Electronic Colloquium on Computational Complexity, Report No. 210 (2025)

Range avoidance, Arthur-Merlin, and TFNP

Surendra Ghentiyala[*] Zeyong Li[†] Noah Stephens-Davidowitz[‡]

November 24, 2025

Abstract

Range avoidance (Avoid) is the computational problem in which the input is an expanding circuit $C : \{0,1\}^n \to \{0,1\}^{n+1}$ and the goal is to find a string $y \in \{0,1\}^{n+1}$ that is *not* in the image of C. Avoid was introduced recently by Kleinberg, Korten, Mitropolsky, and Papadimitriou [ITCS 2021] as an example of a total search problem that appears not to live in TFNP but does live in the second level of the total function polynomial hierarchy. Since then, Avoid has found surprising applications throughout complexity theory, and in theoretical computer science more broadly.

Our main results are as follows.

1. We show that any decision problem that efficiently reduces to Avoid is in $\mathsf{AM} \cap \mathsf{coAM}$ (even for promise problems, and even if the reduction is randomized and makes many adaptive queries). This in particular shows that NP-hardness of Avoid would collapse the polynomial hierarchy, answering an open question that has arisen numerous times in the literature.
2. We show an efficient randomized reduction from Avoid to a problem in TFNP that succeeds with probability $1 - \varepsilon$ for any $\varepsilon \geq 1/\operatorname{poly}(n)$ (under complexity-theoretic assumptions). This provides additional evidence that Avoid is unlikely to be NP-hard. And, it shows that, though Avoid itself is almost certainly not in TFNP, it is in some sense extremely close to lying in TFNP. The randomness in our reduction seems necessary, since Chen and Li [STOC 2024] showed (under cryptographic assumptions) that Avoid $\notin$ SearchNP, while a deterministic reduction from Avoid to a TFNP problem *would* place Avoid in SearchNP.

The high-level idea behind these two results is a rather simple "search Arthur-Merlin-Arthur protocol for Avoid." And, a key technical tool that we use in all of our results is a novel AM protocol for upper bounding the size of the image of a circuit. This latter protocol can be viewed as a sort of dual of the celebrated set-size lower bound protocol due to Goldwasser and Sipser [STOC 1986]. Both protocols seem likely to be of independent interest.

[*] Cornell University. Email: sg974@cornell.edu. This work was supported by NSF Grant No. CCF-2312296, a Packard Foundation Fellowship, and a generous gift from Google.

[†] National University of Singapore. Email: li.zeyong@u.nus.edu. This work was supported by NRF grant NRF-NRFI09-0005.

[‡] Cornell University. Email: noahsd@gmail.com. This work was supported by NSF Grant No. CCF-2312296, a Packard Foundation Fellowship, and a generous gift from Google.

ISSN 1433-8092

Contents

1 Introduction

Range avoidance (Avoid) is the computational problem in which the input is an expanding circuit $C : \{0,1\}^n \to \{0,1\}^m$ for $m > n$, and the goal is to find a string $y \in \{0,1\}^m$ that is *not* in the image of C. We consider the hardest case when $m = n + 1$ (except where we explicitly state otherwise).

Avoid was introduced relatively recently by Kleinberg, Korten, Mitropolsky, and Papadimitriou [KKMP21] as a beautiful example of a total search problem that appears not to be in TFNP but does live in the second level of the total function polynomial hierarchy (formally, $\mathsf{TF\Sigma_2^P}$). Notice in particular that Avoid is in fact a total search problem, since for every expanding function there must be at least one element not in its image. Indeed, Avoid can be thought of as a computational problem that captures the "dual weak pigeonhole principle," which states that if there are many more holes than pigeons, there must be an empty hole.

In fact, an instance of Avoid always has *many* solutions, in the sense that a random element $y \sim \{0,1\}^{n+1}$ will be a solution with probability $1 - |\mathrm{Im}(C)|/2^{n+1} \geq 1/2$. This makes Avoid a rather strange computational problem. On one hand, it is widely believed that Avoid $\notin \mathsf{FP}$, i.e., that there is *not* a deterministic polynomial-time algorithm that solves Avoid. On the other hand, the trivial randomized algorithm that simply outputs a random string $y \sim \{0,1\}^{n+1}$ solves Avoid with probability at least $1/2$!

Avoid grew in prominence due to Korten's celebrated work showing that it is in a certain precise sense "the hardest computational problem that is solved by a random string with decent probability" [Kor21]. In particular, Korten showed reductions to range avoidance from the problems of constructing two-source extractors, truth tables with high circuit complexity, pseudorandom strings, rigid matrices, etc. All of these are objects that we do not know how to construct deterministically in polynomial time, but in all cases a suitable random distribution will yield such an object with good probability.

In the few years since [KKMP21] and [Kor21], Avoid has been the subject of intense study (e.g., [Kor22, RSW22, GLW22, ILW23, CHLR23, GGNS23, CGL+23, CL24, KP24, GLV24, LORS24, KS25a, Kor25, HV25, KPI25, LZ25, KS25b, FIM25, RWZ25, GGLS26]), due mostly to its intimate connections with derandomization and circuit lower bounds. In particular, a series of recent exciting breakthroughs on circuit lower bounds by [CHR24, Li24, CLL25] followed from showing non-trivial algorithms for Avoid in exotic models of computation.

In fact, if we could actually prove that Avoid $\in \mathsf{FP}$, then we would immediately have explicit constructions for many objects of interest, *and* we would have proven that $\mathsf{BPP} = \mathsf{P}$, *and* we would have proven incredibly strong new circuit lower bounds! However, as we have already mentioned, Avoid is widely believed to *not* be in FP. Indeed, work of Ilango, Li, and Williams showed that Avoid $\notin \mathsf{FP}$ under the assumption that $\mathsf{NP} \neq \mathsf{coNP}$ together with a strong cryptographic assumption [ILW23]. And, later work of Chen and Li and Ren, Wang, and Zhong even showed that Avoid $\notin \mathsf{SearchNP}$ under similarly strong assumptions from complexity theory and cryptography (even when the input circuit is quite restricted) [CL24, RWZ25]. However, we do not have any evidence for hardness of Avoid that simply works via reduction from a presumed hard problem. (In particular, all of the explicit construction problems that Korten reduced to Avoid are widely believed to be in FP, in spite of the lack of known algorithms.)

It would be particularly nice to reduce SAT to Avoid, thereby proving that Avoid is NP-hard and therefore that Avoid $\notin \mathsf{FP}$ under the minimal assumption that $\mathsf{P} \neq \mathsf{NP}$. Indeed, the question of whether Avoid is NP-hard or whether NP-hardness of Avoid can be ruled out under standard complexity-theoretic assumptions has arisen repeatedly in the literature (e.g., in [RSW22, ILW23, CGL$^+$23, CL24]), including as Problem 3 in Korten's survey on range avoidance [Kor25]. Prior work has merely observed that reductions from SAT to Avoid with very large stretch m (or reductions with relatively large stretch m that only make a small number of oracle calls) would imply that $\mathsf{NP} = \mathsf{RP}$ [ILW23, CGL$^+$23]. (See Section 1.3.) But, prior work has not managed to rule out NP-hardness of Avoid when the stretch is just $m = n + 1$ and the reduction is unrestricted.

1.1 Our results

1.1.1 Range avoidance is probably not that hard

Here, we briefly present our two main results. In Section 1.1.3, we provide more discussion and interpretation.

Range avoidance is "kind of in $\mathsf{AM} \cap \mathsf{coAM}$." Our first main contribution essentially settles the question of whether Avoid is NP-hard by showing that such hardness would collapse the polynomial hierarchy. This applies even for the hardest case when $m = n + 1$, and even under adaptive randomized reductions that make many queries to an oracle for Avoid. In fact, we show that any decision problem (or even promise problem) that reduces to Avoid must lie in the complexity class $\mathsf{AM} \cap \mathsf{coAM}$. (Below and throughout the paper, we write prC to represent the promise-problem version of a complexity class C.)

Theorem 1.1 (See Theorem 5.2)**.** $\mathsf{BPP}^{\text{Avoid}} \subseteq \mathsf{AM} \cap \mathsf{coAM}$ *and* $\mathsf{prBPP}^{\text{Avoid}} \subseteq \mathsf{prAM} \cap \mathsf{prcoAM}$.

(Note that Theorem 1.1 places *decision* problems that reduce to Avoid into $\mathsf{AM} \cap \mathsf{coAM}$, but it does not seem to directly imply that Avoid itself lies in any particularly natural complexity class.)

As a consequence of Theorem 1.1, NP-hardness of Avoid (even under randomized reductions that make many adaptive queries) would collapse the polynomial hierarchy to AM [BHZ87], thus answering [Kor25, Problem 3]. More generally, one can think of Theorem 1.1 as effectively placing Avoid in a rather low level of the polynomial hierarchy. This is in contrast to the closely related problem Empty (which is sometimes called "strong Avoid") that was also defined in [KKMP21], which is in fact NP-hard.

Range avoidance is *almost* in TFNP. Our second main contribution gives a more direct upper bound on Avoid by showing that it reduces via a (one-query) randomized reduction to a problem in TFNP, under the assumption that sufficiently strong PRGs exist.

What we mean by this is rather subtle, since in general randomized algorithms for search problems are rather subtle. (See, e.g., [ABK24] for recent discussion of this.) In this context, we mean that there is a single search problem $\mathrm{T} \in \mathsf{TFNP}$ such that for any $\varepsilon := \varepsilon(n) \geq 1/\operatorname{poly}(n)$, there is a randomized reduction from Avoid to T that succeeds with probability at least $1 - \varepsilon$ (after making just one oracle query), as in the following theorem. Notice that this is much higher than the success probability of $1/2$ achieved by the trivial randomized algorithm.

Theorem 1.2 (Informal; see Theorem 6.1)**.** *Assuming that sufficiently strong PRGs exist, there is a problem* $\mathrm{T} \in \mathsf{TFNP}$ *such that for every* $\varepsilon := \varepsilon(n) \geq 1/\operatorname{poly}(n)$*, there is a (one-query) randomized reduction from* Avoid *to* T *that succeeds with probability at least* $1 - \varepsilon$*.*

In particular, under this assumption, any decision problem that reduces to Avoid *is in* $\mathsf{BPP}^{\mathsf{TFNP}}$*.*

The specific version of PRGs that we need for this result are PRGs that fool $\mathsf{NP}_{||}$ circuits[1] (of fixed polynomial size). Such PRGs are believed to exist, and they are even known to exist under the quite likely assumption that E does not have $2^{o(n)}$-size non-deterministic circuits [SS24]. (See Sections 2.3 and 6 for more discussion.)

Theorem 1.2 can of course be interpreted as additional evidence that Avoid is unlikely to be NP-hard. Specifically, the theorem of course implies that Avoid is not NP-hard unless SAT $\in \mathsf{BPP}^{\mathsf{TFNP}}$ (assuming the existence of sufficiently strong PRGs). See Section 6 for more discussion.

1.1.2 Two Arthur-Merlin protocols

A $\mathsf{SearchAMA}$ protocol for range avoidance. The high-level idea behind the proofs of both Theorems 1.1 and 1.2 is a simple $\mathsf{SearchAMA}$ protocol for Avoid. This protocol may be of independent interest, and we describe it in detail in Section 1.2.1. (Our definition of $\mathsf{SearchAMA}$ is rather delicate, but the protocol itself is quite simple. See Section 1.2.1 for a nearly complete description of the protocol and Section 2.5 for our definition of $\mathsf{SearchAMA}$.)

Theorem 1.3 (Informal; see Theorem 4.1)**.** Avoid $\in \mathsf{SearchAMA}$.

A range-size upper bound protocol. Finally, a key technical tool that we use to prove Theorems 1.1, 1.2 and 1.3 is a novel AM protocol for upper bounding the image size of a circuit $C : \{0,1\}^n \to \{0,1\}^m$, as in the following theorem.

Theorem 1.4 (A range-size upper bound protocol; see Theorem 3.2)**.** *For any $\varepsilon := \varepsilon(n) \geq 1/\operatorname{poly}(n)$, there is an AM protocol for the following promise problem. The input is a circuit $C : \{0,1\}^n \to \{0,1\}^m$ and an estimate $\tau \in [2^n]$ of the size of the image, $|\mathrm{Im}(C)|$. It is a YES instance if $|\mathrm{Im}(C)| \leq \tau$ and it is a NO instance if $|\mathrm{Im}(C)| \geq \tau + \varepsilon 2^n$.*

We expect that Theorem 1.4 will be of independent interest and have additional applications. In particular, Theorem 1.4 is perhaps best viewed as complementing the celebrated Goldwasser-Sipser set-size lower bound protocol [GS86]. Recall that the set-size lower bound protocol allows Merlin to convince Arthur of a *lower* bound on the size of any set $S \subseteq \{0,1\}^m$ that can be recognized in NP (or even AM; see Section 3.1) up to multiplicative error $1 + 1/\operatorname{poly}(n)$. (Unsurprisingly, we use the set-size lower bound protocol from [GS86] as a key subroutine in our range-size upper bound protocol.)

In particular, notice that the Goldwasser-Sipser protocol allows Merlin to convince Arthur of a *lower* bound on the size of any set $S \subseteq \{0,1\}^m$ that can be represented as the image of a polynomial-size circuit $C : \{0,1\}^n \to \{0,1\}^m$ (i.e., $S = \mathrm{Im}(C)$). Our range-size upper bound protocol allows Merlin to convince Arthur of an *upper* bound on the size of such a set $S \subseteq \{0,1\}^m$, up to additive error $2^n/\operatorname{poly}(n)$.

1.1.3 Some additional discussion

Perhaps the most obvious interpretation of Theorems 1.1, 1.2 and 1.3 is simply as evidence that Avoid is unlikely to be NP-hard. Here, we list a number of additional interpretations and consequences of our results.

[1] Circuits with parallel NP-gates.

Between two trivial algorithms for Avoid**.** We recall two basic facts about Avoid that were already observed when the problem was introduced in [KKMP21]. As we have already discussed, there is a trivial efficient randomized algorithm that solves Avoid with probability 1/2. A closely related fact is that Avoid $\in \mathsf{FZPP}^{\mathsf{NP}}$, i.e., there is a *zero-error* randomized algorithm with access to an NP oracle.[2]

Theorems 1.2 and 1.3 tell us that something interesting happens between the zero-error regime of the $\mathsf{FZPP}^{\mathsf{NP}}$ algorithm and the 1/2 error regime of the trivial algorithm. Specifically, Theorem 1.2 shows that an oracle for a TFNP problem is sufficient if one is willing to accept failure probability $1/\operatorname{poly}(n)$ rather than zero error (assuming that sufficiently strong PRGs exist). And, Theorem 1.3 similarly shows that if one is willing to accept a small polynomial failure probability, then interaction with Merlin is sufficient.

Avoid **is almost certainly not in TFNP, but it comes pretty close.** Notice that Avoid is not in TFNP unless $\mathsf{P} = \mathsf{NP}$,[3] and under strong cryptographic assumptions, Chen and Li and Ren, Wang, and Zhong showed that Avoid is not even in the (much larger) class $\mathsf{SearchNP}$ [CL24, RWZ25]. However, Theorem 1.2 says that Avoid in some sense is "surprisingly close to being contained in TFNP." Specifically, it reduces to a problem in TFNP (under a reasonable complexity-theoretic assumption) via a randomized reduction. Indeed, it is easy to see that any problem that reduces to a problem in TFNP under *deterministic* reductions must lie in $\mathsf{SearchNP}$. So, the randomness in our result is in fact necessary (under the cryptographic assumptions used in [CL24] or [RWZ25]).

Derandomization and algorithms for Avoid**.** Recall that a celebrated result of Korten (closely related to much earlier work of Jeřábek [Jeř04]) showed that E^{NP} does not have circuits of size $2^{o(n)}$ *if and only if* Avoid reduces to a problem in NP under deterministic Turing reductions (i.e., if and only if Avoid $\in \mathsf{FP}^{\mathsf{NP}}$) [Kor21]. Theorem 1.2 can be viewed as a variant of (one direction of) this result. Specifically, Theorem 1.2 shows that an even stronger circuit lower bound (the assumption that E requires exponential-size non-deterministic circuits) implies that Avoid actually reduces to a problem that is not only in NP but is actually in TFNP—albeit under randomized reductions.

