

Derandomizing (Fragments of) Arthur–Merlin with Range Avoidance

Zeyong Li

October 24, 2026

Abstract

We study the subclass of **prAM** problems such that derandomizing the corresponding Arthur–Merlin verifier (i.e. to an **NP** verifier) reduces to the Range Avoidance problem (**Avoid**), the total search problem of finding a string outside the range of an expanding circuit.

We prove a new characterization of this subclass by showing a natural complete problem for it. The problem, which we call Structured Set Lower Bound (**SSLB**), can be seen as a restriction of the classical Set Lower Bound problem by Goldwasser and Sipser (STOC 1986), in which the (small) set is well-structured and can be compactly represented in the **NO** case. In more detail, we show that any problem in **prAM** admits a many-one reduction to **SSLB** if and only if derandomizing its verifier (via a targeted hitting set) reduces to **Avoid**, thereby providing a full characterization of the subclass.

We then show that this class constitutes rich and interesting fragments of **prAM**. First of all, we show that it contains all of **prMA**. Moreover, two important problems from complexity theory and cryptography—Graph Non-Isomorphism (GNI) and the complement of the approximate Shortest Vector Problem ($\overline{\text{GapSVP}}_\gamma$, with approximation factor $\gamma = \Theta(\sqrt{n/\log n})$)—also fall into this class. These are all well-studied problems residing in **prAM**. As a corollary, if the corresponding **Avoid** families are solvable in **SearchNP**, then GNI and GapSVP_γ are in $\text{NP} \cap \text{coNP}$, respectively.

Contents

1	Introduction	1
1.1	Instance-Wise Derandomization	1
1.2	Range Avoidance	2
1.3	Our Contributions	3
1.4	Technical Overview	5
1.5	Discussions and Related Works	7
2	Preliminaries	9
2.1	Arthur–Merlin	9
2.2	Range Avoidance	10
2.3	Permutation groups	10
2.4	Lattices and Euclidean balls	11
3	Structured Set Lower Bound	13
4	Applications	17
4.1	Merlin–Arthur	18
4.2	Graph Non-Isomorphism	18
4.3	Goldreich–Goldwasser for GapSVP	19
5	Consequences of SearchNP Solvability	23

1 Introduction

Randomness is a fundamental resource in computation, playing an important role in algorithm design, interactive proofs, communication protocols and so on. The prevalence of randomness in computation raises a fundamental question: how essential is randomness? Or can deterministic methods achieve the same goals with comparable efficiency? The task of derandomization therefore seeks to understand when such simulations are possible and at what computational cost.

For polynomial-time computation, a central question in derandomization is whether $\text{prBPP} = \text{prP}$: can every probabilistic polynomial-time algorithm be simulated deterministically with only polynomial overhead? The hardness-versus-randomness paradigm of Nisan and Wigderson [NW94] offers a general approach by constructing pseudorandom generators (PRG) from computationally hard functions. A pseudorandom generator (in this setting) deterministically expands a short seed of logarithmic length into a longer string whose distribution is indistinguishable from uniform to any adversary running in bounded polynomial time. Enumerating the (polynomially many) seeds and plugging the outputs of the PRG into the probabilistic algorithm yields a deterministic simulation with only polynomial overhead. This connection between computational hardness and derandomization has led to a rich body of work [STV01, IW01, Uma03, TV07]. In particular, Impagliazzo and Wigderson [IW97] demonstrated that $\text{prBPP} = \text{prP}$ if E contains a language of exponential circuit complexity.

Similar questions are also studied in the setting of proof systems. In particular, the question of whether $\text{prAM} = \text{prNP}$ receives a lot of attention: can every Arthur–Merlin protocol be derandomized to an NP proof? To derandomize prAM , it suffices to construct a hitting set generator (HSG) since every Arthur–Merlin protocol can be made perfectly complete [ZF87, FGM⁺89]. More specifically for a protocol with perfect completeness, every random Arthur challenge admits an accepting Merlin response on a YES instance, whereas a constant fraction of challenges admit no accepting response on a NO instance. It therefore suffices to construct (in non-deterministic polynomial time) a polynomial-size hitting set containing at least one such rejecting challenge for each NO instance. Following the same hardness-versus-randomness paradigm, there has been a long line of research [Lu01, Kv02, GST03, MV05, SU05, SU09], studying the construction of such hitting sets for derandomizing prAM .

1.1 Instance-Wise Derandomization

While a PRG (HSG, respectively) is sufficient to derandomize prBPP (prAM , respectively), it appears to be overkill. For example, for the purpose of derandomizing a probabilistic polynomial-time algorithm on a particular input, it is not necessary to fool all polynomial-time computations simultaneously (as required in the definition of PRG). Instead, it suffices to fool the particular computation and input being simulated. This observation motivates the study of instance-wise derandomization, and hence *targeted* pseudorandom generators, where the generator is allowed to ‘see the computation that it is trying to fool’. In particular, Goldreich [Gol11] (see also [Gol20]) showed that polynomial-time targeted PRGs with logarithmic seed length and sufficiently small error are both sufficient and necessary for proving $\text{prBPP} = \text{prP}$.

In recent years, many exciting results [CT21, CRT22, LP22, LP23, BCT25] have emerged following the study of targeted PRGs and targeted HSGs. Chen and Tell [CT21] developed an instance-wise hardness-versus-randomness framework for probabilistic computation, constructing targeted PRGs and HSGs from suitable uniform hardness assumptions. Chen, Tell, and Williams [CTW23] subsequently refined this targeted-HSG approach and characterized $\text{prBPP} = \text{prP}$ through efficient refuters: algorithms that find inputs on which a given probabilistic algorithm fails to compute a certain hard function. Readers interested in the development of targeted PRGs may refer to [CT23a] for an extensive survey by Chen and Tell.

There is also a line of study for the derandomization of prAM [vMS23, CT23b, vMS25]. In particular, van Melkebeek and Mcelin Sdroievski [vMS25] constructed targeted HSGs from suitable uniform hardness against Arthur–Merlin protocols, establishing analogous instance-wise hardness-versus-randomness tradeoffs.

Moreover, unlike a targeted pseudorandom set whose size must be relatively large to approximate the acceptance probability over the uniform distribution, a targeted hitting set can have size as small as one:

once the target dense set is fixed, any one of its elements suffices. This motivates the definition of the following search problem, asking for one hitting string directly, rather than a list guaranteed to contain one.

Definition 1.1 (Targeted Hitting String, informal; see Definition 3.4). *Let V be an Arthur–Merlin verifier with completeness 1. The V -Targeted Hitting String problem asks, given a NO instance x , to output an Arthur string r such that V always rejects regardless of Merlin’s response.*

Moreover, recall that the following Goldwasser–Sipser Set Lower Bound problem for NP sets [GS86] is complete for prAM [Vol20].

Definition 1.2 (Set Lower Bound; see [Vol20, Definition 2.8]). *Given a positive integer threshold τ and a polynomial-size circuit $P : \{0,1\}^n \times \{0,1\}^{\text{poly}(n)} \rightarrow \{0,1\}$, defining the NP set $S := \{s \in \{0,1\}^n : \exists w, P(s,w) = 1\}$, the task is to distinguish between the following cases:*

- YES: (Large) $|S| \geq 4\tau$;
- NO: (Small) $|S| \leq \tau$.

Therefore, solving V_{GS} -Targeted Hitting String would be sufficient to deduce $\text{prAM} = \text{prNP}$, where V_{GS} is a verifier for Set Lower Bound.

