid PCP proof. Then, we estimate the probability that the PCP verifier accepts. To do so, we enumerate $\mathsf{seed.row}$ and use the $\mathsf{HammingHit}$ data structure to estimate Eq. (1). If for any $\mathsf{seed.row}$ it happens that (1) ≤ 0.01, then we reject; otherwise, we accept.

Since the query algorithm for $\mathsf{HammingHit}$ takes $2^{|\mathsf{seed.col}|}/|\mathsf{seed.col}|^{\omega(1)}$ time, the time complexity of M_{fast} is

$$2^{|\mathsf{seed.row}|} \cdot 2^{|\mathsf{seed.col}|}/|\mathsf{seed.col}|^{\omega(1)} \le (H\ell)/\log^{\omega(1)} \ell.$$

d) The "right" time hierarchy theorem: The above avoidance algorithm is only correct on infinitely many input lengths. The reason is that the nondeterministic time hierarchy in [62] only works infinitely often, i.e., for any $\mathsf{NTIME}[o(H\ell)]$ machine M, L^{hard} and M only disagree on infinitely many input lengths.

To obtain an almost-everywhere avoidance algorithm, we follow the ideas of [16]. The crucial observation is that M_{fast} does not guess too many nondeterministic bits. (In the case of the Algorithmic Method, it only guesses a small circuit encoding the PCP proof; in our case, it only guesses $Hn \ll H\ell$ bits.) There is an almost-everywhere nondeterministic time hierarchy against such machines [23]. Let $\mathsf{NTIMEGUESS}[T(N), g(N)]$ denote the class of languages decidable by a nondeterministic machine running in $T(N)$ time and guessing $g(N)$ bits. Then:

Theorem III.2 ([23]). *Let $T(N)$ be a time-constructible function such that $N \le T(N) \le 2^{\mathrm{poly}(N)}$. There is a language $L^{\mathsf{hard}} \in \mathsf{NTIME}[T(N)] \setminus \text{i.o.-}\mathsf{NTIMEGUESS}[o(T(N)), N/10]$.*

Since we need to guess Hn bits, we set the input length to be $N := 10Hn$. We also set $T(N)$ to be a slightly super-linear function such that $T(10Hn) \approx H\ell$.

There is a small issue: M_{fast} needs to access the data structure DS for $\mathsf{HammingHit}$. We cannot compute DS inside M_{fast} as it needs an NP oracle, therefore our only option is to hardcode DS as advice for M_{fast}. Fortunately, the above NTIME hierarchy theorem also holds against machines with $N/10$ advice bits:

Theorem III.3. *Let $T(N)$ be a time-constructible function such that $N \le T(N) \le 2^{\mathrm{poly}(N)}$. There is a language $L^{\mathsf{hard}} \in \mathsf{NTIME}[T(N)] \setminus \text{i.o.-}\mathsf{NTIMEGUESS}[o(T(N)), N/10]_{\underline{/(N/10)}}$.*

Note that we only need the $\mathsf{HammingHit}$ data structure for the circuit C' whose size is independent of H. By setting H large enough, we can still guarantee that the advice length is $\le N/10 = Hn/10$.[11]

To complete the description of our $\mathsf{FP}^{\mathsf{NP}}$ avoidance algorithm, we still need one ingredient from [16]: a *refuter* for Theorem III.3. Given 1^N and the code of the machine M_{fast} that attempts to compute L^{hard}, as well as the $N/10$ advice bits, if M_{fast} runs in $o(T(N))$ time and uses at most $N/10$ nondeterministic bits, then the refuter finds an input $x \in \{0,1\}^N$ such that $M_{\mathsf{fast}}(x) \ne L^{\mathsf{hard}}(x)$. The refuter runs in polynomial time with access to an NP oracle.

Our $\mathsf{FP}^{\mathsf{NP}}$ avoidance algorithm is as follows. We first compute the $\mathsf{HammingHit}$ structure DS in $\mathsf{FP}^{\mathsf{NP}}$. We also compute (the code of) the machine M_{fast}. Then we use the refuter to find an input $x_{\mathsf{hard}} \in \{0,1\}^N$ such that $M_{\mathsf{fast}}(x_{\mathsf{hard}}) \ne L^{\mathsf{hard}}(x_{\mathsf{hard}})$. It follows that in any valid proof matrix of $x_{\mathsf{hard}} \in L^{\mathsf{hard}}$, there is some row that is not in the range of C. We can then simply use the NP oracle to pick the first such row.

e) Rectangular PCP of proximity: There is another issue: Q_{PCP} depends on the input x_{hard}! As x_{hard} depends on DS (recall that x_{hard} is found by the refuter, which takes DS as input), we cannot preprocess $C' = Q_{\mathsf{PCP}} \circ C$ before we know x_{hard}.