Theorem 1.2 can also be viewed as a variant of a beautiful result by Korten and Santhanam [KS25a]. [KS25a] showed that if even stronger PRGs exist, then the *uniform* variant of Avoid with sufficient stretch is actually in FP. (In other words, for any $\operatorname{poly}(n)$-time computable family of expanding circuits $C_n : \{0,1\}^n \to \{0,1\}^m$, Avoid can be solved in deterministic polynomial time when restricted to this family. See Section 1.3.) We show how to solve a notably harder problem (Avoid with minimal stretch on arbitrary inputs) under a weaker assumption, but only via a randomized algorithm with access to an oracle for a problem in TFNP. (Our proof was also inspired by the ideas in [KS25a], as we discuss in Section 1.3.)

[2] Given access to an NP oracle, one can simply repeatedly sample a random string, use the NP oracle to check if it is in the image of the circuit, and output it if it is not.

[3] This is because the formal definition of TFNP requires that there is a verifier that determines whether any candidate solution is valid. Determining whether a given string is a solution to a given Avoid instance is a coNP-complete problem, so an efficient algorithm for this would imply that $\mathsf{P} = \mathsf{NP}$. Notice that under this definition, it is easy to find examples of search problems that are trivially in FP but also trivially not in TFNP or even FNP. (For example, on input x, either output 0 or output a Turing machine that halts on input x.) So, the fact that Avoid is not in TFNP unless $\mathsf{P} = \mathsf{NP}$ really tells us relatively little about whether Avoid is easy.

Separating Avoid **and the Linear Ordering Principle.** Theorem 1.1 separates Avoid from LOP (the Linear Ordering Principle), which was recently defined by Korten and Pitassi [KP24] and is closely related to Avoid. Indeed, [KP24, Problem 2] asked whether LOP can be reduced to Avoid under $\mathsf{FP}^{\mathsf{NP}}$ reductions, as such a reduction would have very interesting complexity-theoretic consequences. Since LOP is known to be hard for MA and even $\mathsf{P}^{\mathsf{prMA}}$ [KP24, HV25], Theorem 1.1 implies that LOP does not reduce to Avoid unless $\mathsf{P}^{\mathsf{prMA}} \subseteq \mathsf{AM} \cap \mathsf{coAM}$. (So, we rule out FP reductions, and even suitable randomized efficient reductions. We do not expect to rule out $\mathsf{FP}^{\mathsf{NP}}$ reductions, since it is thought that LOP $\in \mathsf{FP}^{\mathsf{NP}}$.)

Generalization to a wider class of problems. We briefly note that Theorems 1.1, 1.2 and 1.3 can be generalized slightly. Specifically, these results apply not only to Avoid but to any search problem such that (1) a random string is a solution with non-negligible probability; (2) the problem of recognizing a solution is in coAM; and (3) the probability that a random string is a non-solution can be upper bounded up to small error in AM (which we prove about Avoid in Theorem 1.4). For example, our results apply to the variant of Avoid in which the circuit takes as input an integer in $[N]$ (represented appropriately) and outputs an integer in $[M]$, provided that $M \geq N(1 + 1/\operatorname{poly}(\log N))$,[4] and they also apply to the two harder variants of Avoid defined in [CGL+23]. However, for simplicity, we simply work with Avoid. (Of course, Theorems 1.1, 1.2 and 1.3 also apply immediately to all of the problems that are now known to reduce to Avoid, and particularly the explicit construction problems in [Kor21].)

1.2 Our techniques

1.2.1 "A SearchAMA protocol for Avoid"

As we mentioned above, the high-level idea behind our two main results is a simple "SearchAMA protocol for Avoid." (See Section 2.5 for the formal definition of SearchAMA.)

In the protocol, polynomially bounded Arthur and computationally unbounded Merlin take as input a circuit $C : \{0,1\}^n \to \{0,1\}^{n+1}$. Let us assume for simplicity that Arthur knows $\alpha := |\mathrm{Im}(C)|/2^{n+1}$, i.e., Arthur knows the fraction $\alpha \leq 1/2$ of elements in $\{0,1\}^{n+1}$ that are in the image of the circuit C. (In our actual protocol, we use Theorem 1.4 to allow Merlin to convince Arthur of a good upper bound on α, which is sufficient.)

Arthur will sample $y_1, \ldots, y_m \sim \{0,1\}^{n+1}$ for some suitably large $m := m(n) = \operatorname{poly}(n)$ and send them to Merlin. Notice that Arthur expects roughly αm of the y_i to be in the image of C and roughly $(1-\alpha)m$ of them to lie outside the image. More precisely, the number of y_i in the image of C will be between, say, $\alpha m - \sqrt{nm}$ and $\alpha m + \sqrt{nm}$ with probability at least $1 - 2^{-\Omega(n)}$.

The idea is to have Merlin simply tell Arthur which of the y_i are in $\mathrm{Im}(C)$ by simply providing preimages for these y_i. So, Merlin will respond to Arthur's message with a list $x_1, \ldots, x_m \in \{0,1\}^n$ that is meant to be a complete list of preimages, in the sense that $C(x_i) = y_i$ for every i such that $y_i \in \mathrm{Im}(C)$. (Of course, when $y_i \notin \mathrm{Im}(C)$, we cannot hope to have $C(x_i) = y_i$.)

Arthur simply computes the number $k := |\{i \; : \; C(x_i) = y_i\}|$ of actual preimages provided by Merlin. If $k < \alpha m - \sqrt{nm}$, then Arthur assumes that Merlin has not in fact provided a complete list of preimages, and Arthur simply aborts the protocol by outputting the special symbol $\perp$. Otherwise,

[4] This problem is known to be equivalent to Avoid under $\mathsf{FP}^{\mathsf{NP}}$ reductions [KKMP21], but it is not known to be equivalent under efficient reductions without an NP oracle.

Arthur samples a uniformly random index $i \sim \{i' : C(x_{i'}) \neq y_{i'}\}$ and outputs y_i. In other words, Arthur outputs a uniformly random element y_i from the list of elements for which Merlin has *not* provided a preimage.

Notice that if the number of elements y_i in $\mathrm{Im}(C_i)$ is between $\alpha m - \sqrt{nm}$ and $\alpha m + \sqrt{nm}$ then (1) Arthur will always output a correct answer if Merlin is honest and sends a complete list of preimages; and (2) if Merlin is dishonest, then Arthur will either output $\perp$, or Arthur will output a correct answer with probability

$$\frac{|\{i' : y_{i'} \notin \mathrm{Im}(C)\}|}{|\{i' : y_{i'} \neq C(x_{i'})\}|} \geq \frac{(1-\alpha)m - \sqrt{nm}}{(1-\alpha)m + \sqrt{nm}} \geq 1 - 4\sqrt{n/m} \ .$$

So, honest Merlin causes Arthur to output a correct answer with very high probability (probability at least $1 - 2^{-\Omega(n)}$), and no matter what Merlin does, with probability at least $1 - 4\sqrt{n/m} - 2^{-\Omega(n)}$, Arthur will output either a solution or $\perp$. We can make the latter probability larger than $1 - \varepsilon$ for any $\varepsilon := \varepsilon(n) \geq 1/\operatorname{poly}(n)$ by taking $m \gg n/\varepsilon^2$ to be a sufficiently large polynomial.

So, this protocol allows Arthur to "interact with Merlin to drastically lower the failure probability of the trivial algorithm for Avoid from $1/2$ to any $\varepsilon = 1/\operatorname{poly}(n)$." See Section 4 for the formal statement. This simple protocol is a first hint that perhaps there is something interesting about range avoidance's relationship with Arthur-Merlin protocols.

Before moving on, we wish to draw attention to one rather annoying technicality about the above protocol. Formally, it is an AMA protocol, and *not* an AM protocol. Recall that in an AM protocol, Arthur sends a uniformly random message to Merlin, Merlin responds, and then Arthur must decide what to output *deterministically based only on these two messages and the input.* In an AMA protocol, Arthur is allowed to flip additional random coins before deciding what to output. It is known that any AMA protocol *for a decision problem* can be efficiently converted generically into an AM protocol [Bab85, BM88] (and, in fact, any constant-round protocol can be converted into a two-round AM protocol). But, for search problems we do not know how to convert an AMA protocol into an AM protocol (perhaps because no such conversion is possible).

1.2.2 Reducing Avoid to a problem in TFNP

To reduce Avoid to a problem in TFNP, our idea is to replace the random strings $y_1, \ldots, y_m \in \{0,1\}^{n+1}$ from the AM protocol described above by a fixed list of suitably *pseudorandom* strings. In particular, suppose that for every n and every circuit size s there exists some canonical list of strings $y_1, \ldots, y_m \in \{0,1\}^{n+1}$ that can be computed in time $\operatorname{poly}(n, s)$ such that for every circuit $C : \{0,1\}^n \to \{0,1\}^{n+1}$ of size at most s, we have

$$\alpha m - \varepsilon m \leq |\{i : y_i \in \mathrm{Im}(C)\}| \leq \alpha m + \varepsilon m$$

where $\alpha := |\mathrm{Im}(C)|/2^{n+1}$ as above and, say, $\varepsilon \leq 1/n^{10}$. In other words, the number of elements y_i in the image of C is always close to what we expect.

Then, we can try to define a total search problem in which the input is a circuit $C : \{0,1\}^n \to \{0,1\}^{n+1}$, and the goal is to output x_i such that $C(x_i) = y_i$ for at least $\alpha m - \varepsilon m$ different choices of i. This is in fact a total search problem by the above assumption that the number of y_i in the image is at least $\alpha m - \varepsilon m$. And, just like in the protocol described above, if we choose a random element for which we have *not* received a preimage, it will be a solution to the Avoid instance with probability at least $1 - O(\varepsilon)$.

As described, however, this problem is probably *not* in TFNP. The problem is that it is not clear how to verify that the number of preimages that we have received is in fact larger than $\alpha m - \varepsilon m$, since we do not know α! For this, we modify the search problem slightly by also asking for the solution to include a proof π that $\alpha := |\mathrm{Im}(C)|/2^{n+1}$ is actually small (specifically, small relative to the fraction of preimages provided).

It remains to show how to construct (1) a list $y_1, \ldots, y_m \in \{0,1\}^{n+1}$ of such pseudorandom strings; and (2) a proof π that the range size of a circuit is small. In fact, it is relatively easy to see that such a list of pseudorandom strings can be constructed if there exists a PRG that fools $\mathsf{NP}_{||}$ circuits. (See Lemma 2.9.) Furthermore, the existence of such a PRG implies that $\mathsf{AM} = \mathsf{NP}$ [KvM99]. Therefore, if such a PRG exists, then the range-size upper bound AM protocol in Theorem 1.4 can be converted into an NP proof π with the desired property. Indeed, such strong PRGs are known to exist under the assumption that E does not have $2^{o(n)}$-size non-deterministic circuits [SS24]. See Sections 2.3 and 6 for more detail.

Putting all of this together, we obtain a problem that *is* in TFNP (assuming such strong PRGs exist, which is in turn implied by circuit lower bounds against non-deterministic circuits) and a randomized reduction from Avoid to this problem. Notice that the reason that the reduction is randomized is more-or-less exactly the same as the reason that the protocol that we described in Section 1.2.1 is an AMA protocol rather than an AM protocol.

1.2.3 An $\mathsf{AM} \cap \mathsf{coAM}$ protocol for decision problems that reduce to Avoid

We now describe our proof that any decision problem that reduces to Avoid must be in $\mathsf{AM} \cap \mathsf{coAM}$. The high-level idea behind the proof is again the simple protocol for Avoid described in Section 1.2.1. Recall that in this protocol, given some circuit $C : \{0,1\}^n \to \{0,1\}^{n+1}$ with approximate image size $\alpha \approx |\mathrm{Im}(C)|/2^{n+1}$ (α should be confirmed with the protocol from Theorem 1.4, but we ignore this for now), Arthur sends Merlin m uniformly random strings $y_1, \ldots, y_m \sim \{0,1\}^{n+1}$, and Merlin responds with $x_1, \ldots, x_m \in \{0,1\}^n$ such that $C(x_i) = y_i$ for roughly αm indices i. Arthur then samples a uniformly random j from the set $\{i \ : \ y_i \neq C(x_i)\}$ of indices for which he has *not* received a preimage and outputs y_j.

So, suppose that L is some decision problem that reduces to Avoid. In other words, suppose that there exists some polynomial-time oracle algorithm $\mathcal{B}^{\mathcal{O}}$ such that if $\mathcal{O}$ solves Avoid then $\mathcal{B}^{\mathcal{O}}(x) = 1$ if and only if x is a YES instance. (For simplicity, we are being slightly less general here. In particular, we are assuming that $\mathcal{B}$ is deterministic, and we are assuming that L is a true decision problem, rather than a promise decision problem. In Section 5, we handle randomized reductions and promise problems as well.)

It is natural to then try to convert $\mathcal{B}^{\mathcal{O}}$ into an $\mathsf{AM} \cap \mathsf{coAM}$ protocol[5] for L by simply replacing all oracle calls made by $\mathcal{B}^{\mathcal{O}}$ with the protocol for Avoid from Section 1.2.1. In more detail, we can consider the protocol in which Arthur simulates $\mathcal{B}^{\mathcal{O}}$. Each time that $\mathcal{B}^{\mathcal{O}}$ makes an oracle call on some circuit $C : \{0,1\}^n \to \{0,1\}^{n+1}$, Arthur sends $y_1, \ldots, y_m \sim \{0,1\}^{n+1}$ to Merlin, receives an appropriate response $x_1, \ldots, x_m \in \{0,1\}^n$ from Merlin, and then chooses a uniformly random y_j for which he did not receive a preimage. Arthur then continues running $\mathcal{B}^{\mathcal{O}}$, taking y_j as the output of the oracle on input C. When he encounters the next oracle query, Arthur engages in another such protocol with Merlin with the new circuit corresponding to the new query.

[5] Here, we mean an AM protocol with the property that honest Merlin will cause Arthur to output the right answer (either YES or NO) with high probability on any input, and regardless of Merlin's behavior, with high probability Arthur will either output the right answer or $\perp$. Having such a protocol is equivalent to lying in $\mathsf{AM} \cap \mathsf{coAM}$.

This *will* yield a correct and sound protocol for L, but the resulting protocol will have roughly $2q$ rounds, where q is the number of oracle queries made by $\mathcal{B}^{\mathcal{O}}$. Since we want to handle reductions that make polynomially many oracle queries, this yields a protocol with polynomially many rounds, which is not useful. (Indeed, any language in PSPACE has an Arthur-Merlin protocol with polynomially many rounds [Sha92]! So, this argument would only allow us to prove that L is in PSPACE, which is trivial.)

We could of course try to lower the number of rounds by running some or all of the interactions with Merlin in parallel. If the queries made by $\mathcal{B}^{\mathcal{O}}$ are non-adaptive (i.e., if each query depends only on the input and not on the responses of the oracle to earlier queries), then this will work. But, if each query made by $\mathcal{B}^{\mathcal{O}}$ can depend on the answers to the previous queries, then Merlin will need to know which response y_j is chosen by Arthur as the simulated oracle response to one query before he can send the preimages x_i to help Arthur respond to the next query. So, it seems that we must either run the queries in series (in which case we obtain a useless protocol); or we need to somehow provide Merlin with enough information to compute y_j himself, which seems like it would violate soundness because a dishonest Merlin who knows how we will choose the index j in advance can easily cause us to pick an invalid y_j.

Notice that the fundamental issue here is again the fact that the protocol from Section 1.2.1 is an AMA protocol, rather than an AM protocol. We get around this issue as follows.