1.2 Range Avoidance

The Range Avoidance problem (Avoid), introduced in [KKMP21] is the following total search problem: given a Boolean circuit $D : \{0,1\}^m \rightarrow \{0,1\}^{m+1}$, find a string $y \in \{0,1\}^{m+1} \setminus \text{Im}(D)$. Notice that a solution is guaranteed to exist by the pigeonhole principle, and a uniformly random $(m+1)$ -bit string is a solution with probability at least $1/2$. Therefore, the algorithmic challenge is to construct such a string deterministically (or pseudodeterministically).

The importance of Avoid stems in part from its connection to explicit constructions. Korten [Kor21] (building on [Jef04]) showed that the construction of many important combinatorial objects reduces to Avoid, including constructing Ramsey graphs, two-source extractors, rigid matrices, and truth tables of high circuit complexity. The problem has since received substantial attention, with various algorithms developed for general circuits [CHLR26, GGLS26, CLL25, RW26] as well as restricted circuit classes [RSW22, GLW22, CHLR23, GGNS23, KPI25, KS25b, GLY26, HLZ26]. Complementing these algorithmic results, a growing line of work establishes conditional hardness of Avoid, concluding that Avoid is unlikely to be solvable in SearchNP [ILW23, CL24, RWZ26, Ila25, LRZ26]. We refer interested readers to Korten’s survey [Kor25] for a broader overview of the complexity of Range Avoidance.

Avoid and Derandomization In this work, we are primarily interested in the close connection between Avoid and derandomization: a deterministic polynomial-time algorithm for Avoid would imply $\text{prBPP} = \text{prP}$. Such an algorithm would construct truth tables of high circuit complexity [Kor21]. The hardness-versus-randomness framework then converts these truth tables into pseudorandom generators with logarithmic seed length [NW94, IW97]. Korten [Kor21] also gave a direct reduction from PRG construction to Avoid, by constructing an expanding circuit whose range contains every candidate sample list that fails to fool small circuits. A string outside this range therefore encodes a suitable pseudorandom set.

A straightforward (and hence folklore) way to extend this connection to the derandomization of prAM is to allow SAT gates in the circuits¹ given to Avoid. Both arguments above relativize, so a SearchNP algorithm for this oracle circuit version would yield generators that fool non-deterministic tests, and hence imply $\text{prAM} = \text{prNP}$ [Kor21].

This, however, asks us to solve a substantially harder problem. On the other hand, solving Avoid (on ordinary Boolean circuits) is a computational task very similar to the task of finding a hitting string for an Arthur–Merlin protocol:

¹These are Boolean circuits with additional SAT oracle gates that answer queries to any SAT instance. It is the circuit analogue of P^{NP} computation.

- Let C be an input circuit for **Avoid**. A solution asks for a string in $S_{\text{Avoid}} := \{y : \forall x, C(x) \neq y\}$;
- Let $V(x, r, w)$ be a verifier of an Arthur–Merlin protocol. A targeted hitting string asks for a string in $S_{\text{Hitting}} := \{r : \forall w, V(x, r, w) = 0\}$.

In particular, both of them ask for a string in a dense co-nondeterministic set: both S_{Avoid} and S_{Hitting} contain at least a constant fraction of strings, and membership in these sets can be verified by a coNP predicate.

Given such similarity, it is natural to ask what fragments of **prAM** can be derandomized if **Avoid** is solvable in, e.g. **SearchNP**. Going a bit further, it is enticing to ask the following question:

*Can we derandomize **prAM** by solving **Avoid** on ordinary Boolean circuits?*

1.3 Our Contributions

We give a partial positive answer to the above question. In particular, we identify a subclass of **prAM** with the characterization that its instance-wise white-box derandomization reduces to **Avoid**. This subclass is represented by the *Structured Set Lower Bound* problem (SSLB), a restriction of the Set Lower Bound Problem from [Definition 1.2](#).

Definition 1.3 (Structured Set Lower Bound, informal; see [Definition 3.1](#)). *An instance of Structured Set Lower Bound (SSLB) specifies an NP set² $S \subseteq \{0, 1\}^n$, a positive integer threshold τ , and a polynomial-size circuit $C : \{0, 1\}^{\lceil \log \tau \rceil} \rightarrow \{0, 1\}^n$. The task is to distinguish between the following cases:*

- YES: (Large) $|S| \geq 4\tau$;
- NO: (Small) $|S| \leq \tau$, and (Structure) $S \subseteq \text{Im}(C)$.

Note that the only difference from the classical Set Lower Bound problem is the additional (Structure) condition in the NO case: when the set S is small, we require that its elements can be (almost optimally when $|S| = \tau$) encoded using about $\log \tau$ bits. The circuit C is essentially a decoder that maps codewords to elements in S . As such, S is a subset of the image of C and elements in S can be succinctly represented by the short codewords. Hence, we consider the set S well-structured.

Since SSLB is a restriction of Set Lower Bound, it trivially lies in **prAM**.

Characterization. Our first result shows that, by leveraging the additional structure condition, the task of finding a targeted hitting string for SSLB can be reduced to **Avoid** under deterministic polynomial-time many-one reductions.

Theorem 1.4 (Informal, see [Theorem 3.5](#)). *SSLB-Targeted Hitting String reduces to **Avoid**.*

In a bit more detail, we first construct a verifier of SSLB based on the Goldwasser–Sipser protocol [[GS86](#)]. On a NO instance x , we then construct an **Avoid** instance D_x , whose range contains *every* Arthur challenge admitting an accepting Merlin response. Any string outside its range is therefore a targeted hitting string witnessing that the instance should be rejected.

In fact, SSLB captures exactly the problems for which this derandomization strategy is possible. We prove a converse: any reduction from a verifier’s Targeted Hitting String problem to **Avoid** induces a reduction from the underlying promise problem to SSLB. Together, we obtain the following equivalence.

Theorem 1.5 (Informal; see [Corollary 3.8](#)). *For any promise problem $\Pi \in \text{prAM}$, the following are equivalent:*

- (i) Π reduces to SSLB.
- (ii) Π admits a **prAM** verifier V such that V -Targeted Hitting String reduces to **Avoid**.

Thus, SSLB, and the corresponding subclass of **prAM**, is fully characterized by this instance-wise, fully white-box derandomization through **Avoid**.

²The set S is specified by an NP circuit that recognises S , which is part of the input.

Applications. As our second main contribution, we show that this class is rich and contains many natural and well-studied problems. These results demonstrate that this class is not merely a trivial or artificial construction, but rather contains important fragments of prAM that have been extensively studied in the literature.

First of all, by leveraging the equivalence above and observing that a PRG that derandomizes prBPP also derandomizes prMA , we show that this class contains all of prMA .

Theorem 1.6 (Informal; see Theorem 4.2). *Every promise problem in prMA reduces to SSLB.*

Next, we show that two well-studied computational problems from complexity theory and cryptography also belong to this class. These problems are known to be in prAM^3 , but not known to be in prNP .

Theorem 1.7 (Informal; see Theorem 4.4). *Graph Non-Isomorphism reduces to SSLB.*

Graph Isomorphism is a central problem in complexity theory and an important candidate for NP-intermediate problems⁴. Its significance extends to logic and computational group theory, as well as applications such as comparing molecular structures in chemistry and detecting symmetries in combinatorial optimization [GS20]. It admits a quasipolynomial-time algorithm [Bab16], and is known to be contained in $\text{NP} \cap \text{coAM}$ [GMW86, GS86, BM88]. But it is not known to be in P , and believed to be not NP-complete (assuming that the polynomial hierarchy does not collapse).

There has also been a lot of effort trying to place it in coNP , or equivalently place its complement Graph Non-Isomorphism in NP . Several works established this containment under strong circuit lower bound assumptions [AK01, Kv02, MV05], and our result can be seen as achieving the same goal of placing Graph Non-Isomorphism in NP assuming solvability of *Avoid*.