Our solution is to use a rectangular PCP of *proximity* (henceforth rectangular PCPP). Recall that a PCPP verifier can only query a small number of bits in both the proof oracle and the input oracle. (As it does not even have time to read the whole input, its query pattern does not depend on it.) In a rectangular PCPP, the input oracle is also accessed in a rectangular fashion. There are three predicates V_{type}, V_{row}, and V_{col}:

- V_{type}, without looking at seed, outputs q symbols, where each symbol is either input or proof.[12]
- V_{row} reads $\mathsf{seed.row}$ and outputs q row indices $r_1, r_2, \ldots, r_q$.
- V_{col} reads $\mathsf{seed.col}$ and outputs q column indices $c_1, c_2, \ldots, c_q$.
- For each query $i \in [q]$, if the i-th symbol is input, then the i-th query asks the (r_i, c_i)-th entry of the input matrix; if the i-th symbol is proof, then the i-th query asks the (r_i, c_i)-th entry of the proof matrix.

We now revise our speed-up algorithm M_{fast} for L^{hard} using rectangular PCPPs. Given an input $x \in \{0,1\}^N$,[13] we still guess $w_1, w_2, \ldots, w_H$ and construct the PCPP proof matrix π whose i-th row is $C(w_i)$. Also, the input is treated as an $H' \times W'$ matrix[14]; let x_i be the i-th row of the input matrix. Now we estimate the probability that $V^{x,\pi}(\mathsf{seed})$ accepts, where V is the PCPP verifier with oracle access to x and π. After enumerating $\mathsf{seed.row}$, we have fixed q_{proof} rows in the proof matrix and q_{input} rows in the input matrix, where $q_{\mathsf{proof}} + q_{\mathsf{input}} = q$, and the output of $V^{x,\pi}(\mathsf{seed.row}, -)$ only depends on these rows. Let $Q_{\mathsf{PCPP}} : \{0,1\}^{q_{\mathsf{proof}} \cdot n + q_{\mathsf{input}} \cdot W'} \to \{0,1\}^{2^{|\mathsf{seed.col}|}}$ be the circuit that maps these rows to the

[11] In the case of almost rectangular PCPs, we need to hardcode a data structure for every possible value of $\mathsf{seed.shared}$. It is still possible to set the parameters so that the total length of these data structures is $\le N/10$.

[12] Note that we consider perfect rectangularity here. In an almost rectangular PCPP, V_{type} depends on $\mathsf{seed.shared}$, but does not depend on $\mathsf{seed.row}$ and $\mathsf{seed.col}$.

[13] Note that a PCPP could only distinguish between $x \in L$ and x being *far* from L. Thus, we need to apply an error-correcting code to the input. For simplicity, we still use x to denote the encoded input.

[14] A technicality here is that we want to set W' to be as small as possible, as the size of C' is proportional to W'. It turns out that we can achieve $W' = n \cdot \mathrm{polylog}(\ell)$.

verifier's outputs on each $\mathsf{seed.col}$,[15] and

$$\begin{aligned}&C'(w_1, w_2, \ldots, w_{q_{\mathsf{proof}}}, x_1, x_2, \ldots, x_{q_{\mathsf{input}}})\\ =&Q_{\mathsf{PCPP}}(C(w_1), C(w_2), \ldots, C(w_{q_{\mathsf{proof}}}), x_1, x_2, \ldots, x_{q_{\mathsf{input}}}).\end{aligned}$$

Note that Q_{PCPP} does not depend on the input x.

For each $\mathsf{seed.row}$, we feed the corresponding rows in the proof matrix and the input matrix into C', and use the $\mathsf{HammingHit}$ data structure to estimate the probability over $\mathsf{seed.col}$ that $V^{x,\pi}(\mathsf{seed})$ accepts. The total running time is

$$2^{|\mathsf{seed.row}|} \cdot 2^{|\mathsf{seed.col}|} / |\mathsf{seed.col}|^{\omega(1)} < H\ell / \log^{\omega(1)} \ell.$$

Finally, our $\mathsf{FP}^{\mathsf{NP}}$ avoidance algorithm is the same as before, except that we use the rectangular PCPP in the code of M_{fast}.

Acknowledgment

Hanlin Ren wants to thank Yuichi Yoshida for helpful discussions about hypergraph cut sparsifiers, Stanislav Živný for helpful discussions about [45], and Lijie Chen for helpful discussions in general and about $\mathsf{THR} \circ \mathsf{THR}$ circuits specifically. We thank Jiatu Li for helpful comments on a draft version of this paper. We thank Ján Pich for helpful discussions during the early stage of this research.

References

[1] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Pseudorandom generators in propositional proof complexity. *SIAM J. Comput.*, 34(1):67–88, 2004.