We build a protocol in which Arthur and Merlin take as input *many* circuits $C_1, \ldots, C_\ell : \{0,1\}^n \to \{0,1\}^{n+1}$, which we think of as ℓ instances of Avoid (together with estimates $\alpha_1, \ldots, \alpha_\ell$ with $\alpha_i \approx |\mathrm{Im}(C_i)|/2^{n+1}$, which we ignore here). In this new protocol, Arthur samples many strings $y_{i,j} \sim \{0,1\}^{n+1}$ for $1 \le i \le \ell$ and $1 \le j \le m$ for some suitable m and sends them to Merlin. Merlin now responds with many strings $x_{i,j} \in \{0,1\}^n$ purporting to contain a preimage for each $y_{i,j}$ that has one, and Arthur then *deterministically* outputs some of these strings $y_{1,j_1}, \ldots, y_{\ell,j_\ell}$ in a very careful way. Specifically, Arthur chooses j_i to simply be the minimal index j for which $C_i(x_{i,j}) \neq y_{i,j}$. Arthur then performs a rather careful check on the preimages provided to see if he suspects Merlin of cheating. (Specifically, Arthur will check whether for every index j, Merlin has provided notably less than the expected number of preimages for $y_{1,j}, \ldots, y_{\ell,j}$.) If so, he outputs $\perp$. Otherwise, he outputs $y_{1,j_1}, \ldots, y_{\ell,j_\ell}$. In particular, notice that this is an AM protocol, rather than an AMA protocol.

Arthur's acceptance criteria are such that if Merlin is honest, then with high probability over the $y_{i,j}$, Arthur will not abort and the y_{i,j_i} will be solutions to the respective Avoid instances, i.e., for all i, $y_{i,j_i} \notin \mathrm{Im}(C_i)$. However, for any dishonest Merlin, with high probability over the $y_{i,j}$, either Arthur will abort or Arthur will output $y_{1,j_1^\dagger}, \ldots, y_{\ell,j_\ell^\dagger}$ with the property that $j_i = j_i^\dagger$ for at least, say, $\ell - m^c\sqrt{\ell}$ values of i. (We will take $\ell \gg m^{2c}$, so that this is a large fraction of the $i \in [\ell]$.) In other words, while a dishonest Merlin can cause Arthur to output a few different strings, nearly all of Arthur's output is unaffected by Merlin's message (unless, of course, Arthur aborts).

So, compared to the much simpler protocol from Section 1.2.1, we have paid a bit of a price in that we now have ℓ different Avoid instances and we are only likely to solve $\ell - m^c\sqrt{\ell}$ of them. But, for this price, we have (1) gotten rid of the final random coins generated by Arthur; and (2) greatly constrained Merlin's ability to control Arthur's output.

We then use this protocol to obtain a protocol for L as follows. We imagine running ℓ copies of $\mathcal{B}^{\mathcal{O}}$ in parallel. Each time that these ℓ parallel reductions make ℓ parallel queries $C_1, \ldots, C_\ell$ to the Avoid oracle, we run the above protocol to receive ℓ responses $y_{1,j_1}, \ldots, y_{1,j_\ell} \in \{0,1\}^{n+1}$, which we treat as the oracle's responses to the respective queries.

This might seem like an unnecessarily complicated variant of the same idea, but notice that because y_{i,j_i} only depend on Arthur's initial message and Merlin's response (and *not* on any additional random coins), Merlin can determine the next circuit queried just based on Arthur's first message. It follows that all of these subprotocols can be run in parallel. This therefore yields a two-round protocol.

And, the soundness guarantee on our subprotocols means that for each of the q queries made by $\mathcal{B}^{\mathcal{O}}$, out of the ℓ responses generated by our protocol, at most $m^c\sqrt{\ell}$ of them will *not* be solutions to their respective Avoid instance (with high probability, assuming that Arthur does not abort). Therefore, the total number of wrong answers that we receive cumulatively in all ℓ parallel simulations is bounded by $qm^c\sqrt{\ell}$. If we take $\ell \gg m^{2c}q^2$, then $qm^c\sqrt{\ell} \ll \ell$, and it follows that the majority of our parallel simulations will terminate with the right answer.

In fact, our subprotocol is sufficiently robust that we can even handle randomized reductions and promise problems, with very little additional work. In more detail, we are able to handle randomized reductions precisely because of the fact that when Arthur does not abort, most of the y_{i,j_i} will be the same regardless of Merlin's answer.

1.2.4 A range-size upper bound protocol

Finally, we describe the range-size upper bound protocol from Theorem 1.4. In other words, we show a protocol in which Merlin can convince Arthur that $|\mathrm{Im}(C)|$ is *small* for some circuit $C : \{0,1\}^n \to \{0,1\}^m$.

For this, we recall the celebrated Goldwasser-Sipser set-size *lower bound* protocol [GS86]. This protocol allows Merlin to convince Arthur that a set $S \subseteq \{0,1\}^m$ is *large*, provided that membership in S can be recognized in AM, i.e., provided that Merlin can convince Arthur that $x \in S$ whenever x is in fact in S. (The Goldwasser-Sipser protocol is commonly described as working for sets S that are recognizable in NP, but it naturally extends to AM and even to prAM. We include a proof in Section 3.1 for completeness.)

So, in order to build our protocol, we would like some way to reduce the problem of showing that the image of a circuit is *small* to the problem of showing that certain recognizable sets are *large*. To do so, we first note a simple double-counting identity, which says that

$$|\mathrm{Im}(C)| = 2^n - \sum_{i=2}^{2^n} |S_i| \ ,$$

where

$$S_i := \{y \in \{0,1\}^m \ : \ |C^{-1}(y)| \geq i\} \ .$$

In other words, S_i is the set of all elements with at least i preimages. This identity shows that in order to prove that $|\mathrm{Im}(C)|$ is small, it suffices to show that the sum is large.

We then "apply Goldwasser-Sipser twice." In particular, notice that membership in $C^{-1}(y)$ can be recognized in $\mathsf{NP} \subseteq \mathsf{AM}$ (i.e., given x and y, one can simply check if $C(x) = y$ to determine whether $x \in C^{-1}(y)$). So, by Goldwasser-Sipser, it follows that membership in S_i can be recognized in AM.[6] Therefore, by applying Goldwasser-Sipser again, we can obtain a lower bound on $|S_i|$.

[6] Here, we have ignored the fact that Goldwasser-Sipser only allows us to *approximately* estimate set size. (Computing such set sizes exactly is of course #P-complete.) This means that Goldwasser-Sipser only lets us distinguish elements that are in S_i from elements that are not even in $S_{(1-1/\operatorname{poly}(n))i}$. This leads to some additional technicalities, but the high-level idea does not change.

Finally, we show that we can replace the sum $\sum_{i=2}^{2^n} |S_i|$ by a much coarser sum

$$\sum_{j=1}^{\mathrm{poly}(n)} \beta_j |S_{i_j}|$$

for carefully chosen indices i_j and weights β_j without incurring much error.

Putting everything together, we can use the Goldwasser-Sipser protocol polynomially many times (in parallel) to obtain lower bounds on S_{i_j} for a suitable choice of polynomially many indices i_j, which in turn yields an upper bound on $|\mathrm{Im}(C)|$.

1.3 Related work

There is already a large body of literature on Avoid. Here, we describe some of the work that is most closely related to the present work.

Korten and Santhanam showed that under very strong derandomization assumptions, certain *uniform* versions of Avoid can be solved by efficient deterministic algorithms [KS25a]. In more detail, they show that for any poly(n)-time computable family of circuits $C_n : \{0,1\}^n \to \{0,1\}^m$, there is an efficient deterministic algorithm that solves Avoid on C_n, where depending on the strength of the derandomization assumption their algorithm works for either $m = \mathrm{poly}(n)$, $m = n \cdot \mathrm{poly}(\log n)$, or $m = n + o(n)$. Our Theorem 1.2 requires a weaker derandomization assumption, and it solves a harder problem in that it works for any input (rather than just poly(n)-time computable families) and for $m = n + 1$ (the hardest choice). (Indeed, the problem that we solve is widely believed to *not* be solvable by efficient deterministic algorithms.) But, our algorithm is randomized and requires access to an oracle for a problem that is in TFNP. In short, we place a weaker upper bound on a harder computational problem under a weaker computational assumption.

Our proof of Theorem 1.2 is heavily inspired by Korten and Santhanam's work. Indeed, [KS25a] similarly uses a suitably pseudorandom string to effectively derandomize the basic idea of using a random string as a solution to Avoid. However, in [KS25a], they need to argue that a single sufficiently pseudorandom string can itself be used as a solution to an Avoid instance consisting of a sufficiently expanding *uniform* circuit, while we need the much simpler observation that a sufficiently pseudorandom *list* of strings must contain many solutions to any Avoid instance (of a given size). (The technical difficulty in proving Theorem 1.2 is in showing how to use this pseudorandom list of candidate solutions to reduce Avoid to a problem in TFNP.)

To the authors' knowledge, prior to the present work the only result showing a barrier to proving hardness of Avoid was the following simple observation in [ILW23, CGL$^+$23]. (To be clear, this observation was not the focus of either [ILW23, CGL$^+$23].) Suppose that SAT reduces to Avoid via a reduction that makes q queries to the Avoid oracle on circuits $C : \{0,1\}^n \to \{0,1\}^{n+s}$ with $q \ll 2^s$. Then, we can conclude that SAT $\in \mathsf{RP}$ (which implies that $\mathsf{NP} = \mathsf{RP}$), since a randomized algorithm can simply simulate solutions to the oracle queries with random strings and succeed with probability at least $1 - q2^{-s}$. Both works asked whether any barriers can be shown to proving NP-hardness of Avoid with a larger number of queries q, smaller stretch s, or both. We resolve this by showing such a barrier for any (polynomial) number of queries and minimal stretch $s = 1$.

Ilango, Li, and Williams [ILW23] Chen and Li [CL24], and Ren, Wang, and Zhong [RWZ25] showed that Avoid is hard under strong cryptographic and complexity-theoretic assumptions. [CL24, RWZ25] even showed that Avoid $\notin \mathsf{SearchNP}$ under strong cryptographic assumptions (even for restricted classes of input circuits). These proofs are beautiful but rather indirect, and it is natural to ask whether we can simply show a reduction from a presumed hard problem to Avoid, ideally an NP-hard problem. Our Theorem 1.1 effectively rules this out by showing that any reduction from an NP-hard problem to Avoid would collapse the polynomial hierarchy. The results of [CL24, RWZ25] also show that the randomness in Theorem 1.2 is necessary, since any problem that reduces to TFNP deterministically must lie in $\mathsf{SearchNP}$.

Like us, Chen, Li, and Liang also provide "an Arthur-Merlin protocol for Avoid" [CLL25]. However, for their application, they must construct a *single-valued* protocol (i.e., a protocol in which Arthur is likely to output some canonical solution, regardless of any randomness or what Merlin says). To obtain a single-valued protocol, they require non-uniform advice and quasi-polynomial time, and they only solve *uniform* Avoid instances for infinitely many different input lengths. ([CLL25] then uses this protocol to prove breakthrough circuit lower bounds in $\mathsf{AMEXP}_{/2^{n^\varepsilon}}$!) We do not know of any direct relationship between their protocol and ours, but we do note that both protocols rely on the same fundamental properties of Avoid—that a random string is a solution with decent probability and that a solution can be verified in coAM (in fact, solutions to Avoid can be verified in coNP, but both protocols work with the weaker property of verifiability in coAM).

[CGL$^+$23] showed how to build cryptography under the assumption that no efficient randomized algorithm can solve Avoid with probability $1 - 1/\operatorname{poly}(n)$. Our Theorems 1.2 and 1.3 show that Avoid can be solved with this probability with access to a TFNP oracle or via interaction with an all-powerful prover Merlin, respectively. This shows that the problem of solving Avoid with probability $1 - 1/\operatorname{poly}(n)$ in some sense lives rather low in the polynomial hierarchy, similar to nearly all known problems whose hardness implies cryptography.

Finally, we note that Fortnow [For87] presented a protocol that is similar to our range-size upper bound protocol from Theorem 1.4. Formally, Fortnow gave a protocol that allows Merlin to convince Arthur of an upper bound on the size of a set S, provided that Arthur has the ability to sample a random element from S. For example, if the circuit C is somehow known to be regular (i.e., if every element in the image has the same number of preimages), then Fortnow's protocol allows Merlin to convince Arthur of an upper bound on the size of the image (in fact, with multiplicative error, whereas we only achieve additive error). (We thank Korten for bringing this similarity to our attention.)

1.4 Future directions

We briefly list some possible future directions.

First, recall that we show that $\mathsf{BPP}^{\mathrm{Avoid}} \subseteq \mathsf{AM} \cap \mathsf{coAM}$. In other words, any decision problem that reduces to Avoid must be in $\mathsf{AM} \cap \mathsf{coAM}$. This effectively rules out NP-hardness of Avoid, but it does not rule out the possibility of a reduction from some presumed hard decision problem in $\mathsf{AM} \cap \mathsf{coAM}$ to Avoid. (E.g., cryptographers study many problems that are thought to be hard but are in $\mathsf{AM} \cap \mathsf{coAM}$ and even in much smaller classes.) So, perhaps we can still reduce a well-studied presumed hard problem to Avoid.

Alternatively, perhaps $\mathsf{BPP}^{\text{Avoid}}$ or ($\mathsf{P}^{\text{Avoid}}$) lies in an even smaller complexity class. Indeed, since there is no known reduction from a presumed hard decision problem to Avoid, we see no fundamental barrier to proving that decision problems that reduce to Avoid lie in much smaller complexity classes than $\mathsf{AM} \cap \mathsf{coAM}$.[7] Perhaps a natural first subclass of $\mathsf{AM} \cap \mathsf{coAM}$ to consider is SZK, though our current protocols are certainly not SZK protocols.

In a different direction, recall that we have shown that there is a randomized reduction from Avoid to a problem in TFNP (under a complexity-theoretic assumption). Chen and Li's proof that Avoid $\notin$ $\mathsf{SearchNP}$ (under cryptographic assumptions) shows that we are unlikely to derandomize this reduction. However, we can hope to improve on it in a different direction. In particular, the specific problem $\mathrm{T} \in \mathsf{TFNP}$ that we reduce to is rather artificial. So, we ask whether there is a (necessarily randomized) reduction from Avoid to a more natural problem in TFNP. For example, as was already observed in [KKMP21], Avoid seems closely related to the subclass $\mathsf{PWPP} \subseteq \mathsf{TFNP}$. Informally, PWPP is the complexity class that captures "computational problems whose totality is proven via the (weak) pigeonhole principle" while Avoid corresponds to "computational problems whose totality is proven via the *dual* (weak) pigeonhole principle." It is quite natural to ask whether there is a reduction between Avoid and PWPP in either direction or whether one can show an oracle separation. The same problem is interesting for any of the well-studied subclasses of TFNP.

Finally, we note that the range-size upper bound protocol described in Theorem 1.4 seems like quite a general tool that complements the celebrated Goldwasser-Sipser protocol [GS86]. We therefore ask whether it has additional applications. (In particular, our protocol works for any circuit and does not require the circuit to be expanding.)

Acknowledgments

The authors would like to thank Huck Bennett, Sasha Golovnev, and Bobby Kleinberg for helpful discussions. We also thank Oliver Korten for comments on an earlier version of this work. We acknowledge the use of ChatGPT for identifying typos and for help with Figure 1.

2 Preliminaries

2.1 Search problems

A search problem is different from a decision problem in that it may have a multi-bit output and some (or all) inputs may have multiple possible solutions. Search problems are therefore formalized as relations.

Definition 2.1. *A search problem is a relation* $\mathrm{R} \subseteq \{0,1\}^* \times \{0,1\}^*$. *We say that* y *is a solution to* R *on input* x *if* $(x, y) \in \mathrm{R}$. *We write* $\mathrm{R}(x)$ *to denote* $\{y : (x, y) \in \mathrm{R}\}$.

An important complexity class in this context is TFNP, which consists of search problems for which (1) there always exists a solution; and (2) solutions are efficiently checkable.