Theorem 1.8 (Informal; see Theorem 4.6). *$\overline{\text{GapSVP}}_\gamma$ reduces to SSLB for $\gamma(n) = \Theta(\sqrt{n/\log(n)})$.*

GapSVP (of which $\overline{\text{GapSVP}}$ is the complement) is a fundamental problem for lattice-based cryptography. In particular, Regev’s worst-case-to-average-case reduction bases the security of Learning With Errors (LWE) on the (quantum) hardness of GapSVP for a polynomial approximation factor γ [Reg09], which lays the security foundation for lattice-based cryptography. In other words, the security of lattice-based cryptography relies on the worst-case hardness of GapSVP, and it is therefore crucial to pinpoint the exact complexity of GapSVP for different approximation factors γ .

Readers interested in the complexity landscape of GapSVP_γ may look at [Ben23] for an extensive survey. Here we highlight a pair of complexity-theoretic results relevant to us: For $\gamma(n) = \Theta(\sqrt{n/\log n})$, Goldreich and Goldwasser [GG00] showed that $\text{GapSVP}_\gamma \in \text{prNP} \cap \text{prcoAM}$. Aharonov and Regev [AR05] showed that $\text{GapSVP}_\gamma \in \text{prNP} \cap \text{prcoNP}$ for approximation factor $\Theta(\sqrt{n})$.

There has been a lot of effort trying to improve the prcoNP approximation factor to $\Theta(\sqrt{n/\log n})$, some of which leads to other exciting progress [ABB⁺23]. Yet the problem itself remains open and was explicitly mentioned in [Ben23, Open Problem 4.2]. Derandomizing the Goldreich–Goldwasser protocol would close the $\Theta(\sqrt{\log n})$ gap between the prcoAM and prcoNP protocols. And our result provides a sufficient condition for doing so: solving the corresponding family of *Avoid* instances in SearchNP .

Finally, by composing the reductions, we identify two natural families of *Avoid* instances whose solvability has direct implications for the complexity landscape of important problems like Graph Isomorphism and GapSVP.

We consider these families natural since their inputs are graphs and lattices, which are natural and well-studied mathematical objects with rich structures, as compared to arbitrary Boolean circuits.

Theorem 1.9 (Informal; see Corollaries 5.2 and 5.3). *For each of Graph Isomorphism and GapSVP_γ for $\gamma(n) = \Theta(\sqrt{n/\log n})$, there is a family of *Avoid* instances whose solvability in SearchNP implies that the problem is in $\text{NP} \cap \text{coNP}$.*

³In fact, both of them are in $\text{prAM} \cap \text{prcoNP}$.

⁴A problem is NP-intermediate if it belongs to NP but is neither in P nor NP-complete.

1.4 Technical Overview

We now present some high-level ideas behind our main results.

1.4.1 Reducing Targeted Hitting String to Avoid

Let V be a prAM verifier with perfect completeness and soundness $1/3$. Recall that our goal is to reduce the task of finding a hitting string to solving **Avoid**, and it suffices to construct an expanding circuit whose range contains every answerable Arthur string on a NO instance.

Randomness in Goldwasser–Sipser Our reduction is fully white-box. Therefore, to demonstrate the intuition behind such reductions, let us start by opening up the Goldwasser–Sipser protocol and examine the role of randomness there.

For simplicity, assume that $S \subseteq \{0, 1\}^n$, the threshold $\tau = 2^{k-2}$, and $|S| \leq \tau$ on a NO input $x = (S, \tau)$. The protocol has many variants, but all of them employ a hash-based argument. In one of the simplest variants, Arthur’s challenge is a uniformly chosen pairwise independent hash function $(A, b) \in \{0, 1\}^{k \times n} \times \{0, 1\}^k$. Merlin is expected to reply with an element $s \in S$ such that $As \oplus b = 0^k$ (or equivalently $As = b$); otherwise Arthur would reject.

Completeness follows from pairwise independence: when $|S| \geq 4\tau = 2^k$, the random target b lies in $A(S)$ with probability at least $1/2$. Soundness follows from counting: when $|S| \leq \tau = 2^{k-2}$, the image $A(S)$ covers at most a quarter of the possible targets, so Arthur accepts with probability at most $1/4$.

Constructing the Avoid Circuit Therefore, let’s define the set of answerable challenges as

$$G := \{(A, b) : \exists s \in S, As = b\}.$$

All we need to do is to construct a circuit $D_x : \{0, 1\}^{kn+k-1} \rightarrow \{0, 1\}^{kn+k}$ such that $\text{Im}(D_x) \supseteq G$. This requires a compact encoding of G with at most $(kn + k - 1)$ bits, where we notice that G can be represented as

$$G = \{(A, As) : A \in \{0, 1\}^{k \times n}, s \in S\}.$$

We need exactly kn bits to describe A , so we have to enumerate all $s \in S$ with at most $k-1$ bits. Information-theoretically, this is feasible since $|S| \leq 2^{k-2}$. To achieve this computationally, we appeal to the structure promise of SSLB: the circuit C allows us to enumerate all $s \in S$ with $\lceil \log \tau \rceil = k - 2$ bits, which is exactly what we need. D_x can then be defined as

$$D_x(A, y) := (A, A \cdot C(y)).$$

Achieving Perfect Completeness There is one missing piece in the argument above: the simple hash-based protocol above does not give perfect completeness. Therefore, we need to apply standard techniques to achieve perfect completeness, by using the covering argument from [ZF87, FGM⁺89] and then performing round reduction from [BM88].

The final protocol is as follows: Arthur sends t copies of independent uniform challenges $\rho_1, \dots, \rho_t$, where each challenge $\rho_i = (A, b)$ is a pairwise independent hash function as above. Merlin replies with

1. A short list of $\ell \ll t$ challenges $\Delta_1, \dots, \Delta_\ell$;
2. For each $i \in [t]$, an index $j_i \in [\ell]$ and the answer for the shifted challenge $\rho_i \oplus \Delta_{j_i}$.

Finally Arthur accepts if and only if all t answers for $\rho_i \oplus \Delta_{j_i}$ are accepted.

We omit the analysis here as it is fairly standard, and focus on the reduction to **Avoid**. Following the same idea, for one copy of the challenge, the circuit $D_{x,i}$ can be defined as:

$$D_{x,i}(A, y, j) := (A, A \cdot C(y)) \oplus \Delta_j.$$

And we hope to concatenate t copies of $D_{x,i}$ to build D_x . Two issues immediately show up:

1. $D_{x,i}$ now maps $(kn + k - 2 + \lceil \log \ell \rceil)$ bits to $(kn + k)$ bits, which is no longer expanding. To fix this issue, notice that the initial gain of $k - (k - 2) = 2$ bits comes from the multiplicative gap (which is $4\tau/\tau = 4$) of $|S|$ between the YES and NO cases. In particular, if we have a multiplicative gap of 4^d for sufficiently large d , then we can describe y with about $k - 2d$ bits and make $D_{x,i}$ expanding. Indeed, the size gap can be easily amplified by considering the product set S^d instead of S , which is standard in the literature.
2. The list of shifts $\Delta_1, \dots, \Delta_\ell$ is also provided by Merlin, and needs to be encoded as part of the input to D_x . This introduces additional input bits and the resulting circuit may no longer be expanding. To get around this issue, the observation is that the same list of shifts can be reused across all t challenges. From the argument above, we can gain (i.e. number of output bits minus number of input bits) at least one bit per challenge. By choosing t sufficiently large, we can save enough bits to pay for the description of the common shift list.