[2] Josh Alman and Lijie Chen. Efficient construction of rigid matrices using an NP oracle. In *Proc. 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1034–1055, 2019.

[3] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM Journal of Computing*, 36(4):845–888, 2006.

[4] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[5] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[6] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. *SIAM J. Comput.*, 36(4):889–974, 2006.

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

[8] Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM J. Comput.*, 38(2):551–607, 2008.

[9] Eli Ben-Sasson and Emanuele Viola. Short PCPs with projection queries. In *Proc. 41st International Colloquium on Automata, Languages and Programming (ICALP)*, volume 8572 of *Lecture Notes in Computer Science*, pages 163–173, 2014.

[10] Amey Bhangale, Prahladh Harsha, Orr Paradise, and Avishay Tal. Rigid matrices from rectangular PCPs or: Hard claims have complex proofs. In *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 858–869, 2020.

[11] Harry Buhrman, Lance Fortnow, and Thomas Thierauf. Nonrelativizing separations. In *Proc. 13th Annual IEEE Conference on Computational Complexity (CCC)*, pages 8–12, 1998.

[12] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *Proc. 31st Computational Complexity Conference (CCC)*, volume 50 of *LIPIcs*, pages 10:1–10:24, 2016.

[13] Lijie Chen. Non-deterministic quasi-polynomial time is average-case hard for ACC circuits. In *Proc. 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1281–1304, 2019.

[14] Lijie Chen, Zhenjian Lu, Xin Lyu, and Igor Carboni Oliveira. Majority vs. approximate linear sum and average-case complexity below NC^1. In *Proc. 48th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 198 of *LIPIcs*, pages 51:1–51:20, 2021.

[15] Lijie Chen and Xin Lyu. Inverse-exponential correlation bounds and extremely rigid matrices from a new derandomized XOR lemma. In *Proc. 53rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 761–771, 2021.

[16] Lijie Chen, Xin Lyu, and R. Ryan Williams. Almost-everywhere circuit lower bounds from non-trivial derandomization. In *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1–12, 2020.

[17] Lijie Chen and Hanlin Ren. Strong average-case lower bounds from non-trivial derandomization. In *Proc. 52nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 1327–1334, 2020.

[18] Lijie Chen and R. Ryan Williams. Stronger connections between circuit analysis and circuit lower bounds, via PCPs of proximity. In *Proc. 34th Computational Complexity Conference (CCC)*, volume 137 of *LIPIcs*, pages 19:1–19:43, 2019.

[19] Ruiwen Chen, Igor Carboni Oliveira, and Rahul Santhanam. An average-case lower bound against ACC^0. In *Proc. 13th Latin American Theoretical Informatics Symposium (LATIN)*, volume 10807 of *Lecture Notes in Computer Science*, pages 317–330, 2018.

[20] Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007.

[21] Paul Erdős. Graph theory and probability. *Canadian Journal of Mathematics*, 11:34–38, 1959.

[22] Magnus Gausdal Find, Alexander Golovnev, Edward A. Hirsch, and Alexander S. Kulikov. A better-than-$3n$ lower bound for the circuit complexity of an explicit function. In *Proc. 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 89–98, 2016.

[23] Lance Fortnow and Rahul Santhanam. New non-uniform lower bounds for uniform classes. In *Proc. 31st Computational Complexity Conference (CCC)*, volume 50 of *LIPIcs*, pages 19:1–19:14, 2016.

[24] Gramoz Goranci, Harald Räcke, Thatchaphol Saranurak, and Zihan Tan. The expander hierarchy and its applications to dynamic graph algorithms. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2212–2228, 2021.

[25] Aryeh Grinberg, Ronen Shaltiel, and Emanuele Viola. Indistinguishability by adaptive procedures with advice, and lower bounds on hardness amplification proofs. In *Proc. 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 956–966, 2018.

[26] Dan Gutfreund and Guy N. Rothblum. The complexity of local list decoding. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, August 25-27, 2008. Proceedings*, volume 5171 of *Lecture Notes in Computer Science*, pages 455–468. Springer, 2008.

[27] Xuangui Huang and Emanuele Viola. Average-case rigidity lower bounds. In *Proc. 16th International Computer Science Symposium in Russia (CSR)*, volume 12730 of *Lecture Notes in Computer Science*, pages 186–205, 2021.

[28] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. Comput. Syst. Sci.*, 65(4):672–694, 2002.

[29] Russell Impagliazzo and Avi Wigderson. Randomness vs time: Derandomization under a uniform assumption. *Journal of Computer and System Sciences*, 63(4):672–688, 2001.

[30] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Proc. 41st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 294–304, 2000.

[31] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *Proc. 29th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 244–256, 2002.