Definition 2.2. *A relation* R *is in* TFNP *if the following conditions hold:*

[7] As far as we know, it is even plausible that $\mathsf{BPP}^{\text{Avoid}} = \mathsf{BPP}$, even if Avoid itself is hard. For example, letting Avoid_m be the version of Avoid in which the input circuit maps n bits to $m := m(n)$ bits, $\mathsf{BPP}^{\text{Avoid}_m} = \mathsf{BPP}$ for any $m > n + \omega(\log n)$ [ILW23, CGL$^+$23], but it is still certainly plausible (perhaps likely) that $\text{Avoid}_m \notin \mathsf{FP}$. Similarly, the problem of outputting an n-bit string with large Kolmogorov complexity is provably uncomputable (and therefore certainly not in FP), but for suitable parameters any decision problem that reduces to it is again in BPP.

1. R *is total: for all* $x \in \{0,1\}^*$*, there exists* $y \in \{0,1\}^*$ *such that* $(x,y) \in \mathrm{R}$*.*

2. R *is polynomial: For all* $(x,y) \in \mathrm{R}$*,* $|y| \leq \mathrm{poly}(|x|)$*.*

3. *There exists a polynomial time Turing machine* $\mathcal{M}$ *such that* $(x,y) \in \mathrm{R}$ *if and only if* $\mathcal{M}(x,y) = 1$*.*

While the notion of a reduction from one decision problem to another is clear, what it means for one *search* problem to reduce to another can be rather tricky since search problems may have multiple solutions. The situation becomes more complicated still when the reduction is a randomized Turing reduction. We will be somewhat informal about this (as is standard in the literature) and simply consider the strongest type of randomized reduction $\mathcal{M}^{\mathrm{B}}$ from A to B that succeed with probability 2/3 whenever the responses of $\mathcal{O}$ to queries are solutions to B. The weaker type is where the oracle $\mathcal{O}$ is fixed in advance and the reduction should work for all *fixed* oracles $\mathcal{O}$ solving B. We will not dwell on this distinction except to say that our results always hold for whichever type of reduction gives the stronger result. In particular, the results in Section 5 hold even when we work with the weaker notion of randomized Turing reductions to Avoid and the results in Section 6 hold even when we work with the stronger notion of randomized Turing reductions to a search problem.

2.2 Avoid

The following useful lemma tells us that we can always "upward self-reduce" Avoid simply by padding the input and output.

Lemma 2.3. *There is a (simple) linear-time, deterministic, single-query reduction from* Avoid *on a circuit* $C : \{0,1\}^n \to \{0,1\}^{n+1}$ *with* $|C| = s$ *to* Avoid *on a circuit* $C' : \{0,1\}^{n'} \to \{0,1\}^{n'+1}$ *where* $|C'| \leq {n'}^2$ *for any* $n' > s$*.*

Proof. The reduction constructs a circuit $C' : \{0,1\}^n \times \{0,1\}^{n'-n} \to \{0,1\}^{n'+1}$ such that $C'(x,x') = C(x) \circ x'$. It is not hard to see that $|C'| \leq s + O(n') = O(n')$ which is less than ${n'}^2$ for sufficiently large s. We feed C' to our range avoidance oracle to get back an avoided element $y \in \{0,1\}^{n'+1}$. We decompose $y = y_1 \circ y_2$ such that $y_1 \in \{0,1\}^{n+1}$ and output y_1. The reduction clearly runs in $O(s + n')$ time since all we need to do to construct C' is pad C with $n' - n$ extra input and output bits. To see that y_1 is not in the range of C, simply observe that if it were and had preimage x_1 in C, then $y_1 \circ y_2$ would be in the range of C' since $C'(x_1, y_2) = C(x_1) \circ y_2 = y_1 \circ y_2$. □

2.3 Pseudorandom generators against $\mathsf{NP}_{||}$ circuits

One of the key assumptions that we will make is the existence of pseudorandom generators against $\mathsf{NP}_{||}$ circuits. We first define non-deterministic circuits and pseudorandom generators.

Definition 2.4. *A non-deterministic circuit* $C : \{0,1\}^n \to \{0,1\}$ *is a circuit which has along with its usual* n *bit input,* m *non-deterministic bits as input. We say that the circuit evaluates to* 1 *on input* $x \in \{0,1\}^n$ *if there exists an assignment of the* m *non-deterministic bits that causes the circuit to output* 1*, and* 0 *otherwise.*

We note that non-deterministic circuits are different from $\mathsf{NP}_{||}$ circuits, which is how one may initially think to formalize non-deterministic circuits.

Definition 2.5. *An* $\mathsf{NP}_{||}$ *oracle circuit is a circuit with oracle gates to* SAT *where all oracle queries are made in parallel (i.e. non-adaptively).*

Definition 2.6. *A distribution X on n bits is ε-pseudorandom for a class $\mathcal{C}$ of functions, if for every $C \in \mathcal{C}$,*

$$\left| \Pr_{y \sim X}[C(y) = 1] - \Pr_{y \sim \{0,1\}^n}[C(y) = 1] \right| \leq \varepsilon.$$

A function $G : \{0,1\}^d \to \{0,1\}^n$ is an ε-PRG for $\mathcal{C}$ if $G(X)$ is ε-pseudorandom for $\mathcal{C}$ where $X \sim \{0,1\}^d$ is a uniformly random seed.

Our key assumption will be the existence of PRGs against $\mathsf{NP}_{||}$ circuits. This assumption can itself be based on the plausible assumption that E (the set of problems solvable in $2^{O(n)}$ time) requires exponential size non-deterministic circuits. One should view this as a variant of the classic result of Impagliazzo and Wigderson that if E requires exponential size circuits, then we have PRGs good enough to get $\mathsf{BPP} = \mathsf{P}$ [IW97], and it follows from a long line of work [IW97, KvM99, SU06, SS24]. The precise version that we use is from [SS24, Theorem 2.8].

Assumption 2.7. *For every constant $c > 1$, there exists a constant $\alpha > 1$ such that for every sufficiently large m, there is a $G : \{0,1\}^{\alpha \cdot \log m} \to \{0,1\}^m$ that is a $\frac{1}{m^c}$-PRG for $\mathsf{NP}_{||}$ circuits of size m^c. Furthermore, G is computable in time* $\mathrm{poly}(m)$.

Theorem 2.8 ([IW97, KvM99, SU06, SS24])**.** *If* E *requires exponential size non-deterministic circuits, then Assumption 2.7 holds.*

We now prove the following useful lemma which tells us that the output $G(1), \ldots, G(k)$ of a PRG against $\mathsf{NP}_{||}$ circuits will contain roughly as many solutions to an Avoid instance as a random list of strings.

Lemma 2.9. *Let $G : \{0,1\}^{\alpha \cdot \log(n+1)} \to \{0,1\}^{n+1}$ be a $\frac{1}{n^3}$-PRG which is secure against $\mathsf{NP}_{||}$ circuits of size at most n^3. Let $C : \{0,1\}^n \to \{0,1\}^{n+1}$ be a circuit of size at most n^2. Then*

$$\left| \Pr_{i \sim [(n+1)^\alpha]}[G(i) \in \mathrm{Im}(C)] - \frac{|\mathrm{Im}(C)|}{2^{n+1}} \right| \leq \frac{1}{n^3}.$$

Proof. Let $D : \{0,1\}^{n+1} \to \{0,1\}$ be the $\mathsf{NP}_{||}$ circuit which on input $y \in \{0,1\}^{n+1}$ outputs 1 if there exists an x such that $C(x) = y$ and outputs 0 otherwise. There is the minor subtlety that writing the statement $\exists x, C(x) = y$ as a SAT instance incurs some blowup, so D may be slightly larger than C. However, we can safely bound the blowup so that $|D| \leq n^3$.

Notice also that

$$\Pr_{y \sim \{0,1\}^{n+1}}[D(y) = 1] = \frac{|\mathrm{Im}(C)|}{2^{n+1}} .$$

The result follows by observing that G is a $\frac{1}{n^3}$-PRG which fools size n^3 circuits, including D. □

2.4 Checkability

Intuitively, a relation R is checkable if we have an algorithm which, when given oracle access to an oracle $\mathcal{O}$ which claims to solve R, our algorithm either solves R or correctly reports that $\mathcal{O}$ in fact does not solve R. Checkability is a notion originally defined in [BK95].

For a relation $\mathrm{R} \subseteq \{0,1\}^* \times \{0,1\}^*$, we say that $y \in \{0,1\}^* \cup \{\perp\}$ is a *solution* to $\mathrm{R}(x)$ if $y \in \mathrm{R}(x)$ *or* if $\mathrm{R}(x) = \varnothing$ and $y = \perp$.

Definition 2.10 ([BK95, MX10])**.** *Let* $\mathrm{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ *be a relation.* R *is checkable if there exists an efficient randomized oracle algorithm* $\mathcal{A}$ *such that for all oracles* $\mathcal{O}$*, the following holds.*

- **Completeness***: Suppose that for all* x*,* $\mathcal{O}(x)$ *is a solution to* $\mathrm{R}(x)$*. Then for all* x*,* $\mathcal{A}^{\mathcal{O}}(x)$ *is also a solution to* $\mathrm{R}(x)$*.*
- **Soundness***: For any* $\mathcal{O}$ *and* x*, with overwhelming probability* $\mathcal{A}^{\mathcal{O}}(x) \in \mathrm{R}(x) \cup \{\perp\}$*.*

We are particularly interested in the checkability of SAT. The question of whether SAT is checkable was first proposed in [BK95] and remains open to this day. It seems there is no consensus on whether SAT should be checkable or not. However, [MX10] showed that a randomized Turing reduction from SAT to any TFNP problem would imply checkability of SAT, thereby giving a barrier against showing such a reduction.

Theorem 2.11 ([MX10])**.** *If* $\mathrm{SAT} \in \mathsf{BPP}^{\mathsf{TFNP}}$*, then* SAT *is checkable.*

2.5 Arthur-Merlin protocols

Crucial to our results will be the notion of Arthur-Merlin protocols [Bab85]. Informally, we say that a language $\mathrm{L} \subseteq \{0,1\}^*$ has a k-round Arthur-Merlin protocol if there is a k-round game between a prover $\mathcal{M}$ (Merlin) and a verifier $\mathcal{A}$ (Arthur) where if $x \in \mathrm{L}$, then an honest Merlin can convince Arthur of that fact; and if $x \notin \mathrm{L}$, even a dishonest Merlin cannot convince Arthur that $x \in \mathrm{L}$. The protocol begins with Arthur sampling some randomness r_1 and sending it to Merlin, who sends back a message $m_1 = \mathcal{M}(x, r_1)$ which only depends on the randomness sampled so far. Arthur then samples some new randomness r_2 which he sends to Merlin and Merlin sends back a new message $m_2 = \mathcal{M}(x, r_1, m_1, r_2)$. This continues until k messages have been sent in total, at which point $\mathcal{A}$, given access to all the messages sent so far decides if $x \in \mathrm{L}$. We now define 2-round AM, which by [BM88] is known to be equivalent to any k-round AM for any constant k.

Definition 2.12. *We say that a promise problem* $\Pi = (\Pi_{\mathsf{YES}}, \Pi_{\mathsf{NO}})$ *is in* $\mathsf{prAM}_{c(n),s(n)}$ *if there exists a polynomial time algorithm* $\mathcal{A}$ *and some (*$\mathrm{poly}(n)$*-time computable)* $m := m(n) \le \mathrm{poly}(n)$ *such that the following conditions hold.*

- ***Completeness****: there exists a (computationally unbounded) prover* $\mathcal{M}$ *such that if* $x \in \Pi_{\mathsf{YES}}$*, then*
$$\Pr_{r \sim \{0,1\}^m}[\mathcal{A}(x, r, \mathcal{M}(x, r)) = 1] \ge c(n).$$
- ***Soundness****: If* $x \in \Pi_{\mathsf{NO}}$*, then for all (computationally unbounded) provers* $\mathcal{M}^\dagger$*,*
$$\Pr_{r \sim \{0,1\}^m}[\mathcal{A}(x, r, \mathcal{M}^\dagger(x, r)) = 1] \le s(n).$$

We say that a promise problem Π *is in* prAM *if* Π *is in* $\mathsf{prAM}_{2/3,1/3}$*. A language* $\mathrm{L} \subseteq \{0,1\}^*$ *is in* AM *if* $(\mathrm{L}, \overline{\mathrm{L}})$ *is in* prAM*.*

We will often work with promise problems in AM and use standard properties of Arthur-Merlin protocols which hold in the promise problem setting as well.

Theorem 2.13 ([FGM+89, BGG93, Dru10])**.** *Let* $p(n), q(n)$ *be any polynomials. If a promise problem* Π *is in* $\mathsf{prAM}_{s(n)+1/p(n),s(n)}$*, then* Π *is also in* $\mathsf{prAM}_{1,2^{-q(n)}}$*.*

Theorem 2.14 ([Bab85, GS86, BM88])**.** *Let k be any constant. If a promise problem Π has a k-round Arthur-Merlin protocol, then $\Pi \in \mathsf{prAM}$.*

We will also be interested in Arthur-Merlin-like protocols that solve search problems (namely Avoid). We therefore define $\mathsf{SearchAMA}$.

Definition 2.15. *Let R be a search problem. We say that $\mathrm{R} \in \mathsf{SearchAMA}_{c(n),s(n)}$ if there exists an efficient verifier $\mathcal{A}$ and some (*$\mathrm{poly}(n)$*-time computable) $m := m(n) \leq \mathrm{poly}(n)$ such that the following holds.*

- ***Completeness**: There exists a (computationally unbounded) prover $\mathcal{M}$ with polynomially bounded output length such that for all $x \in \{0,1\}^n$ with $\mathrm{R}(x) \neq \varnothing$,*

$$\Pr_{r_1,r_2\sim\{0,1\}^m}[\mathcal{A}(x,r_1,\mathcal{M}(x,r_1),r_2) \in \mathrm{R}(x)] \geq c(n).$$

- ***Soundness**: For all (computationally unbounded) provers $\mathcal{M}^\dagger$ and all $x \in \{0,1\}^n$,*

$$\Pr_{r_1,r_2\sim\{0,1\}^m}[\mathcal{A}(x,r_1,\mathcal{M}^\dagger(x,r_1),r_2) \in \mathrm{R}(x) \cup \{\bot\}] \geq 1 - s(n).$$

We now highlight a few differences between AM and $\mathsf{SearchAMA}$. The first difference is that, unlike in the case of decision problems in AM, where parallel repetition allows us to boost the success probability, we do not know whether the success probability can be boosted for $\mathsf{SearchAMA}$. So, we parameterize $\mathsf{SearchAMA}_{c(n),s(n)}$ by the completeness $c(n)$ and soundness error $s(n)$ respectively. The second difference is that we allow Arthur randomness after he sees the prover's answer in $\mathsf{SearchAMA}$—this is what makes this $\mathsf{SearchAMA}$ and not $\mathsf{SearchAM}$. In the case of decision problems, this extra randomness is known to be irrelevant (i.e., $\mathsf{AMA} = \mathsf{AM}$ [Bab85, BM88]). It is unclear if one can show a similar result in the case of search Arthur-Merlin protocols.

Finally, we note that under the strong (but standard) derandomization assumption presented in Assumption 2.7, we can derandomize prAM.

Theorem 2.16 ([KvM99])**.** *If Assumption 2.7 holds, then $\mathsf{prAM} = \mathsf{prNP}$.*

2.6 Pairwise independent hashing

We will need a family of pairwise independent hash functions, which we define as follows.

Definition 2.17. *A family of functions $\mathcal{H} = \{h : S \to [M]\}$ is called a family of pairwise independent hash functions if for all $x \neq y \in S$ and all $u, v \in [M]$,*

$$\Pr_{h\sim\mathcal{H}}[h(x) = u \text{ and } h(y) = v] = \frac{1}{M^2}\ .$$

Lemma 2.18. *Let n be an integer and p be a prime. There exists an explicit family of pairwise independent hash functions $\mathcal{H} = \{h : \{0,1\}^n \to [p]\}$ that are computable in time $\mathrm{poly}(n, \log p)$.*

Lemma 2.19. *Let $\mathcal{H} = \{h : [N] \to [M]\}$ be a family of pairwise independent hash functions and let $S \subseteq [N]$ be an arbitrary subset. Then,*

$$\begin{aligned}\Pr_{h\sim\mathcal{H},v\sim[M]}[\exists y \in S, h(y) = v] &\geq \sum_{y\in S}\Pr_{h,v}[h(y) = v] - \sum_{\substack{y,y'\in S\\ y\neq y'}}\Pr_{h,v}[h(y) = v, h(y') = v] \\ &= \frac{|S|}{M} - \frac{|S|(|S|-1)}{2M^2}\ .\end{aligned}$$

2.7 The Chernoff-Hoeffding bound

We will need the following version of the Chernoff-Hoeffding bound.