The final **Avoid** circuit D_x is therefore the concatenation of t copies of $D_{x,i}$, with additional input bits interpreted as the list $\Delta_1, \dots, \Delta_\ell$.

1.4.2 Reducing Graph Non-Isomorphism to SSLB

To see the reduction from Graph Non-Isomorphism to SSLB, let G_0 and G_1 be graphs on n vertices. We start by recalling the classical reduction to Set Lower Bound [BHZ87]. Define

$$K_b := \{(H, \alpha) : H \cong G_b, \alpha \in \text{Aut}(H)\}, \quad K := K_0 \cup K_1.$$

Each G_b has $n!/|\text{Aut}(G_b)|$ isomorphic copies, each with $|\text{Aut}(G_b)|$ automorphisms, so $|K_b| = n!$. Thus $|K| = 2n!$ in the YES case, when the graphs are non-isomorphic, and $|K| = n!$ in the NO case, when $K_0 = K_1$.

It remains to enumerate K_0 compactly in the NO case. Notice that each element $(H, \alpha) \in K_0$ can be represented as $(\pi G_0, \pi r^{-1})$ for some permutation π , where r is a canonical representative of the coset $\pi \text{Aut}(G_0)$. If we can compute a canonical representative for each coset $\pi \text{Aut}(G_0)$ given only π , then we can enumerate K_0 by enumerating all $n!$ permutations π .

Indeed, such an algorithm is feasible given a generating set Γ for the subgroup of automorphisms $\text{Aut}(G_0)$ by Theorem 2.6. While we do not know how to compute Γ efficiently, the structure circuit can take it as part of its input. By Lemma 2.5, such a generating set can be encoded in length $L = O(\log^2(n!))$.

The remaining obstacle is that these extra L bits might overwhelm the budget of $\lceil \log \tau \rceil$ input bits. As in the preceding reduction, we overcome this by observing that the same generating set can be reused over and over again. By taking $S := K^d$, the size gap is amplified to 2^d , and we can set $\tau = 2^{d-2}(n!)^d$. Enumerating K_0^d needs about $\log((n!)^d)$ bits, by enumerating d independent permutations as outlined above (assuming access to a generating set). Therefore we have a budget of about $(d-2)$ bits to encode the generating set, which is sufficient as long as $d \gg L$.

More concretely, the structure circuit C takes one generating set Γ together with a tuple of d permutations $(\pi_1, \dots, \pi_d)$, and transforms each π_i to produce a d -tuple that belongs to K_0^d . This in particular enumerates $S = K^d$ in the NO case.

1.4.3 Reducing $\overline{\text{GapSVP}}_\gamma$ to SSLB

For the lattice reduction, we use the geometric idea behind the Goldreich–Goldwasser protocol [GG00]. For simplicity, we will work with continuous Euclidean balls, whereas the actual proof needs to handle the discretization.

Suppose that we place balls of radius $r = \gamma d/2$ at every lattice point. In the YES case, the balls are pairwise disjoint, since distinct lattice points are more than γd apart. In the NO case, there is a non-zero lattice vector $\mathbf{B}\mathbf{v}$ of length at most d , so $B_2(\mathbf{w}, r)$ overlaps heavily with $B_2(\mathbf{w} + \mathbf{B}\mathbf{v}, r)$ for any lattice vector $\mathbf{w}$.

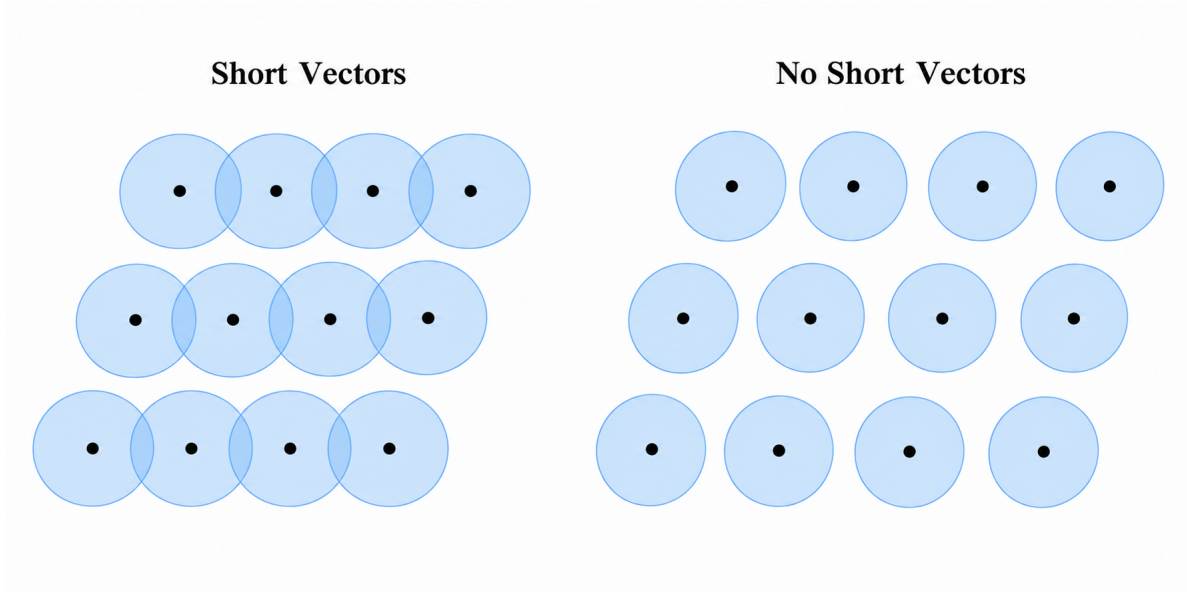


Figure 1: An illustration of the geometric idea for $\overline{\text{GapSVP}}$.

Following this intuition, for some suitably chosen hypercube $A \subseteq \mathbb{Z}^n$, define the set

$$K := \bigcup_{z \in A} (Bz + B_2(\mathbf{0}, r)).$$

In the YES case, every non-zero lattice vector has length greater than γd , so the balls are disjoint and $\text{vol}(K) = |A| \text{vol}(B_2(\mathbf{0}, r))$. In the NO case, a non-zero lattice vector $B\mathbf{v}$ of length at most d gives nearby pairs of centers Bz and $B(z + \mathbf{v})$. The corresponding balls overlap in an inverse-polynomial fraction of their volume by [Lemma 2.9](#). For sufficiently large A , we can tolerate the loss at the boundary and pair a constant fraction of lattice vectors, giving an inverse-polynomial gap in the volume of K .

It remains to turn this overlap into compact descriptions of (the discretization of) K , which can be done via dynamic programming ([Lemma 2.11](#)) and careful counting. The compact description requires knowing a short vector $B\mathbf{v}$ for pairing lattice vectors in A . While we do not know how to find $\mathbf{v}$ efficiently (which would require solving SVP), the structure circuit can take it as part of its input, just as it takes an automorphism-group generating set in the Graph Non-Isomorphism reduction. Likewise, we reuse the short vector in multiple independent copies of K to amortize the cost for encoding $\mathbf{v}$ and ensure the structure circuit meets the encoding budget.

1.4.4 The common mechanism

We would like to reiterate a common theme that comes up across all reductions. We often need to pay for some (seemingly expensive) encoding cost of some special structure (e.g. the list of shifts, the generating set of $\text{Aut}(G)$, the shortest vector). However, the same object can be reused across multiple independent challenges, and we can amortize the encoding cost over the multiple challenges, and still ensure that the resulting circuit is expanding.