[32] Kazuo Iwama and Hiroki Morizumi. An explicit lower bound of $5n - o(n)$ for Boolean circuits. In *Proc. 27th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 2420 of *Lecture Notes in Computer Science*, pages 353–364, 2002.

[15] Actually, Q_{PCPP} also depends on $O(q)$ *parity-check* bits. We ignore this technical detail in the overview.

[33] Emil Jeřábek. Dual weak pigeonhole principle, Boolean complexity, and derandomization. *Ann. Pure Appl. Log.*, 129(1-3):1–37, 2004.

[34] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos Papadimitriou. Total functions in the polynomial hierarchy. In *Proc. 12th Conference on Innovations in Theoretical Computer Science (ITCS)*, volume 185 of *LIPIcs*, pages 44:1–44:18, 2021.

[35] Johannes Köbler and Osamu Watanabe. New collapse consequences of NP having small circuits. *SIAM J. Comput.*, 28(1):311–324, 1998.

[36] Oliver Korten. The hardest explicit construction. In *Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 433–444. IEEE, 2021.

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

[38] Jan Krajíček. Dual weak pigeonhole principle, pseudo-surjective functions, and provability of circuit lower bounds. *J. Symb. Log.*, 69(1):265–286, 2004.

[39] Jan Krajíček. On the proof complexity of the Nisan-Wigderson generator based on a hard $\mathsf{NP} \cap \mathsf{coNP}$ function. *J. Math. Log.*, 11(1), 2011.

[40] Kasper Green Larsen and R. Ryan Williams. Faster online matrix-vector multiplication. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2182–2189. SIAM, 2017.

[41] Jiatu Li and Tianqi Yang. $3.1n - o(n)$ circuit lower bounds for explicit functions. In *Proc. 54th Annual ACM Symposium on Theory of Computing (STOC)*, 2022. To appear.

[42] Cody D. Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime from a new easy witness lemma. *SIAM J. Comput.*, 49(5), 2020.

[43] V. A. Nepomnjascii. Rudimentary predicates and Turing computations. In *Doklady Akademii Nauk*, volume 195, pages 282–284. Russian Academy of Sciences, 1970.

[44] Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Proc. 32nd Computational Complexity Conference (CCC)*, volume 79 of *LIPIcs*, pages 18:1–18:49, 2017.

[45] Eden Pelleg and Stanislav Živný. Additive sparsification of CSPs. In *Proc. 29th European Symposium on Algorithms (ESA)*, volume 204 of *LIPIcs*, pages 75:1–75:15, 2021.

[46] Mark S Pinsker. On the complexity of a concentrator. In *7th International Telegraffic Conference*, volume 4, pages 1–318, 1973.

[47] Mihai Pătraşcu and Mikkel Thorup. Planning for fast connectivity updates. In *Proc. 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 263–271, 2007.

[48] Alexander Razborov. Pseudorandom generators hard for k-DNF resolution and polynomial calculus resolution. *Annals of Mathematics*, 181(2):415–472, 2015.

[49] Alexander A. Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997.

[50] Rahul Santhanam and R. Ryan Williams. On medium-uniformity and circuit lower bounds. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 15–23. IEEE Computer Society, 2013.

[51] Ronen Shaltiel and Emanuele Viola. Hardness amplification proofs require majority. *SIAM J. Comput.*, 39(7):3122–3154, 2010.

[52] Claude E. Shannon. The synthesis of two-terminal switching circuits. *Bell System technical journal*, 28(1):59–98, 1949.

[53] Avishay Tal. Formula lower bounds via the quantum method. In *Proc. 49th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1256–1268. ACM, 2017.

[54] Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *Proc. 6th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 53 of *Lecture Notes in Computer Science*, pages 162–176, 1977.

[55] Emanuele Viola. New lower bounds for probabilistic degree and AC^0 with parity gates. *Electron. Colloquium Comput. Complex.*, page 15, 2020.

[56] Nikhil Vyas and R. Ryan Williams. Lower bounds against sparse symmetric functions of ACC circuits: Expanding the reach of #SAT algorithms. In *Proc. 37th Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 154 of *LIPIcs*, pages 59:1–59:17, 2020.

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

[58] R. Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014.

[59] R. Ryan Williams. Natural proofs versus derandomization. *SIAM Journal of Computing*, 45(2):497–529, 2016.

[60] R. Ryan Williams. Limits on representing Boolean functions by linear combinations of simple functions: Thresholds, ReLUs, and low-degree polynomials. In *Proc. 33rd Computational Complexity Conference (CCC)*, volume 102 of *LIPIcs*, pages 6:1–6:24, 2018.

[61] R. Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. *Theory Comput.*, 14(1):1–25, 2018.

[62] Stanislav Zák. A Turing machine time hierarchy. *Theor. Comput. Sci.*, 26:327–333, 1983.