Lemma 2.20. *If $X_1, \ldots, X_\ell \in \{0,1\}$ are independent Bernoulli random variables (not necessarily identical) with $\mu := \sum_{i=1}^{\ell} \mathbb{E}[X_i]$, then for any $m \geq 1$,*

$$\Pr\left[\Big|\sum_{i=1}^{\ell} X_i - \mu\Big| \geq \sqrt{m\ell}\right] \leq 2^{-\Omega(m)} .$$

3 A range-size upper bound protocol

In this section, we present a key technical result, showing how to upper bound the size of a given circuit's range in AM. To that end, we consider the following promise problem.

Definition 3.1. *For any $\varepsilon := \varepsilon(n) > 0$, the ε-GapRangeSize problem is a promise problem defined as follows: the input consists of a circuit $C : \{0,1\}^n \to \{0,1\}^m$ and a size threshold τ.*

- $\Pi_{\mathsf{YES}} : |\mathrm{Im}(C)| \leq \tau$
- $\Pi_{\mathsf{NO}} : |\mathrm{Im}(C)| > \tau + \varepsilon 2^n$

We will build towards proving the following theorem.

Theorem 3.2. *ε-GapRangeSize is in $\mathsf{prAM} \cap \mathsf{prcoAM}$ for any $\varepsilon := \varepsilon(n) \geq 1/\operatorname{poly}(n)$.*

3.1 The Goldwasser-Sipser protocol

We recall the celebrated Goldwasser-Sipser set-size lower bound protocol from [GS86]. This protocol is typically presented as a protocol in which Merlin can convince Arthur that a certain set S has size at least τ when we in fact have $|S| \geq (1+\delta)\tau$, provided that membership in S can be recognized in NP. We need a slight generalization of this in which we replace NP-recognizable languages with AM-recognizable promise problems, as follows. The original proof in [GS86] immediately applies to this more general setting, but we include our own proof for completeness.

Definition 3.3. *Let $\Pi' = (\Pi'_{\mathsf{YES}}, \Pi'_{\mathsf{NO}})$ be any promise problem where $\Pi'_{\mathsf{YES}}, \Pi'_{\mathsf{NO}} \subseteq \{0,1\}^* \times \{0,1\}^*$. For any $\delta > 0$, the $(1+\delta)$-$\mathrm{GapSize}_{\Pi'}$ problem, parameterized by the promise problem Π' is a promise problem defined as follows on input x, 1^n and a size threshold τ.*

- Π_{YES}*:* $|\Pi'_{\mathsf{YES},n}(x)| \geq (1+\delta)\tau$
- Π_{NO}*:* $2^n - |\Pi'_{\mathsf{NO},n}(x)| \leq \tau$

where $\Pi'_{\mathsf{YES},n}(x) := \{y \in \{0,1\}^n : (x,y) \in \Pi'_{\mathsf{YES}}\}$ and $\Pi'_{\mathsf{NO},n}(x) := \{y \in \{0,1\}^n : (x,y) \in \Pi'_{\mathsf{NO}}\}$

Theorem 3.4 (The Goldwasser-Sipser protocol [GS86])**.** *$(1+\delta)$-$\mathrm{GapSize}_{\Pi'}$ is in prAM for any $\Pi' \in \mathsf{prAM}$ and any $\delta := \delta(n) \geq 1/\operatorname{poly}(n)$.*

Proof. We assume without loss of generality that $\delta \le 1/10$. We show a 4-round protocol achieving an additive $\mathrm{poly}(\delta)$ gap between completeness and soundness. This is sufficient as the probability can be amplified and the number of rounds can be compressed to 2 due to Theorems 2.13 and 2.14.

Since $\Pi' \in \mathsf{prAM}$ by assumption, it follows from Theorem 2.13 that $\Pi' \in \mathsf{prAM}_{1,2^{-n}}$. (I.e., we can boost completeness to 1 and soundness to 2^{-n}.)

Now, on input x, 1^n, and τ, Arthur and Merlin behave as follows.

1. Arthur samples a pairwise independent hash function $h : \{0,1\}^n \to [p]$ and a uniformly random $v \sim [p]$ and sends them to Merlin, where p is the smallest prime satisfying that $p \ge \frac{2}{\delta}\lceil(1+\delta)\tau\rceil$.
2. Merlin replies with an element $y \in \{0,1\}^n$ that is meant to satisfy that $y \in \Pi'_{\mathsf{YES},n}(x)$ and $h(y) = v$.
3. Next, Arthur and Merlin perform the protocol for Π' guaranteed by the fact that $\Pi' \in \mathsf{prAM}_{1,2^{-n}}$ on input (x, y), which takes 2 rounds.
4. Finally, Arthur accepts if and only if both $h(y) = v$ and Arthur accepts in the subprotocol.

Completeness: Let $S := \Pi'_{\mathsf{YES},n}(x)$, and suppose that the input is a YES instance so that $|S| \ge (1+\delta)\tau$. Then, there exists an $S' \subseteq S$ such that $|S'| = \lceil(1+\delta)\tau\rceil$. By Lemma 2.19,

$$\begin{aligned}\Pr_{h,v}[\exists y \in S, h(y) = v] &\ge \Pr_{h,v}[\exists y \in S', h(y) = v] \\ &\ge |S'| \cdot \frac{1}{p} - \frac{|S'|(|S'|-1)}{2} \cdot \frac{1}{p^2} \\ &\ge \varepsilon - \varepsilon^2 ,\end{aligned}$$

where $\varepsilon := \lceil(1+\delta)\tau\rceil/p \approx \delta/2$.

The probability that the subprocedure rejects is at most 2^{-n}. Hence the overall completeness is at least $\varepsilon - \varepsilon^2 - 2^{-n}$.

Soundness: Let $S := \{0,1\}^n \setminus \Pi'_{\mathsf{NO},n}(x)$ and suppose that the input is a NO instance so that $|S| \le \tau$. By a simple union bound, for any hash function h, we have

$$\Pr_v[\exists y \in S, h(y) = v] \le \frac{|h(S)|}{p} \le \frac{|S|}{p} \le \frac{\varepsilon}{1+\delta} .$$

So, except with probability at most $\varepsilon/(1+\delta)$, Merlin will not be able to provide a $y \in S$ satisfying $h(y) = v$. In this case, either $h(y) \ne v$, in which case Arthur rejects, or $y \in \Pi'_{\mathsf{NO},n}(x)$, in which case the subprocedure rejects with probability at least $1 - 2^{-n}$. Hence, the overall soundness is at least $1 - 2^{-n} - \varepsilon/(1+\delta)$.

By the above analysis, the gap between the completeness and soundness probabilities is at least

$$\begin{aligned}\varepsilon - \varepsilon^2 - \frac{\varepsilon}{1+\delta} - 2^{-n+1} &= \varepsilon\left(1 - \varepsilon - \frac{1}{1+\delta}\right) - 2^{-n+1} \\ &= \varepsilon\left(\frac{\delta}{1+\delta} - \varepsilon\right) - 2^{-n+1} \\ &\ge \frac{\varepsilon^2}{10} .\end{aligned}$$

The result follows by noting that, e.g., $\varepsilon \ge \delta/4$ (since there is always a prime between x and $2x$). □

3.2 An approximation of the range size

We now show a convenient approximation of the range size of a circuit.

For a circuit $C : \{0,1\}^n \to \{0,1\}^m$ and $r \in \mathbb{R}$, let $s_C(r) := |\{y \in \{0,1\}^m \ : \ |C^{-1}(y)| \geq r\}|$ be the number of elements in the range with at least r preimages. Notice that though the more natural domain for s_C is integers between 1 and 2^n, we allow for arbitrary real number inputs r, which will be convenient. Notice also that $s_C(1) = |\mathrm{Im}(C)|$ is the size of the image of C, the quantity that we would like to upper bound.

In particular, instead of upper bounding $s_C(1)$, we try to lower bound $2^n - s_C(1)$ using the following simple lemma.

Lemma 3.5. *For any circuit $C : \{0,1\}^n \to \{0,1\}^m$,*

$$2^n - s_C(1) = \int_1^{2^n} s_C(r)\mathrm{d}r \ .$$

Proof. Notice that by double counting,

$$\begin{aligned} 2^n &= \sum_{k=1}^{2^n} k|\{y \in \{0,1\}^m \ : \ |C^{-1}(y)| = k\}| \\ &= \sum_{j=1}^{2^n} |\{y \in \{0,1\}^m \ : \ |C^{-1}(y)| \geq j\}| \\ &= s_C(1) + \int_1^{2^n} s_C(r)\mathrm{d}r \ , \end{aligned}$$

as needed. □

It is then natural to define $T := 2^n - s_C(1) = \int_1^{2^n} s_C(r)\mathrm{d}r$. In order to approximate T, we further define two (easier to compute) quantities. In particular, for a fixed integer $\ell \geq 1$, we define

$$T_{\mathsf{odd}} := (1-2^{-1/\ell}) \sum_{i=1}^{\ell n} 2^{(2i-1)/(2\ell)} s_C(2^{(2i-1)/(2\ell)}) \qquad T_{\mathsf{even}} := (1-2^{-1/\ell}) \sum_{i=1}^{\ell n} 2^{(2i)/(2\ell)} s_C(2^{(2i)/(2\ell)}) \ .$$

These are good approximations of T, as captured in the following lemma.

Lemma 3.6. *The following inequalities hold.*

- $T_{\mathsf{odd}} \geq 2^{n-1/\ell} - s_C(1)$.
- $T_{\mathsf{even}} \leq T$.

Proof. Since $s_C(r)$ is non-increasing in r, we see that for any i,

$$\begin{aligned} \int_{2^{(2i-1)/(2\ell)}}^{2^{(2i+1)/(2\ell)}} s_C(r)\mathrm{d}r &\leq (1 - 2^{-1/\ell})2^{(2i+1)/(2\ell)} s_C(2^{(2i-1)/(2\ell)}) \ , \\ \int_{2^{(i-1)/\ell}}^{2^{i/\ell}} s_C(r)\mathrm{d}r &\geq (1 - 2^{-1/\ell})2^{i/\ell} s_C(2^{i/\ell}) \ . \end{aligned} \tag{1}$$

For the first inequality, we know

$$\begin{aligned}
T &= \int_1^{2^n} s_C(r)\mathrm{d}r \\
&= \int_1^{2^{1/(2\ell)}} s_C(r)\mathrm{d}r + \int_{2^{1/(2\ell)}}^{2^n} s_C(r)\mathrm{d}r \\
&\le \int_1^{2^{1/(2\ell)}} s_C(r)\mathrm{d}r + (1-2^{-1/\ell})\sum_{i=1}^{\ell n} 2^{(2i+1)/(2\ell)} s_C(2^{(2i-1)/(2\ell)}) \\
&= (2^{1/(2\ell)}-1)s_C(1) + 2^{1/\ell}(1-2^{-1/\ell})\sum_{i=1}^{\ell n} 2^{(2i-1)/(2\ell)} s_C(2^{(2i-1)/(2\ell)}) \\
&= (2^{1/(2\ell)}-1)s_C(1) + 2^{1/\ell} T_{\mathsf{odd}} \ .
\end{aligned}$$

Rearranging and recalling that $T := 2^n - s_C(1)$, we have

$$T_{\mathsf{odd}} \ge 2^{n-1/\ell} - 2^{-1/(2\ell)} s_C(1) > 2^{n-1/\ell} - s_C(1) \ .$$

For the second inequality, summing Equation (1) from $i = 1$ to ℓn, we see that

$$T = \sum_{i=1}^{\ell n} \int_{2^{(i-1)/\ell}}^{2^{i/\ell}} s_C(r)\mathrm{d}r \ge \sum_{i=1}^{\ell n} (1-2^{-1/\ell}) 2^{i/\ell} s_C(2^{i/\ell}) = T_{\mathsf{even}} \ ,$$

as needed. □

Intuitively, if we have good lower bounds for $s_C(2^{j/(2\ell)})$, we can convert them into a good lower bound for T and hence a good upper bound for $|\mathrm{Im}(C)|$. We now show how to lower bound $s_C(2^{j/(2\ell)})$ by "applying Theorem 3.4 twice."

Lemma 3.7. *For any circuit $C : \{0,1\}^n \to \{0,1\}^m$, integer $1 \le \ell \le \mathrm{poly}(n)$, $1 \le i \le \ell n$ and a threshold value τ, the following promise problem Π is in* prAM*:*

- $\Pi_{\mathsf{YES}} : s_C(2^{i/\ell}) \ge \tau$
- $\Pi_{\mathsf{NO}} : s_C(2^{(i-1)/\ell}) \le 2^{-1/\ell} \cdot \tau$

Proof. Consider the following promise problem $\Pi' = (\Pi'_{\mathsf{YES}}, \Pi'_{\mathsf{NO}})$:

- $\Pi'_{\mathsf{YES}} : \{(C, y) : |C^{-1}(y)| \ge 2^{i/\ell}\}$
- $\Pi'_{\mathsf{NO}} : \{(C, y) : |C^{-1}(y)| \le 2^{(i-1)/\ell}\}$

Notice that for any circuit C and $y \in \{0,1\}^m$, the set $C^{-1}(y) = \{x \in \{0,1\}^n : C(x) = y\}$ can be decided in P (by simply checking if $C(x) = y$). More specifically, the promise problem (in fact a language) $(C^{-1}(y), \overline{C^{-1}(y)})$ is contained in $\mathsf{P} \subseteq \mathsf{prAM}$. It follows from Theorem 3.4 that Π' is in prAM.

Since Π' is in prAM, again by Theorem 3.4 we know that Π is also in prAM. In particular, by definition we have $s_C(2^{i/\ell}) = |\Pi'_{\mathsf{YES},m}(C)|$ and $s_C(2^{(i-1)/\ell}) = 2^m - |\Pi'_{\mathsf{NO},m}(C)|$, following the notations of Theorem 3.4. □

3.3 An AM protocol for upper bounding range size

We are now ready to show that upper bounding the range size is contained in AM.

Proof of Theorem 3.2. The fact that GapRangeSize is in prcoAM is immediate from Theorem 3.4. So, we only need to present an AM protocol for GapRangeSize. To that end, we present a 3-round AM protocol for GapRangeSize as follows. One can then compress the number of rounds to 2 via Theorem 2.14.

Set $\ell := \lceil 2/\log(1+\varepsilon)\rceil$. For each $i \in [1, \ell n]$, Merlin sends a value $s_i \in [2^n]$, claiming that $s_C(2^{(2i-1)/(2\ell)}) = s_i$. Arthur then runs the protocol from Lemma 3.7 (with parameters $\tau' = s_i$, $\ell' = 2\ell$ and $i' = 2i-1$) for each $i \in [1, \ell n]$ in parallel (note that we would use a protocol in $\mathsf{prAM}_{1,2^{-n}}$ guaranteed by Theorem 2.13). Arthur rejects if any of these ℓn protocols reject.

Arthur then proceeds by computing

$$\widetilde{T} := (1 - 2^{-1/\ell}) \sum_{i=1}^{\ell n} 2^{(2i-1)/(2\ell)} s_i \ .$$

He accepts if and only if $\widetilde{T} \geq 2^{n-1/\ell} - \tau$.

Completeness: Suppose that Merlin is honest and $|\mathrm{Im}(C)| \leq \tau$. In particular, $s_i = s_C(2^{(2i-1)/(2\ell)})$ for all $i \in [1, \ell n]$ and all executions of Lemma 3.7 would accept. And we have from Lemma 3.6.

$$\widetilde{T} = T_{\mathsf{odd}} \geq 2^{n-1/\ell} - \tau \ .$$

Soundness: Suppose that Arthur accepts at the end of the protocol.