1.5 Discussions and Related Works

Instance-wise, White-box Derandomization This work can be seen as a continuation of the line of study on targeted, instance-wise derandomization [[Gol11](#), [Gol20](#), [CT21](#), [CRT22](#), [LP22](#), [LP23](#), [CTW23](#), [vMS23](#), [CT23b](#), [BCT25](#), [vMS25](#)]. Moreover, our reduction is fully white-box: not only does the algorithm see

the input x , it also opens up the specific Arthur–Merlin verifier and exploits its behaviour: it leverages the concrete hash-based argument in the Goldwasser–Sipser protocol, as demonstrated in the technical overview.

Fully white-box approaches are relatively rare in the derandomization of time-bounded computation. One particular example comes from meta-complexity: Liu and Pass characterizes $\text{prBPP} = \text{prP}$ through hardness of conditional Levin–Kolmogorov complexity [LP22], where the reconstruction argument explicitly incorporates the distinguisher’s code into a short program. We do note that fully white-box derandomization has found more applications in the derandomization of space-bounded computation [SZ99, CH22, PP23, PRZ23, DPT24].

Natural family of Avoid As a by-product of our result, we identify two natural input-indexed families of Avoid instances: $\mathcal{C}_{\text{GI}}$ and $\mathcal{C}_{\text{GapSVP}}$. This type of formulation of Avoid (i.e. decoupling the input and the uniform algorithm that generates the circuit) was recently brought to attention in [LT26], in the context of algorithmic plasticity. It can be seen as a generalization of uniform Avoid studied in [Kor22, KS25a]. Prior to this work, there have been very few natural families of Avoid instances: [Ila25] studies the oracle derandomization problem formulated as Avoid instances, which treats the input as a truth table.

Our families of Avoid instances take in graphs and lattices as their inputs, which we understand better (as compared to Boolean circuits). And as we have seen in the results, the complexity of these families of Avoid is closely related to the derandomization of important problems. They are therefore good starting points for studying the hardness (or easiness) of Avoid.

Moreover, there are rich studies of random graphs and random lattices in the literature. It is therefore much more natural and convenient to define and study the average-case complexity of these Avoid families, which may be of independent interest as illustrated in [CGL⁺26, LT26].

Avoid for restricted circuits If one considers Avoid on SAT-oracle circuits as the ‘baseline’ for derandomizing prAM , then our result also fits in the line of research on Avoid with restricted circuit families [RSW22, GLW22, CHLR23, GGNS23, KPI25, KS25b, GLY26, HLZ26].

For instance, [GGNS23] demonstrated that the construction of rigid matrices reduces to Avoid on NC_3^0 circuits rather than general Boolean circuits by exploiting properties of rigid matrices. In our case, we showed that derandomization of certain prAM protocols reduces to Avoid on ordinary Boolean circuits, rather than SAT-oracle circuits, by exploiting properties of the specific protocols.

Stretch of Avoid As per Remark 3.6, the Avoid circuits we constructed for derandomizing prAM can achieve an additive stretch of any polynomial. This nicely corresponds to the property of prAM where the soundness error can be amplified to $2^{-\text{poly}(n)}$ by parallel repetitions. However, the multiplicative stretch is always $1 + o(1)$ using the current technique. This leaves open the possibility of improving the multiplicative stretch in such reductions.

Is a targeted HSG necessary for derandomizing prAM Recall that Goldreich [Gol11, Gol20] showed that a targeted PRG is both sufficient and necessary for derandomizing prBPP . On the other hand, it remains open whether a targeted HSG is necessary for the derandomization of prAM . (See [vMS25] for some recent progress.)

Our result can be seen as opening up a new direction for attacking the problem. In particular, if one can deduce SearchNP algorithms for specific Avoid families from assuming $\text{prAM} = \text{prNP}$, then one can construct targeted HSGs (at least for the subclass of prAM we consider). It is therefore a new and potentially easier path for progress, since we are attacking a weaker problem (i.e. building targeted HSGs for the subclass instead of all of prAM) with a concrete algorithmic target (i.e. designing SearchNP algorithms for specific families of Avoid).

While there has been evidence that Avoid might be hard for SearchNP [ILW23, CL24, RWZ26, Ila25, LRZ26], we note that SearchNP algorithms for Avoid are not completely beyond reach. For instance, [LT26] constructed an average-case SearchNP algorithm for Avoid based on plausible cryptographic assumptions;

[GLS26] constructed a SearchAMA algorithm for Avoid, and one might hope to derandomize it into a SearchNP algorithm.

Connection to $\text{prAM} \cap \text{prcoNP}$ Notice that both GNI and $\overline{\text{GapSVP}}_\gamma$ fall in $\text{prAM} \cap \text{prcoNP}$. This raises two natural questions for future investigation:

1. Does every problem in $\text{prAM} \cap \text{prcoNP}$ reduce to SSLB?
2. Can we identify interesting problems not in prcoNP that reduce to SSLB? Recall that all of prMA reduces to SSLB. Assuming $\text{MA} \not\subseteq \text{coNP}$, such problems should exist.

We also note that a similar phenomenon related to Avoid was known. More specifically, [GLS26] showed that every promise problem reducible to Avoid (even with polynomially many adaptive queries) is contained in $\text{prAM} \cap \text{prcoAM}$.

Toward derandomizing $\overline{\text{GapCVP}}$ As noted in Remark 4.7, we are unable to generalize the reduction from $\overline{\text{GapSVP}}$ to $\overline{\text{GapCVP}}$, despite the fact that the original prAM containment result [GG00] was proven for $\overline{\text{GapCVP}}$.

On one hand, one may interpret this as evidence that GapCVP is genuinely harder than GapSVP . On the other hand, this might be purely technical, and one might hope to obtain a better understanding of the two problems (e.g. a better reduction from GapCVP to GapSVP) through designing a reduction from GapCVP -Targeted Hitting String to Avoid.

2 Preliminaries

In this paper, all reductions considered are deterministic polynomial-time many-one reductions.

2.1 Arthur–Merlin

Definition 2.1 (Promise Merlin–Arthur). *For $0 \leq s < c \leq 1$, a promise problem $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}}) \in \text{prMA}_{c,s}$ if there are polynomials R, W and a polynomial-time predicate*

$$V : \{0, 1\}^n \times \{0, 1\}^{W(n)} \times \{0, 1\}^{R(n)} \rightarrow \{0, 1\}$$

such that

$$\begin{aligned} x \in \Pi_{\text{YES}} : & \exists w, \Pr_{r \sim \{0, 1\}^{R(n)}} [V(x, w, r) = 1] \geq c, \\ x \in \Pi_{\text{NO}} : & \forall w, \Pr_{r \sim \{0, 1\}^{R(n)}} [V(x, w, r) = 1] \leq s. \end{aligned}$$

We denote $\text{prMA} = \text{prMA}_{1,1/3}$, where the verifier has perfect completeness and soundness error at most $1/3$.

Definition 2.2 (Promise Arthur–Merlin). *For $0 \leq s < c \leq 1$, a promise problem $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}}) \in \text{prAM}_{c,s}$ if there are polynomials R, W and a polynomial-time predicate*

$$V : \{0, 1\}^n \times \{0, 1\}^{R(n)} \times \{0, 1\}^{W(n)} \rightarrow \{0, 1\}$$

such that

$$\begin{aligned} x \in \Pi_{\text{YES}} : & \Pr_{r \sim \{0, 1\}^{R(n)}} [\exists w, V(x, r, w) = 1] \geq c, \\ x \in \Pi_{\text{NO}} : & \Pr_{r \sim \{0, 1\}^{R(n)}} [\exists w, V(x, r, w) = 1] \leq s. \end{aligned}$$

This is a two-message public-coin protocol: Arthur sends the uniform string r , and Merlin chooses w after seeing r . We denote $\text{prAM} = \text{prAM}_{1,1/3}$, where the verifier has perfect completeness and soundness error at most $1/3$.

2.2 Range Avoidance

A search problem is specified by a relation $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$. On input x , the task is to find any y such that $(x, y) \in \mathcal{R}$. A promise search problem additionally specifies a set of promised inputs Π , on each of which a solution exists, and correctness is required only on this set.

For a circuit $C : \{0,1\}^\alpha \rightarrow \{0,1\}^n$, denote its image (or its range) by $\text{Im}(C) = \{y : \exists x, C(x) = y\}$.

Definition 2.3 (Range Avoidance). *In the Range Avoidance problem (Avoid) the input is a circuit $C : \{0,1\}^\alpha \rightarrow \{0,1\}^n$ where $\alpha < n$ and the goal is to find an n -bit $y \notin \text{Im}(C)$. We refer to $\alpha(n) \mapsto n$ as the stretch of the instance C .*

We use many-one reductions from any promise search problem $(\mathcal{R}, \Pi)$ to Avoid as follows: such a reduction consists of a pair of polynomial-time algorithms f and g . For every input x , $f(x)$ produces a circuit $D_x : \{0,1\}^{m(x)} \rightarrow \{0,1\}^{m(x)+1}$, and $g(x, y)$ produces a candidate solution to $\mathcal{R}$. For every promised input $x \in \Pi$ and every $y \notin \text{Im}(D_x)$, it is guaranteed that $(x, g(x, y)) \in \mathcal{R}$.

2.3 Permutation groups

For $n \in \mathbb{N}$, let $[n] := \{1, \dots, n\}$ and let $\mathfrak{S}_n$ denote the symmetric group on $[n]$. We compose permutations from right to left, so $(\sigma\eta)(i) = \sigma(\eta(i))$, and order them lexicographically by their one-line representations $(\sigma(1), \dots, \sigma(n))$. A permutation group $A \leq \mathfrak{S}_n$ is represented by a set of generators $\Gamma \subseteq \mathfrak{S}_n$, and we write $A = \langle \Gamma \rangle$. For $\pi \in \mathfrak{S}_n$, the left coset of A represented by π is

$$\pi A := \{\pi a : a \in A\}.$$

For a graph $G = ([n], E)$ and $\pi \in \mathfrak{S}_n$, let $\pi(G)$ be the graph with edge set

$$\{\{\pi(u), \pi(v)\} : \{u, v\} \in E\}.$$

The automorphism group of G is

$$\text{Aut}(G) := \{\pi \in \mathfrak{S}_n : \pi(G) = G\}.$$

Thus $H \cong G$ iff $H = \pi(G)$ for some $\pi \in \mathfrak{S}_n$, and

$$\text{Aut}(\pi(G)) = \pi \text{Aut}(G) \pi^{-1}.$$

By the orbit-stabilizer theorem, G has exactly $n! / |\text{Aut}(G)|$ distinctly labeled isomorphic copies.

Definition 2.4 (Graph Non-Isomorphism). *The Graph Non-Isomorphism problem (GNI) takes as input two graphs G_0, G_1 on the vertex set $[n]$. The input is a YES instance if $G_0 \not\cong G_1$, and a NO instance if $G_0 \cong G_1$.*

Lemma 2.5 (Generating-set bound). *Every finite group A has a generating set Γ satisfying*

$$|\Gamma| \leq \lceil \log |A| \rceil. \tag{1}$$

Proof. Start with $\Gamma = \emptyset$. While $\langle \Gamma \rangle \neq A$, adjoin an element of $A \setminus \langle \Gamma \rangle$ to Γ . Each step at least doubles the order of the generated subgroup, so the process terminates after at most $\log |A|$ steps. $\square$

Theorem 2.6 (Lexicographically least coset representative [AK02, Theorem 3.3]). *Given generators for a subgroup $A \leq \mathfrak{S}_n$ and a permutation $\pi \in \mathfrak{S}_n$, one can compute in deterministic polynomial time the lexicographically least element r of the coset πA , together with an element $\gamma \in A$ such that $\pi = r\gamma$.*

2.4 Lattices and Euclidean balls

A full-rank rational lattice is specified by a nonsingular matrix $\mathbf{B} \in \mathbb{Q}^{n \times n}$ and is defined as the additive subgroup

$$\mathcal{L}(\mathbf{B}) := \{\mathbf{B}\mathbf{z} : \mathbf{z} \in \mathbb{Z}^n\} \subseteq \mathbb{R}^n.$$

For $\mathbf{x} \in \mathbb{R}^n$, we consider the Euclidean and infinity norms

$$\|\mathbf{x}\|_2 := \left(\sum_{i=1}^n x_i^2 \right)^{1/2}, \quad \|\mathbf{x}\|_\infty := \max_{i \in [n]} |x_i|,$$

and note that

$$\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq \sqrt{n} \|\mathbf{x}\|_\infty.$$

We denote

$$B_2(\mathbf{x}, r) := \{\mathbf{y} \in \mathbb{R}^n : \|\mathbf{y} - \mathbf{x}\|_2 \leq r\}$$

as the closed Euclidean ball of radius r centered at $\mathbf{x}$.

The length of a shortest non-zero vector of a lattice $\mathcal{L}$ is

$$\lambda_1(\mathcal{L}) := \min\{\|\mathbf{y}\|_2 : \mathbf{y} \in \mathcal{L} \setminus \{0\}\}.$$

Definition 2.7 (GapSVP). *For an approximation factor $\gamma := \gamma(n) \geq 1$, an instance of GapSVP_γ consists of a basis $\mathbf{B} \in \mathbb{Q}^{n \times n}$ and a threshold $d \in \mathbb{Q}_{>0}$, with the promise that exactly one of the following holds, and the task is to distinguish the two cases:*

$$\begin{aligned} \text{YES : } & \lambda_1(\mathcal{L}(\mathbf{B})) \leq d, \\ \text{NO : } & \lambda_1(\mathcal{L}(\mathbf{B})) > \gamma d. \end{aligned}$$

In this work, we focus on the complement problem $\overline{\text{GapSVP}}_\gamma$.

Definition 2.8 ($\overline{\text{GapSVP}}$). *For an approximation factor $\gamma := \gamma(n) \geq 1$, an instance of $\overline{\text{GapSVP}}_\gamma$ consists of a basis $\mathbf{B} \in \mathbb{Q}^{n \times n}$ and a threshold $d \in \mathbb{Q}_{>0}$, with the promise that exactly one of the following holds, and the task is to distinguish the two cases:*

$$\begin{aligned} \text{YES : } & \lambda_1(\mathcal{L}(\mathbf{B})) > \gamma d, \\ \text{NO : } & \lambda_1(\mathcal{L}(\mathbf{B})) \leq d. \end{aligned}$$

Roundings. We use coordinatewise nearest-integer rounding for any vector in $\mathbb{R}^n$. In particular, for every $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$,

$$\|\text{round}(\mathbf{x}) - \mathbf{x}\|_2 \leq \frac{\sqrt{n}}{2}$$

and

$$\|\text{round}(\mathbf{x}) - \text{round}(\mathbf{y}) - (\mathbf{x} - \mathbf{y})\|_2 \leq \sqrt{n}.$$

We also use the following coefficient bound. If $\mathbf{v} \in \mathbb{Z}^n$ satisfies $\|\mathbf{B}\mathbf{v}\|_2 \leq d$, then

$$|v_i| = \left| \sum_j (\mathbf{B}^{-1})_{ij} (\mathbf{B}\mathbf{v})_j \right| \leq d \sum_j |(\mathbf{B}^{-1})_{ij}| \quad \text{for every } i \in [n].$$

The following lemma from [ABB⁺23] provides a lower bound on the volume of the overlap between two nearby Euclidean balls; see also [GG00].