By a union bound over all polynomially many executions of the protocol from Lemma 3.7, we know that for each $i \in [1, \ell n]$, $s_C(2^{(2i-1)/(2\ell)}) \geq 2^{-1/(2\ell)} s_i$. We have

$$\begin{aligned}
2^{1/\ell}\widetilde{T} &= (1 - 2^{-1/\ell}) \sum_{i=1}^{\ell n} 2^{(2i-1)/(2\ell)} \cdot 2^{1/\ell} \cdot s_i \\
&\leq (1 - 2^{-1/\ell}) \sum_{i=1}^{\ell n} 2^{(2i)/(2\ell)} \cdot 2^{1/\ell} \cdot s_C(2^{(2i-2)/(2\ell)}) \\
&\leq (1 - 2^{-1/\ell}) \sum_{i=1}^{\ell n} 2^{(2i)/(2\ell)} \cdot 2^{2/\ell} \cdot s_C(2^{(2i)/(2\ell)}) + (1 - 2^{-1/\ell}) 2^{2/\ell} s_C(1) \\
&= 2^{2/\ell} T_{\mathsf{even}} + (2^{2/\ell} - 2^{1/\ell}) s_C(1) \\
&\leq 2^{2/\ell} (T_{\mathsf{even}} + s_C(1)) - 2^{1/\ell} s_C(1) \\
&\leq 2^{2/\ell} \cdot 2^n - 2^{1/\ell} s_C(1) \ ,
\end{aligned}$$

where the last inequality follows from Lemma 3.6 and the fact that $T + s_C(1) = 2^n$.

Combining the above with the fact that Arthur accepts so that $\widetilde{T} \geq 2^{n-1/\ell} - \tau$, we see that $2^n - 2^{1/\ell}\tau \leq 2^{2/\ell} \cdot 2^n - 2^{1/\ell} s_C(1)$, which after rearranging gives

$$s_C(1) \leq \tau + \varepsilon \cdot 2^{n-1/\ell} < \tau + \varepsilon \cdot 2^n \ ,$$

as needed. □

4 An "AM protocol for Avoid"

Combining the high-level outline from Section 1.2.1 with the tools from Section 3, we now show an AMA protocol for Avoid that succeeds with probability $1 - 1/\operatorname{poly}(n)$.

Theorem 4.1. *For any $s(n) \geq 1/\operatorname{poly}(n)$, there is a $\mathsf{SearchAMA}_{1-2^{-n},s(n)}$ algorithm for Avoid on circuits $C : \{0,1\}^n \to \{0,1\}^{n+1}$.*

Proof. Let $\varepsilon = (s - 2^{-n})/2$. Without loss of generality, we assume that the AM protocol from Theorem 3.2 has two rounds, completeness 1, and soundness error 2^{-2n} via Theorem 2.13.

The protocol consists of two parallel parts:

1. Part A: For $i \in [\lceil 1/\varepsilon \rceil]$, set $\tau_i = i\varepsilon 2^n$. Arthur runs the protocol from Theorem 3.2 with parameters $(\varepsilon' = \varepsilon, \tau' = \tau_i)$, all in parallel.
2. Part B: Arthur samples $y_1, \ldots, y_T$ uniformly from $\{0,1\}^{n+1}$ and sends them to Merlin for $T = \lceil n/\varepsilon^2 \rceil$. Merlin is expected to send to Arthur $x_1, \ldots, x_T$ where $C(x_i) = y_i$ for any $y_i \in \operatorname{Im}(C)$ (and arbitrary x_i for $y_i \notin \operatorname{Im}(C)$).

Next, we specify Arthur's behavior upon receiving Merlin's response from the two parallel parts.

Let $j \in [\lceil 1/\varepsilon \rceil]$ be the smallest index such that Arthur accepts the range-size upper bound protocol. In particular, we have that $|\operatorname{Im}(C)| \leq \tau_j + \varepsilon 2^n$.

Let $t := |\{i : C(x_i) = y_i\}|$. Arthur rejects if $t \leq j\varepsilon T - \sqrt{nT}$. Otherwise, Arthur outputs a uniformly sampled string from $\{y_i : C(x_i) \neq y_i\}$.

Next, we proceed to analyze the correctness of the protocol. Let Y_i be a random variable such that $Y_i = 1$ if and only if $y_i \in \operatorname{Im}(C)$. Let $\alpha := \frac{|\operatorname{Im}(C)|}{2^{n+1}}$. Let $Y = \sum_{i=1}^{T} Y_i$. By the Chernoff-Hoeffding bound, we have

$$\Pr_{Y}[|Y - \alpha T| \geq \sqrt{nT}] \leq 2\exp\left(-\frac{2nT}{T}\right) = 2e^{-2n} .$$

Completeness: Let j be the index such that $\tau_j \leq |\operatorname{Im}(C)| \leq \tau_{j+1}$. In particular, $\alpha T > j\varepsilon T$ and $\Pr[Y < j\varepsilon T - \sqrt{nT}] \leq 2e^{-2n}$. Moreover, the range-size upper bound protocol for $|\operatorname{Im}(C)| \geq \tau_j$ errs with probability at most 2^{-2n}. The overall completeness is at least $(1 - 2e^{-2n} - 2^{-2n}) \leq 1 - 2^{-n}$.

Soundness: By soundness of Theorem 3.2, we know that $|\operatorname{Im}(C)| \leq \tau_{j+1}$ except with probability 2^{-2n}. In particular, with probability $(1 - 2e^{-2n})$ we have $Y \leq (j+1)\varepsilon T + \sqrt{nT}$. We also have that $t \geq j\varepsilon T - \sqrt{nT}$. The probability that Arthur outputs a string in the image of C is at most:

$$\frac{Y - t}{T - t} \leq \frac{\varepsilon T + 2\sqrt{nT}}{T} \leq 2\varepsilon .$$

By union bound, the overall soundness error is at most $2^{-2n} + 2e^{-2n} + 2\varepsilon < s$. □

5 Any decision problem that reduces to Avoid is in $\mathsf{AM} \cap \mathsf{coAM}$

We now show that any decision problem that reduces to range avoidance (even under randomized reductions that make many oracle queries adaptively) lies quite low down in the polynomial hierarchy, specifically in $\mathsf{AM} \cap \mathsf{coAM}$. In particular, this implies that range avoidance cannot be NP-hard unless the polynomial hierarchy collapses (to AM [BHZ87]).

We will first need a rather technical proposition that instantiates the ideas described in Section 1.2.3. To that end, for $C : \{0,1\}^n \to \{0,1\}^{n+1}$ and $y_1, \ldots, y_m \in \{0,1\}^{n+1}$, we define

$$\mathrm{firstAvoid}(C, y_1, \ldots, y_m) := \min\{j \ : \ y_j \notin \mathrm{Im}(C)\} \ .$$

(If all y_i are in $\mathrm{Im}(C)$, then we simply say that firstAvoid is undefined. This will be a low probability event over uniformly sampled y_i when m is reasonably large, so we will not be concerned with this case.) In other words, firstAvoid is the index of the first solution to the Avoid instance C among the list of candidate solutions $y_1, \ldots, y_m \in \{0,1\}^{n+1}$. For a list $\boldsymbol{C} := (C_1, \ldots, C_\ell)$ of such circuits and a list $\boldsymbol{y} := (y_{i,j})_{1 \le i \le \ell, 1 \le j \le m}$ of such strings, we write

$$\mathrm{firstAvoid}(\boldsymbol{C}, \boldsymbol{y}) := (j_1, \ldots, j_\ell) := (\mathrm{firstAvoid}(C_i, y_{i,1}, \ldots, y_{i,m}))_{1 \le i \le \ell} \in [m]^\ell$$

for the list of indices obtained by applying firstAvoid to $(C_i, y_{i,1}, \ldots, y_{i,m})$ for each i.

From our perspective, the only important property of firstAvoid is that y_{j_i} is a solution to the Avoid instance C_i and it is uniquely determined by $\boldsymbol{C}$ and $\boldsymbol{y}$. In other words, we think of $\mathrm{firstAvoid}(\boldsymbol{C}, \boldsymbol{y})$ as representing a canonical choice of solutions y_{j_i} to Avoid for each of the instances $C_1, \ldots, C_\ell : \{0,1\}^n \to \{0,1\}^{n+1}$ from the candidate solutions $\boldsymbol{y}$.

Finally, for two lists $\boldsymbol{j} := (j_1, \ldots, j_\ell), \boldsymbol{j}' := (j'_1, \ldots, j'_\ell) \in [m]^\ell$ of indices, we write

$$\Delta(\boldsymbol{j}, \boldsymbol{j}') := |\{i \ : \ j_i \neq j'_i\}|$$

for the number of indices in which they differ.

With this, we can now present our main technical result, which essentially says that there is "an Arthur-Merlin protocol for computing $\boldsymbol{j}'$ such that $\Delta(\mathrm{firstAvoid}(\boldsymbol{C}, \boldsymbol{y}), \boldsymbol{j}') \lesssim \sqrt{\ell}$."

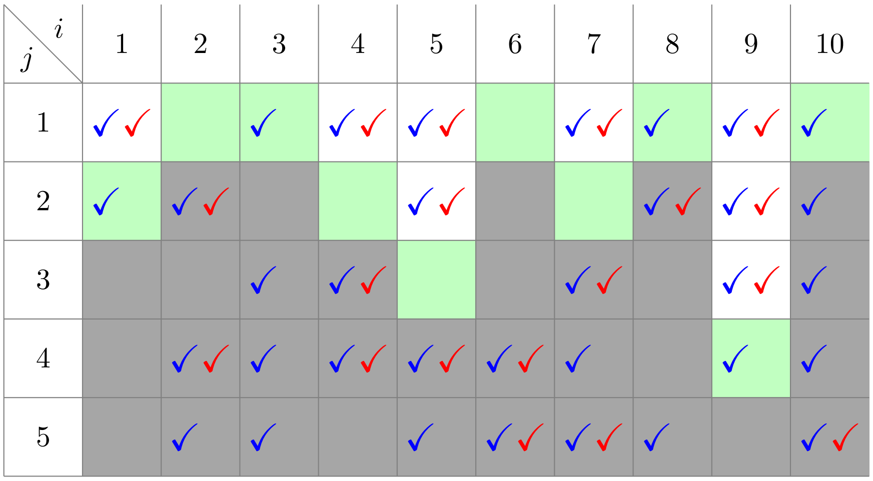
Proposition 5.1. *There is a deterministic polynomial-time algorithm $\mathcal{A}$ (Arthur) and a (simple but computationally unbounded) function $\mathcal{M}$ (honest Merlin) with the following properties.*

- **(Syntax of honest Merlin.)** *$\mathcal{M}$ takes as input a list of circuits $C_1, \ldots, C_\ell : \{0,1\}^n \to \{0,1\}^{n+1}$ (possibly with duplicates) and bit strings $y_{i,j} \in \{0,1\}^{n+1}$ for $1 \le i \le \ell$ and $1 \le j \le m$ and outputs a list of strings $x_{i,j} \in \{0,1\}^n$.*
- **(Syntax of Arthur.)** *$\mathcal{A}$ takes as input circuits $C_1, \ldots, C_\ell : \{0,1\}^n \to \{0,1\}^{n+1}$, bit strings $y_{i,j} \in \{0,1\}^{n+1}$ and $x_{i,j} \in \{0,1\}^n$ for $1 \le i \le \ell$ and $1 \le j \le m$, and approximations $T_1, \ldots, T_\ell \in [2^n]$ of $|\mathrm{Im}(C_i)|$ and either outputs $\perp$ or a list of indices $j_1, \ldots, j_\ell \in [m]$.*
- **(Completeness.)** *For any list of circuits $\mathbf{C} := (C_1, \ldots, C_\ell)$, any m, and $\boldsymbol{T} := (T_1, \ldots, T_\ell) \in [2^n]^\ell$ with $T_i \le |\mathrm{Im}(C_i)|$ for all i,*

  $$\Pr_{\boldsymbol{y} \sim (\{0,1\}^{n+1})^{m\ell}}[\mathcal{A}(\mathbf{C}, \boldsymbol{y}, \mathcal{M}(\mathbf{C}, \boldsymbol{y}), \boldsymbol{T}) = \mathrm{firstAvoid}(\boldsymbol{C}, \boldsymbol{y})] \ge 1 - \ell 2^{-\Omega(m)} \ .$$

  (In particular, Arthur's output $(j_1, \ldots, j_\ell)$ will satisfy that $y_{i,j_i} \notin \mathrm{Im}(C_i)$ for all i with probability $1 - \ell 2^{-\Omega(m)}$.)
- **(Soundness.)** *For* any *function $\mathcal{M}^\dagger$, any list of circuits $\mathbf{C} := (C_1, \ldots, C_\ell)$, any m, and any $\boldsymbol{T} := (T_1, \ldots, T_\ell) \in [2^n]^\ell$,*

  $$\Pr_{\boldsymbol{y} \sim (\{0,1\}^{n+1})^{m\ell}}[\boldsymbol{j} \leftarrow \mathcal{A}(\mathbf{C}, \boldsymbol{y}, \mathcal{M}^\dagger(\mathbf{C}, \boldsymbol{y}), \boldsymbol{T}) \ : \ \boldsymbol{j} = \perp \ \mathit{or} \ \Delta(\mathrm{firstAvoid}(\boldsymbol{C}, \boldsymbol{y}), \boldsymbol{j}) \le E] \ge 1 - \ell 2^{-\Omega(m)} \ ,$$



j i	1	2	3	4	5	6	7	8	9	10
1	✓✓	🟩	🟩 ✓	✓✓	✓✓	🟩	✓✓	🟩 ✓	✓✓	🟩 ✓
2	🟩 ✓	▨ ✓✓	▨	🟩	✓✓	▨	🟩	▨ ✓✓	✓✓	▨ ✓
3	▨	▨	▨ ✓	▨ ✓✓	🟩	▨	▨ ✓✓	▨	✓✓	▨ ✓
4	▨	▨ ✓✓	▨ ✓	▨ ✓✓	▨ ✓✓	▨ ✓✓	▨ ✓	▨	🟩 ✓	▨ ✓
5	▨	▨ ✓	▨ ✓	▨	▨ ✓	▨ ✓✓	▨ ✓✓	▨ ✓	▨	▨ ✓✓

Figure 1: A visualization of how Arthur parses his input in the protocol described in Proposition 5.1. If the box (i, j) contains a blue checkmark, then $y_{i,j}$ has a preimage under C_i (which Arthur does not necessarily know). If (i, j) has a red checkmark, then Merlin provided a preimage for $x_{i,j}$ such that $C_i(x_{i,j}) = y_{i,j}$ (which Arthur does know). Green shaded boxes indicate Arthur's candidate indices j_i. In this case, Arthur's candidate indices are $(j_1 = 2, j_2 = 1, j_3 = 1, j_4 = 2, j_5 = 3, j_6 = 1, j_7 = 2, j_8 = 1, j_9 = 4, j_{10} = 1)$. Gray shaded boxes are ignored by Arthur because they occur in boxes (i, j) with $j_i < i$. During execution, Arthur will find $A_1 = \{1, \ldots, 10\}, A_2 = \{1, 4, 5, 7, 9\}, A_3 = \{5, 9\}, A_4 = \{9\}, A_5 = \varnothing$ and during our analysis in Proposition 5.1, we will find that $\ell_1 = 8, \ell_2 = 3, \ell_3 = 1, \ell_4 = 1, \ell_5 = 0$.

where

$$E := m^{1.5}\sqrt{\ell} + m \sum_{i=1}^{\ell} \max\left\{\frac{|\mathrm{Im}(C_i)| - T_i}{2^{n+1}}, 0\right\} .$$

(In other words, Arthur is likely to either output $\perp$ *or a list that matches* $\mathrm{firstAvoid}(\boldsymbol{C}, \boldsymbol{y})$ *in all but at most* E *places.)*

Proof. The function $\mathcal{M}$ representing honest Merlin is quite simple. Merlin sets $x_{i,j}$ to be any element $x_{i,j} \in C_i^{-1}(y_{i,j})$ when this set is not empty. In other words, he provides Arthur with one preimage of $y_{i,j}$ whenever such a preimage exists. (The choice of preimage does not matter, nor does it matter what value $x_{i,j}$ takes when $y_{i,j} \notin \mathrm{Im}(C_i)$.)