Lemma 2.9 ([ABB⁺23, Lemma 2.1]). *Let $n \geq 1$ be an integer, $r > 0$, and $\mathbf{w} \in \mathbb{R}^n$ with $\|\mathbf{w}\|_2 \leq 2r$. Then*

$$\frac{\text{vol}(B_2(\mathbf{0}, r) \cap B_2(\mathbf{w}, r))}{\text{vol}(B_2(\mathbf{0}, r))} \geq \frac{1}{\sqrt{2\pi n}} \left(1 - \frac{\|\mathbf{w}\|_2^2}{4r^2}\right)^{(n+1)/2}.$$

For an integer $R \geq 0$, let

$$E_R := B_2(\mathbf{0}, R) \cap \mathbb{Z}^n.$$

The continuous estimate above transfers to these finite-grid balls.

Lemma 2.10 (Finite-grid ball overlap). *Let $R > \sqrt{n}/2$ be an integer, put $r_- := R - \sqrt{n}/2$ and $r_+ := R + \sqrt{n}/2$, and let $\mathbf{w} \in \mathbb{Z}^n$ satisfy $\|\mathbf{w}\|_2 \leq 2r_-$. Then*

$$\frac{|E_R \cap (E_R + \mathbf{w})|}{|E_R|} \geq \frac{1}{\sqrt{2\pi n}} \left(\frac{r_-}{r_+}\right)^n \left(1 - \frac{\|\mathbf{w}\|_2^2}{4r_-^2}\right)^{(n+1)/2}.$$

Proof. Associate with each $\mathbf{z} \in \mathbb{Z}^n$ its half-open unit cell $\mathbf{z} + [-1/2, 1/2)^n$. Every point in $B_2(\mathbf{0}, r_-) \cap B_2(\mathbf{w}, r_-)$ belongs to the cell of some $\mathbf{z} \in E_R \cap (E_R + \mathbf{w})$. Conversely, the union of the cells associated with E_R is contained in $B_2(\mathbf{0}, r_+)$. Since the cells are disjoint and have unit volume,

$$\begin{aligned} |E_R \cap (E_R + \mathbf{w})| &\geq \text{vol}(B_2(\mathbf{0}, r_-) \cap B_2(\mathbf{w}, r_-)), \\ |E_R| &\leq \text{vol}(B_2(\mathbf{0}, r_+)). \end{aligned}$$

Applying Lemma 2.9 at radius r_- and using

$$\frac{\text{vol}(B_2(\mathbf{0}, r_-))}{\text{vol}(B_2(\mathbf{0}, r_+))} = \left(\frac{r_-}{r_+}\right)^n$$

gives the claimed bound. $\square$

Besides an approximate lower bound on the size, we show how to efficiently count the number of elements $N = |E_R \cap (E_R + \mathbf{w})|$, rank and unrank these elements by their lexicographic order. In particular, this implies an efficient encoding and decoding scheme for the elements using $\lceil \log N \rceil$ bits. The algorithms are based on dynamic programming and similar ideas appeared in [Cov73]; see also [GvHKW20]. We include a self-contained proof for completeness.

Lemma 2.11 (Counting and indexing finite-grid balls). *Let $n \geq 1$ and $R \geq 0$ be integers, and let $\mathbf{w} \in \mathbb{Z}^n$. Set $N := |E_R \cap (E_R + \mathbf{w})|$, and let*

$$\text{rk} : E_R \cap (E_R + \mathbf{w}) \rightarrow \{0, \dots, N-1\}$$

be the lexicographic ranking bijection. There are deterministic algorithms to compute N and evaluate rk and its inverse, all running in time polynomial in n and R .

Proof. First check whether $|w_i| > 2R$ for some i . If so, the intersection is empty and $N = 0$.

Counting. Every coordinate of a point in the intersection lies in $\{-R, \dots, R\}$. For $0 \leq i \leq n$ and integer budgets $0 \leq a, b \leq R^2$, let $D_i(a, b)$ count the suffixes $(e_{i+1}, \dots, e_n) \in \mathbb{Z}^{n-i}$ satisfying

$$\sum_{h=i+1}^n e_h^2 \leq a, \quad \sum_{h=i+1}^n (e_h - w_h)^2 \leq b.$$

Set $D_i(a, b) = 0$ if either budget is negative. The empty suffix gives $D_n(a, b) = 1$ for nonnegative budgets. By examining one coordinate at a time, we have the recurrence

$$D_{i-1}(a, b) = \sum_{t=-R}^R D_i(a - t^2, b - (t - w_i)^2) \quad (1 \leq i \leq n).$$

Evaluating this recurrence for decreasing i and increasing a, b computes the entire table. In particular, $N = D_0(R^2, R^2)$.

Moreover, for any fixed prefix $(p_1, \dots, p_i)$, let $c(p_1, \dots, p_i)$ denote the number of points in the set extending such a prefix, we have that

$$c(p_1, \dots, p_i) = D_i \left(R^2 - \sum_{h=1}^i p_h^2, R^2 - \sum_{h=1}^i (p_h - w_h)^2 \right).$$

Ranking. To evaluate $\text{rk}(\mathbf{e})$ for a point $\mathbf{e}$ in the intersection, start with rank zero. For each $i = 1, \dots, n$, add the extension count $c(e_1, \dots, e_{i-1}, t)$ for every $t < e_i$. Each point lexicographically preceding $\mathbf{e}$ is counted exactly once, at its first coordinate differing from $\mathbf{e}$, so the result is $\text{rk}(\mathbf{e})$.

Unranking. To evaluate $\text{rk}^{-1}(q)$ for $0 \leq q < N$, we construct $\mathbf{e} = (e_1, \dots, e_n)$ inductively, maintaining q as the desired rank among extensions of the current prefix. Initially the prefix is empty and $c() = N$. Suppose we have chosen $(e_1, \dots, e_{i-1})$ with $0 \leq q < c(e_1, \dots, e_{i-1})$. Choose $e_i \in \{-R, \dots, R\}$ such that

$$\sum_{t=-R}^{e_i-1} c(e_1, \dots, e_{i-1}, t) \leq q < \sum_{t=-R}^{e_i} c(e_1, \dots, e_{i-1}, t).$$

The extensions form consecutive lexicographic blocks of sizes $c(e_1, \dots, e_{i-1}, t)$, so this choice exists and is unique. Subtracting the left-hand sum from q gives its rank within the chosen block, preserving the inductive invariant. After n steps, the prefix has exactly one extension, namely $\mathbf{e}$ itself, so $q = 0$ and $\mathbf{e}$ has the original input rank.

Running time. The table has at most $(n+1)(R^2+1)^2$ entries, each obtained by summing $2R+1$ terms. Every entry is at most $(2R+1)^n$ and therefore has $O(n \log(R+2))$ bits. Ranking and unranking require at most $n(2R+1)$ queries to the table. Therefore all algorithms run in time polynomial in n and R , as desired. $\square$

3 Structured Set Lower Bound

In this section, we study the complexity of the Structured Set Lower Bound problem. We start by giving a formal definition of the problem.

Definition 3.1 (Structured Set Lower Bound). *The Structured Set Lower Bound problem (SSLB) is defined as follows. An instance consists of a positive integer threshold τ , and polynomial-size circuits*

$$P : \{0, 1\}^n \times \{0, 1\}^{\text{poly}(n)} \rightarrow \{0, 1\}, \quad C : \{0, 1\}^{\lceil \log \tau \rceil} \rightarrow \{0, 1\}^n.$$

The deterministic predicate P defines an NP set

$$S := \{s \in \{0, 1\}^n : \exists w, P(s, w) = 1\}.$$

The input is promised to satisfy exactly one of the following:

- $\Pi_{\text{YES}}: |S| \geq 4\tau$;
- $\Pi_{\text{NO}}: |S| \leq \tau$ and $S \subseteq \text{Im}(C)$.