Figure 1 gives a schematic representation of the behavior of Arthur.

On input $C_1, \ldots, C_\ell : \{0,1\}^n \to \{0,1\}^{n+1}$, $(y_{i,j})_{i\in[\ell],j\in[m]}$, $(x_{i,j})_{i\in[\ell],j\in[m]}$, and $T_1, \ldots, T_\ell \in [2^n]$, the function $\mathcal{A}$ representing Arthur behaves as follows. Arthur first simply generates a candidate list $j_1, \ldots, j_\ell$ of indices by taking j_i to be minimal such that y_{i,j_i} does not have a preimage in the list—i.e., $j_i := \min\{j \ : \ y_{i,j} \neq C_i(x_{i,j})\}$. (If there is some i such that all of the $y_{i,j} = C_i(x_{i,j})$ for all j, then j_i is not defined and Arthur outputs $\perp$. This will be a low-probability event.)

Arthur will then either output the indices $j_1, \ldots, j_\ell$ or $\perp$, depending on the following test. Define for each $j \in [m]$ the set $A_j := \{i \in [\ell] \ : \ j_i \geq j\}$. In other words A_j is the set of indices i for which Arthur received a preimage for $y_{i,j'}$ for all $j' < j$. (Arthur simply ignores $x_{i,j}$ when $i \notin A_j$.) And, let

$$b_j := \sum_{i\in A_j} \frac{T_i}{2^{n+1}} .$$

Notice that if $T_i = |\mathrm{Im}(C_i)|$, then a uniformly random string in $\{0,1\}^{n+1}$ will be in the image of C_i with probability $T_i/2^{n+1}$. And notice that for $i \in A_j$, we will have $i \in A_{j+1}$ if and only if Arthur received a preimage for $y_{i,j}$. We therefore expect $|A_{j+1}| \approx b_j$ if Merlin is honest, and Arthur should therefore be suspicious if A_{j+1} is significantly smaller than this (since this would suggest that Merlin has failed to provide some preimages or that T_i is not a good estimate for $|\mathrm{Im}(C_i)|$). Therefore, Arthur outputs $\perp$ if there exists a j such that

$$|A_{j+1}| < b_j - \frac{\sqrt{m\ell}}{2} \ .$$

Otherwise, Arthur outputs $j_1, \ldots, j_\ell$.

Clearly Arthur is efficient and both Arthur and Merlin have the claimed syntax.

To prove completeness and soundness, we first simply observe that the probability that there exists an index i such that for all j we have $y_{i,j} \in \mathrm{Im}(C_i)$ is at most

$$\sum_{i=1}^{\ell} \left(\frac{|\mathrm{Im}(C_i)|}{2^{n+1}} \right)^m \leq \ell 2^{-m} \ .$$

So, in what follows, we may assume that this does not happen. In particular, this means that the j_i are well defined.

Let $\ell_j := |\{i \in A_j \ : \ y_{i,j} \in \mathrm{Im}(C_i)\}|$ and

$$b_j^* := \sum_{i \in A_j} \frac{|\mathrm{Im}(C_i)|}{2^{n+1}} \ .$$

Notice that if Merlin is honest, then $|A_{j+1}| = \ell_j$, and notice that ℓ_j is a sum of $|A_j| \leq \ell$ independent Bernoulli random variables, with expectation $\mathbb{E}[\ell_j] = b_j^*$. By the Chernoff-Hoeffding bound (Lemma 2.20) and the union bound, we have that

$$|\ell_j - b_j^*| \leq \frac{\sqrt{m\ell}}{2} \tag{2}$$

for all $j \in [m]$ except with probability at most $m2^{-\Omega(m)} \leq \ell 2^{-\Omega(m)}$ over the $y_{i,j}$.

We first claim that Equation (2) implies completeness. In particular, if $T_i \leq |\mathrm{Im}(C_i)|$ then clearly $b_j \leq b_j^*$. And if Merlin is honest, then $|A_{j+1}| = \ell_j$. So, if Equation (2) holds, then Arthur will not output $\perp$. It is immediate that if Arthur does not output $\perp$ and Merlin is honest, we have $j_i = \mathrm{firstAvoid}(C_i, y_{i,1}, \ldots, y_{i,m})$ for all i. So, completeness holds.

For soundness, consider an arbitrary $\mathcal{M}^\dagger$, and let $\boldsymbol{j}^\dagger := (j_1^\dagger, \ldots, j_\ell^\dagger) \in [m]^\ell$ be the indices generated by Arthur on input $\mathbf{C}$, $\mathbf{y}$, $\mathcal{M}^\dagger(\boldsymbol{C}, \boldsymbol{y})$, and $\boldsymbol{T}$. We claim that if Equation (2) holds, then either Arthur outputs $\perp$ or $\Delta(\boldsymbol{j}^\dagger, \mathrm{firstAvoid}(\boldsymbol{C}, \boldsymbol{y})) \leq E$. To see this, notice that for any j, $\ell_j - |A_{j+1}|$ is precisely the number of elements $y_{i,j}$ with $i \in A_j$ that are in the image of C_i but that $\mathcal{M}^\dagger$ has (very rudely) not provided a preimage for. It follows that

$$\Delta(\boldsymbol{j}^\dagger, \mathrm{firstAvoid}(\boldsymbol{C}, \boldsymbol{y})) = \sum_{j=1}^{m} (\ell_j - |A_{j+1}|) \ .$$

We therefore proceed to show that $\ell_j - |A_{j+1}| \leq E/m$ for all j, assuming that Merlin provides enough preimages so that Arthur does not output $\perp$.

Indeed, recall that Arthur *will* output $\perp$ unless

$$|A_{j+1}| \geq b_j - \frac{\sqrt{m\ell}}{2}$$

for all j. So, if Equation (2) holds and Arthur does not output $\perp$, then we have for all j that

$$\ell_j - |A_{j+1}| \leq \sqrt{m\ell} + b_j^* - b_j = \sqrt{m\ell} + \sum_{i \in A_j} \frac{|\mathrm{Im}(C_i)| - T_i}{2^{n+1}} \leq \sqrt{m\ell} + \sum_{i=1}^{\ell} \max\Big\{\frac{|\mathrm{Im}(C_i)| - T_i}{2^{n+1}}, 0\Big\} = \frac{E}{m}\ .$$

The result follows. □

From this (together with the range-size upper bound protocol from Section 3), we derive the main result of this section.

Theorem 5.2. $\mathsf{BPP}^{\mathsf{Avoid}} \subseteq \mathsf{AM} \cap \mathsf{coAM}$ *and* $\mathsf{prBPP}^{\mathsf{Avoid}} \subseteq \mathsf{prAM} \cap \mathsf{prcoAM}$. *In other words, any (promise) decision problem that efficiently reduces to* Avoid *(even via randomized reductions that make many adaptive oracle queries) is in (promise)* $\mathsf{AM} \cap \mathsf{coAM}$*.*

In particular, Avoid *is not* NP*-hard (even under randomized reductions that make many adaptive oracle queries) unless the polynomial hierarchy collapses (to* AM*).*

Proof. Let $\mathcal{B}^{\mathcal{O}}$ be a randomized reduction from some problem L to Avoid—i.e., if $\mathcal{O}$ is an oracle for Avoid, then $\mathcal{B}^{\mathcal{O}}(x)$ outputs 1 with probability at least $2/3$ for any YES instance x of L and outputs 0 with probability at least $2/3$ for any NO instance x of L.[8]

We present a four-round Arthur-Merlin protocol in which Arthur outputs either YES, NO, or $\perp$ such that if Merlin is honest then for all YES instances of L Arthur outputs YES with probability at least $1 - 2^{-\Omega(n)}$ and for all NO instances of L Arthur outputs NO with probability at least $1 - 2^{-\Omega(n)}$. And, if Merlin is dishonest, Arthur either outputs the correct answer or $\perp$ with probability at least $1 - 2^{-\Omega(n)}$. Notice that this immediately implies that both L and its complement have four-round Arthur-Merlin protocols. And, by Theorem 2.14, these can be converted into a two-round protocol, and we immediately see that $\mathrm{L} \in \mathsf{AM} \cap \mathsf{coAM}$ (or $\mathsf{prAM} \cap \mathsf{prcoAM}$ if L is a promise problem), as claimed.

To that end, let $R := R(n) \leq \mathrm{poly}(n)$, $Q := Q(n) \leq \mathrm{poly}(n)$, and $10n \leq N := N(n) \leq \mathrm{poly}(n)$ be such that on any input of length n, $\mathcal{B}^{\mathcal{O}}$ flips at most R coins, makes at most Q queries to its oracle, and such that all queries are on circuits with input length at most N. By possibly padding the input lengths as in Lemma 2.3, we may assume without loss of generality that all oracle queries are made on circuits $C : \{0,1\}^N \to \{0,1\}^{N+1}$ that take precisely N bits of input. We may also assume without loss of generality that the reduction always makes precisely Q oracle queries.

On input $x \in \{0,1\}^n$, Arthur and honest Merlin behave as follows. Let $m := 10NQ$ and $\ell := 100Q^2m^4$. Arthur first samples $r_1, \ldots, r_\ell \sim \{0,1\}^R$ (random coins to be used to run $\mathcal{B}$) and $y_{i,j}^{(k)} \sim \{0,1\}^{N+1}$ (candidate solutions to various instances of Avoid) for $1 \leq i \leq \ell$, $1 \leq j \leq m$, and $1 \leq k \leq Q$. Arthur then sends all of this to Merlin.

Honest Merlin does the following. He simulates ℓ different runs of the reduction $\mathcal{B}^{\mathcal{O}}$ in parallel, using r_i as the random coins of $\mathcal{B}^{\mathcal{O}}$ in the ith instance. When the reductions make their kth oracle queries, $\mathbf{C}^k := (C_1^k, \ldots, C_\ell^k)$, Merlin computes $\boldsymbol{X}^k := \mathcal{M}(\mathbf{C}, \boldsymbol{y}^k)$ for $\boldsymbol{y}^k :=$

[8] In fact, our protocol works even if $\mathcal{B}^{\mathcal{O}}$ only outputs the correct answer with probability at least $2/3$ when $\mathcal{O}$ is sampled uniformly at random from the set of all Avoid oracles. But, we do not make this formal.

$(y^k_{i,j})_{1\le i\le \ell, 1\le j\le m}$, where $\mathcal{M}$ is the honest Merlin algorithm from Proposition 5.1. And, Merlin sets $\boldsymbol{T}^k := (|\mathrm{Im}(C^k_1)|, \ldots, |\mathrm{Im}(C^k_\ell)|)$. Let $(j^k_1, \ldots, j^k_\ell) \leftarrow \mathcal{A}(\mathbf{C}^k, \boldsymbol{y}^k, \boldsymbol{X}^k, \boldsymbol{T}^k)$ be the output generated by honest Arthur in the protocol from Proposition 5.1. (If $\mathcal{A}$ outputs $\perp$, then the protocol simply fails.) Merlin continues simulating the ℓ parallel reductions using $y^k_{i,j^k_i} \in \{0,1\}^{N+1}$ as the oracle's response to the query $C^k_i : \{0,1\}^N \to \{0,1\}^{N+1}$.

Finally, once all queries are made, Merlin sends $\boldsymbol{X}^1, \ldots, \boldsymbol{X}^Q$ and $\boldsymbol{T}^1, \ldots, \boldsymbol{T}^Q$ to Arthur.

Arthur then runs the same simulation as Merlin. In other words, he first uses random coins $r_1, \ldots, r_\ell$ to run ℓ parallel copies of the reduction to compute ℓ circuits $\boldsymbol{C}^1 := (C^1_1, \ldots, C^1_\ell)$ corresponding to the first query made in each parallel run. Then, for $k = 1, \ldots, Q$, he computes $(j^k_1, \ldots, j^k_\ell) \leftarrow \mathcal{A}(\boldsymbol{C}^k, \boldsymbol{y}^k, \boldsymbol{X}^k, \boldsymbol{T}^k)$, uses y^k_{i,j^k_i} as the response to the kth oracle query in the ith parallel run of the reduction. If any of these subprocedures output $\perp$, Arthur outputs $\perp$. Otherwise, each parallel simulation of $\mathcal{B}^{\mathcal{O}}$ terminates and outputs either YES or NO. Arthur takes the majority output as his provisional output, but he will not output it yet.

Instead, Merlin and Arthur then engage in $Q\ell$ parallel runs of the range-size upper bound protocol from Theorem 3.2 on input $(C^k_i, T^k_i)_{1\le i\le \ell, 1\le k\le q}$ with $\varepsilon := 1/\ell \ge 1/\mathrm{poly}(n)$ and using Theorem 2.13 to boost the soundness to, say, 2^{-N}. In other words, "Merlin convinces Arthur that $|\mathrm{Im}(C^k_i)| \le T^k_i$ in such a way that Arthur will catch a cheating Merlin with probability at least $1 - 2^{-N}$ if $|\mathrm{Im}(C^k_i)| > T^k_i + \varepsilon 2^N$." If Merlin fails to convince Arthur in any of these runs, Arthur outputs $\perp$. Otherwise, Arthur outputs the provisional answer from above.

Completeness of the protocol is then more-or-less immediate from completeness of the protocols from Theorem 3.2 and Proposition 5.1. In particular, completeness of the protocol for Proposition 5.1 implies that if Merlin is honest then y^k_{i,j^k_i} will be a valid solution to C^k_i for all i and k except with probability at most $\ell Q 2^{-\Omega(m)} \le 2^{-\Omega(n)}$. When this happens, correctness of $\mathcal{B}^{\mathcal{O}}$ together with the Chernoff-Hoeffding bound from Lemma 2.20 implies that the majority of the simulated reductions will output the correct answer except with probability at most $2^{-\Omega(\ell)} \le 2^{-\Omega(n)}$. In fact, we will need a slightly stronger fact below—that with probability $1 - 2^{-\Omega(n)}$ at least, say, $3/5$ of the simulated reductions will output the correct answer. And, completeness of the protocol from Theorem 3.2 implies that Arthur will output the resulting majority vote in this case (i.e., he will not output $\perp$ because one of the range-size upper bound protocols fails).

To see that the protocol is sound, first notice that soundness of the range-size upper bound protocol from Theorem 3.2 implies that Arthur will reject with probability at least $1 - 2^{-\Omega(n)}$ *unless* we have $|\mathrm{Im}(C^k_i)| - T^k_i \le \varepsilon 2^N$ simultaneously for all i and k. Let

$$E := m^{1.5}\sqrt{\ell} + \varepsilon m \ell < \frac{\ell}{10Q} \; .$$

The soundness property of the protocol from Proposition 5.1 then implies that except with probability at most $\ell Q 2^{-\Omega(m)} \le 2^{-\Omega(n)}$, Arthur either outputs $\perp$ or for all k we have that $\Delta(\mathrm{firstAvoid}(\boldsymbol{C}^k, \boldsymbol{y}^k), \boldsymbol{j}^k) \le E$. If this happens, then we see that the simulated responses y^k_{i,j^k_i} in this interaction agree with the simulated responses $y^k_{i,(j^k_i)'}$ in an interaction with honest Merlin (with the same randomness $r_1, \ldots, r_\ell$) for all but at most QE choices of i and k.[9] Therefore, with

[9] There is a subtlety here that we wish to call attention to, though our proof manages to avoid discussing it explicitly. The subtlety is that it would *not* be enough to argue that with high probability most of the y^k_{i,j^k_i} are valid solutions to their respective Avoid instances C^k_i. In particular, it could be the case that $\mathcal{B}^{\mathcal{O}}$ outputs the wrong answer if y^k_{i,j^k_i} is a

probability at least $1 - 2^{-\Omega(n)}$, either Arthur outputs $\perp$ or $\ell - QE > 9\ell/10$ of the simulated runs of the protocol will be identical to the runs in an honest interaction with Merlin. Since in an honest interaction with Merlin we have already established that with probability at least $1 - 2^{-\Omega(n)}$ at least $3\ell/5$ of the runs of the protocol will output the correct answer. The result follows. □

6 Avoid is almost in TFNP

The Avoid problem is in $\mathsf{TF\Sigma_2^P}$ rather than TFNP; although Avoid is total, one needs an NP oracle to verify any proposed solution. In fact, since checking if 0 is in the range of a circuit $C : \{0,1\}^n \to \{0,1\}^{n+1}$ is coNP-hard, putting Avoid in TFNP would imply $\mathsf{P} = \mathsf{NP}$. (See Footnote 3.)

However, to our surprise, we find that under plausible derandomization assumptions, Avoid has a randomized reduction to a TFNP problem which succeeds with very high probability.

Theorem 6.1. *Assuming Assumption 2.7, there exists a* TFNP *problem* A *such that for any constant* c*,* Avoid *has a randomized one-query reduction to* A *which succeeds with probability at least* $1 - 1/n^c$*.*

Proof. Let $G : \{0,1\}^{\alpha \cdot \log(n+1)} \to \{0,1\}^{n+1}$ be a $\frac{1}{n^3}$-PRG which fools n^3 size $\mathsf{NP}_{||}$ circuits, whose existence we have assumed by Assumption 2.7. By Theorem 2.16, the existence of such a G in particular implies that $\mathsf{prAM} = \mathsf{prNP}$. Since by Theorem 3.2, ε-GapRangeSize is in prAM for, say, $\varepsilon := 1/n^{50}$, it follows that ε-GapRangeSize is in prNP. Formally, there exists a polynomial-time verifier V such that for any circuit $C : \{0,1\}^n \to \{0,1\}^m$, if $|\mathrm{Im}(C)| \geq \tau + \varepsilon 2^n$, then for all π, $V(C, \tau, \pi) = 0$, but if $|\mathrm{Im}(C)| \leq \tau$, then there exists a π such that $V(C, \tau, \pi) = 1$.

The problem A is defined as follows. For sufficiently large n, given a circuit $C : \{0,1\}^n \to \{0,1\}^{n+1}$ where $|C| \leq n^2$, output both of the following.

1. $\tau \in [2^n], \pi \in \{0,1\}^{\mathrm{poly}(n)}$ such that $V(C, \tau, \pi) = 1$. In other words, an estimate τ of the range size of C and a proof π that $\tau \geq \tau^* - \varepsilon 2^n$, where $\tau^* := |\mathrm{Im}(C)|$.
2. $w_1, \ldots, w_{(n+1)^\alpha} \in \{0,1\}^n$ such that for at least a $\frac{\tau}{2^{n+1}} - \frac{1}{n^3}$ fraction of $i \in [(n+1)^\alpha]$, $C(w_i) = G(i)$.

We first show that A is a TFNP problem by showing that A is an FNP problem and that it is total. Since V is efficient and since checking if $C(w_i) = G(i)$ for sufficiently many $i \in [(n+1)^\alpha]$ can be done efficiently, there is a simple efficient procedure for verifying solutions to A, and A is therefore in FNP. To see that A is total, we claim that there is always a solution to A with $\tau = \tau^*$. Indeed, if $\tau = \tau^*$, then by the definition of V, there exists a polynomial size witness π such that $V(C, \tau, \pi) = 1$. And, for $\tau = \tau^*$, Lemma 2.9 tells us that at least a

$$\frac{\tau}{2^{n+1}} - \frac{1}{n^3}$$

fraction of the $G(i)$ must have a preimage under C. So, there exist solutions $w_1, \ldots, w_{(n+1)^\alpha}$ in this case.

valid solution but if it depends on the random coins r_i of $\mathcal{B}^{\mathcal{O}}$ in some way. This is a bit worrisome, because in our protocol we have sent r_i to Merlin, so one might worry that a malicious prover could cause Arthur to generate valid solutions y^k_{i,j^k_i} to the Avoid instances while still causing the protocol to fail. We are therefore being careful to point out that most of the y^k_{i,j^k_i} are not *only* valid solutions to the Avoid instance C^k_i. They are in fact equal to *specific* valid solutions $y^k_{i,(j^k_i)'}$ that are independent of r_i—specifically, the same valid solutions that Arthur finds when interacting with honest Merlin.

We now show how to reduce Avoid to A. By Lemma 2.3, we can reduce our avoid instance $C : \{0,1\}^n \to \{0,1\}^{n+1}$ where $|C| = s$ to an Avoid instance $C' : \{0,1\}^{n'} \to \{0,1\}^{n'+1}$ where $|C'| \leq {n'}^2$ and $n' \geq n^c$ in polynomial time. Notice that solving Avoid on C' with probability at least $1 - 1/n'$ will yield a solution to Avoid on C with probability at least $1 - 1/n^c$. We will therefore assume without loss of generality that we only wish to solve range avoidance on circuits $C : \{0,1\}^n \to \{0,1\}^{n+1}$ where $|C| \leq n^2$ with probability at least $1 - 1/n$.

We show a randomized reduction from Avoid on $C : \{0,1\}^n \to \{0,1\}^{n+1}$ where $|C| \leq n^2$ to A such that the reduction succeeds with probability at least $1 - 1/n$. The reduction feeds our circuit C to an oracle for A to get back τ, π (a witness that $\tau \gtrsim \tau^*$), and $w_1, \ldots, w_{(n+1)^\alpha} \in \{0,1\}^n$. Let $I := \{i : C(w_i) \neq G(i)\}$. The reduction samples i^* uniformly from I and outputs $G(i^*)$.

The reduction clearly runs in polynomial time. We now show correctness of the reduction. Let $J := \{i : G(i) \notin \mathrm{Im}(C)\}$. In other words, $G(i)$ is a solution to the Avoid instance if and only if $i \in J$. Since we must have $J \subseteq I$, it follows that the reduction succeeds with probability exactly $|J|/|I|$. So, it remains to prove that $|J|/|I| \geq 1 - 1/n$.

We will first bound $|I|$ and $|J|$ separately. First, by Lemma 2.9, we have

$$\left|\frac{|J|}{(n+1)^\alpha} - \left(1 - \frac{\tau^*}{2^{n+1}}\right)\right| \leq \frac{1}{n^3} \ .$$

Rearranging gets us

$$\frac{|J|}{(n+1)^\alpha} \geq 1 - \frac{\tau^*}{2^{n+1}} - \frac{1}{n^3} \ .$$

We now bound $|I|$. By the definition of A,

$$\frac{|I|}{(n+1)^\alpha} \leq 1 - \frac{\tau}{2^{n+1}} + \frac{1}{n^3} \leq 1 - \frac{\tau^* - \varepsilon 2^n}{2^{n+1}} + \frac{1}{n^3} \leq 1 - \frac{\tau^*}{2^{n+1}} + \frac{2}{n^3}.$$

Let $k = \tau^*/2^{n+1}$. Note that $k \leq 1/2$. We first bound $|I| - |J|$ and will then use that to bound $|J|/|I|$. We have

$$|I| - |J| \leq (n+1)^\alpha \Big((1 - k + 2/n^3) - (1 - k - 1/n^3)\Big) = 3(n+1)^\alpha/n^3 \ .$$

Therefore,

$$\frac{|J|}{|I|} = 1 - \frac{|I| - |J|}{|I|} \geq 1 - \frac{3(n+1)^\alpha/n^3}{|I|} \geq 1 - \frac{3/n^3}{1 - k + 2/n^3} \geq 1 - \frac{3/n^3}{1/2 + 2/n^3} \geq 1 - 1/n \ ,$$

as needed, where the last inequality holds for sufficiently large n. □

The following corollary follows immediately from combining Theorem 6.1 above with Theorem 2.8 (the fact that sufficiently strong circuit lower bounds imply sufficiently strong PRGs).

Corollary 6.2. *If* E *does not have* $2^{o(n)}$*-size non-deterministic circuits, then there exists a* TFNP *problem* A *such that for all constants c, there exists a randomized reduction from* Avoid *to* A *which succeeds with probability at least* $1 - 1/n^c$.

We are now able to use Theorem 6.1 to show that if plausible derandomization assumptions hold and SAT is not checkable, then Avoid is not NP-hard. This is an alternative proof (under less standard assumptions than PH ≠ AM) that Avoid is not NP-hard.

Corollary 6.3. *If Assumption 2.7 holds and* SAT *is not checkable, then* Avoid *is not* NP*-hard under randomized Turing reductions. Consequently, if* E *requires exponential size non-deterministic circuits and* SAT *is not checkable, then* Avoid *is not* NP*-hard under randomized Turing reductions.*

Proof. We will show that if Assumption 2.7 holds, and Avoid is NP-hard, then SAT is checkable. Say there is a randomized Turing reduction from SAT on an instance of length n to Avoid which runs in time n^c for some constant c. We assume without loss of generality (by standard probability amplification techniques) that the randomized reduction succeeds with probability at least 99/100. Since Assumption 2.7 holds, by Theorem 6.1, there exists a TFNP problem A such that Avoid has a randomized reduction to A which succeeds with probability at least $1 - 1/n^{c+1}$. Chaining these two reductions together, we see that we have a Turing reduction from SAT to A which succeeds with the following probability.

$$\frac{99}{100}\left(1 - \frac{1}{n^{c+1}}\right)^{n^c} \geq \frac{99}{100}e^{-\Theta(1/n)} \geq \frac{99}{100}(1 - \Theta(1/n))$$

Therefore, SAT has a randomized reduction to the TFNP problem A that succeeds with probability at least 2/3. It follows from Theorem 2.11 that SAT is checkable. □

References

[ABK24] Scott Aaronson, Harry Buhrman, and William Kretschmer. A qubit, a coin, and an advice string walk into a relational problem. In *ITCS*, 2024. 2

[Bab85] László Babai. Trading group theory for randomness. In *STOC*, 1985. 6, 15, 16

[BGG93] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Randomness in interactive proofs. *Computational Complexity*, 3(4):319–354, 1993. 15

[BHZ87] Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25:127–132, 1987. 2, 22

[BK95] Manuel Blum and Sampath Kannan. Designing programs that check their work. *Journal of the ACM*, 42(1):269–291, 1995. 14, 15

[BM88] László Babai and Shlomo Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988. 6, 15, 16

[CGL+23] Eldon Chung, Alexander Golovnev, Zeyong Li, Maciej Obremski, Sidhant Saraogi, and Noah Stephens-Davidowitz. The hardness of range avoidance for randomized algorithms implies Minicrypt. ECCC:2023/193/, 2023. 1, 2, 5, 10, 11, 12

[CHLR23] Yeyuan Chen, Yizhi Huang, Jiatu Li, and Hanlin Ren. Range avoidance, remote point, and hard partial truth table via satisfying-pairs algorithms. In *STOC*, 2023. 1

[CHR24] Lijie Chen, Shuichi Hirahara, and Hanlin Ren. Symmetric exponential time requires near-maximum circuit size. In *STOC*, 2024. 1

[CL24] Yilei Chen and Jiatu Li. Hardness of range avoidance and remote point for restricted circuits via cryptography. In *STOC*, 2024. 1, 2, 4, 10, 11

[CLL25] Lijie Chen, Jiatu Li, and Jingxun Liang. Maximum circuit lower bounds for exponential-time Arthur Merlin. In *STOC*, 2025. 1, 11

[Dru10] Andrew Donald Drucker. *PCPs for Arthur-Merlin games and communication protocols.* PhD thesis, Massachusetts Institute of Technology, 2010. 15

[FGM$^+$89] Martin Fürer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. On completeness and soundness in interactive proof systems. *Advances in Computing Research*, 5:429–442, 1989. 15

[FIM25] Noah Fleming, Deniz Imrek, and Christophe Marciot. Provably total functions in the polynomial hierarchy. In *CCC*, 2025. 1

[For87] Lance Fortnow. The complexity of perfect zero-knowledge. In *STOC*, 1987. 11

[GGLS26] Karthik Gajulapalli, Surendra Ghentiyala, Zeyong Li, and Sidhant Saraogi. Downward self-reducibility in the total function polynomial hierarchy. In *SODA*, 2026. To appear. 1

[GGNS23] Karthik Gajulapalli, Alexander Golovnev, Satyajeet Nagargoje, and Sidhant Saraogi. Range avoidance for constant depth circuits: Hardness and algorithms. In *RANDOM*, 2023. 1

[GLV24] Karthik Gajulapalli, Zeyong Li, and Ilya Volkovich. Oblivious complexity classes revisited: Lower bounds and hierarchies. In *FSTTCS*, 2024. 1

[GLW22] Venkatesan Guruswami, Xin Lyu, and Xiuhan Wang. Range avoidance for low-depth circuits and connections to pseudorandomness. In *RANDOM*, 2022. 1

[GS86] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *STOC*, 1986. 3, 9, 12, 16, 17

[HV25] Edward A. Hirsch and Ilya Volkovich. Upper and lower bounds for the linear ordering principle. arXiv:2503.19188, 2025. 1, 5

[ILW23] Rahul Ilango, Jiatu Li, and R. Ryan Williams. Indistinguishability obfuscation, range avoidance, and bounded arithmetic. In *STOC*, 2023. 1, 2, 10, 12

[IW97] Russell Impagliazzo and Avi Wigderson. P= BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *STOC*, 1997. 14

[Jeř04] Emil Jeřábek. Dual weak pigeonhole principle, Boolean complexity, and derandomization. *Annals of Pure and Applied Logic*, 129:1–37, 2004. 4

[KKMP21] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos Papadimitriou. Total functions in the polynomial hierarchy. In *ITCS*, 2021. 1, 2, 4, 5, 12

[Kor21] Oliver Korten. The hardest explicit construction. In *FOCS*, 2021. 1, 4, 5

[Kor22] Oliver Korten. Derandomization from time-space tradeoffs. In *CCC*, 2022. 1

[Kor25] Oliver Korten. Range avoidance and the complexity of explicit constructions. *Bulletin of EATCS*, 145(1), 2025. 1, 2

[KP24] Oliver Korten and Toniann Pitassi. Strong vs. weak range avoidance and the linear ordering principle. In *FOCS*, 2024. 1, 5

[KPI25] Oliver Korten, Toniann Pitassi, and Russell Impagliazzo. Stronger cell probe lower bounds via local PRGs. In *FOCS*, 2025. 1

[KS25a] Oliver Korten and Rahul Santhanam. How to construct random strings. In *CCC*, 2025. 1, 4, 10

[KS25b] Neha Kuntewar and Jayalal Sarma. Avoiding range via Turan-type bounds. In *RANDOM*, 2025. 1

[KvM99] Adam R Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. In *STOC*, 1999. 7, 14, 16

[Li24] Zeyong Li. Symmetric exponential time requires near-maximum circuit size: Simplified, truly uniform. In *STOC*, 2024. 1

[LORS24] Zhenjian Lu, Igor C. Oliveira, Hanlin Ren, and Rahul Santhanam. On the complexity of avoiding heavy elements. In *FOCS*, 2024. 1

[LZ25] Xin Li and Yan Zhong. Range avoidance and remote point for low-depth circuits: New algorithms and hardness. ECCC:2025/049/, 2025. 1

[MX10] Mohammad Mahmoody and David Xiao. On the power of randomized reductions and the checkability of SAT. In *CCC*, 2010. 15

[RSW22] Hanlin Ren, Rahul Santhanam, and Zhikun Wang. On the range avoidance problem for circuits. In *FOCS*, 2022. 1, 2

[RWZ25] Hanlin Ren, Yichuan Wang, and Yan Zhong. Hardness of range avoidance and proof complexity generators from demi-bits. arXiv:2511.14061, 2025. 1, 4, 10, 11

[Sha92] Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39:869–877, 1992. 8

[SS24] Ronen Shaltiel and Jad Silbak. Explicit codes for poly-size circuits and functions that are hard to sample on low entropy distributions. In *STOC*, 2024. 3, 7, 14

[SU06] Ronen Shaltiel and Christopher Umans. Pseudorandomness for approximate counting and sampling. *Computational Complexity*, 15(4):298–341, 2006. 14