Remark 3.2. *The factor 4 in the size gap is chosen for the ease of presentation. As with Set Lower Bound, any factor $\geq 1 + 1/p$ for polynomial $p(n)$ preserves the complexity of the problem, as one can take Cartesian products of the set to amplify the gap. There is only one caveat: the structure circuit C should now be defined as $C : [(1 + 1/2p)\tau] \rightarrow \{0, 1\}^n$, since (for τ not a power of 2) $2^{\lceil \log \tau \rceil} \geq (1 + 1/p)\tau$ and one needs to be more careful about the counting (and rounding).*

Structured Set Lower Bound is a special case of Set Lower Bound, hence its membership in prAM follows from the Goldwasser–Sipser protocol and standard perfect-completeness transformations [ZF87, FGM⁺89]. We give a self-contained proof because the specific verifier will be used in the reduction to **Avoid**.

Theorem 3.3. *Structured Set Lower Bound is in $\text{prAM}_{1,1/3}$.*

Proof. Without loss of generality, assume $\tau \leq 2^{n-2}$, for otherwise there are no YES instances and Arthur trivially rejects.

Parameter Setup. Let $d := 12n$, $N := dn$, and $X := S^d \subseteq \{0, 1\}^N$. Thus

$$|X| \geq (4\tau)^d \quad \text{in the YES case,} \quad |X| \leq \tau^d \quad \text{in the NO case.}$$

Choose

$$k := 2d - 1 + \lfloor d \log \tau \rfloor = \lfloor \log(2^{2d-1} \tau^d) \rfloor.$$

Then

$$2^{2d-2} \tau^d < 2^k \leq 2^{2d-1} \tau^d.$$

Moreover, $\tau \leq 2^{n-2}$ implies $k \leq 2d - 1 + d(n - 2) = N - 1$.

A Goldwasser–Sipser challenge is a uniformly random tuple

$$\rho = (A, b) \in \mathbb{F}_2^{k \times N} \times \mathbb{F}_2^k,$$

whose description length is $R := k(N + 1)$.

Set $\ell := R + 1$ and $t := \ell R + 2$.

The Protocol. Arthur sends t independent uniform challenges $\rho_1, \dots, \rho_t \in \{0, 1\}^R$.

Merlin replies with ℓ shifts $\Delta_1, \dots, \Delta_\ell \in \{0, 1\}^R$ and, for each $i \in [t]$, an index $j_i \in [\ell]$, a tuple $\mathbf{s}_i = (s_{i,a})_{a \in [d]} \in (\{0, 1\}^n)^d$, and membership witnesses $\{w_{i,a}\}_{a \in [d]}$.

Writing $\rho_i \oplus \Delta_{j_i}$ as (A'_i, b'_i) in the challenge encoding, Arthur accepts iff, for every $i \in [t]$,

$$A'_i \mathbf{s}_i = b'_i \quad \text{and} \quad P(s_{i,a}, w_{i,a}) = 1 \quad \text{for every } a \in [d].$$

Correctness Analysis. Define the set of answerable challenges

$$G := \{(A, b) : \exists x \in X, Ax = b\} \subseteq \{0, 1\}^R.$$

For a random challenge $\rho = (A, b)$, let

$$Z_\rho := \sum_{x \in X} \mathbf{1}_{\{Ax=b\}} = |\{x \in X : Ax = b\}|.$$

The indicators in this sum are pairwise independent. Hence, writing $M := |X|$,

$$\mathbb{E}[Z_\rho] = \frac{M}{2^k} \quad \text{and} \quad \text{Var}(Z_\rho) \leq \frac{M}{2^k}.$$

In the YES case, $\frac{M}{2^k} \geq 2$, so Chebyshev's inequality gives

$$\Pr[\rho \notin G] = \Pr[Z_\rho = 0] \leq \frac{1}{2}.$$

In the NO case, Markov's inequality gives

$$\Pr[\rho \in G] = \Pr[Z_\rho > 0] \leq \mathbb{E}[Z_\rho] < 2^{2-2d}.$$

Consequently,

$$|G| \geq 2^{R-1} \quad \text{in the YES case,} \quad |G| < 2^{R+2-2d} \quad \text{in the NO case.}$$

Completeness. On a YES instance, suppose $\Delta_1, \dots, \Delta_\ell \in \{0, 1\}^R$ are chosen independently and uniformly at random. Fix a challenge $\rho \in \{0, 1\}^R$. With probability at most $\frac{1}{2^\ell}$, ρ is not contained in $G \oplus \Delta_j$ for any $j \in [\ell]$. Taking a union bound over all challenges, the probability that any challenge is not contained in $G \oplus \Delta_j$ for any $j \in [\ell]$ is at most

$$2^R 2^{-\ell} = \frac{1}{2}.$$

Therefore, there exists a list of shifts such that

$$\bigcup_{j \in [\ell]} (G \oplus \Delta_j) = \{0, 1\}^R.$$

Merlin uses this list. Every challenge can then be shifted into G . Recall that challenges in G are answerable and Arthur would accept given the correct message from Merlin. Hence, the protocol has perfect completeness.

Soundness. Fix any shift list $(\Delta_1, \dots, \Delta_\ell)$ and consider the set of challenges that can be answered using this list:

$$\bigcup_{j \in [\ell]} (G \oplus \Delta_j).$$

On a NO instance, this set has density at most $\eta := \ell 2^{2-2d} \leq 1/2$ for sufficiently large n . For a fixed shift list, acceptance requires all t independent challenges to lie in the set above, which happens with probability at most η^t . Taking a union bound over all $2^{\ell R}$ possible shift lists, we have

$$\Pr[\text{Arthur accepts}] \leq 2^{\ell R} \eta^t \leq \frac{1}{4} < \frac{1}{3}.$$

□

Next, we define the Targeted Hitting String problem, which intuitively asks for a *single* hitting string on a NO instance of a promise problem in $\text{prAM}_{1,1/3}$.

Definition 3.4 (Targeted Hitting String). *Let*

$$V : \{0, 1\}^n \times \{0, 1\}^{R(n)} \times \{0, 1\}^{W(n)} \rightarrow \{0, 1\}$$

be a fixed verifier witnessing some promise problem $\Pi \in \text{prAM}_{1,1/3}$. The V -Targeted Hitting String problem (V-THS) is the following promise search problem. The input is a string x promised to lie in Π_{NO} , and a solution is a string $r \in \{0, 1\}^R$ such that

$$\forall w \in \{0, 1\}^W, \quad V(x, r, w) = 0.$$

When the verifier is clear from context, we also call this the Π -Targeted Hitting String problem.

We now show our main theorem, which says that finding a targeted hitting string for the Structured Set Lower Bound verifier reduces to solving the Range Avoidance problem. In particular, if a promise problem $\Pi \in \text{prAM}$ reduces to Structured Set Lower Bound, then derandomizing Π through a targeted hitting string reduces to solving Range Avoidance.

Theorem 3.5. *SSLB-THS reduces to Avoid, for the verifier V constructed in the proof of [Theorem 3.3](#).*

Proof. Let $x = (\tau, P, C)$ be a NO instance of Structured Set Lower Bound. We will construct an Avoid instance, a circuit $D_x : \{0, 1\}^{Q-1} \rightarrow \{0, 1\}^Q$ such that every non-image of D_x is a targeted hitting string for the verifier V on input x .

Parameter Setup. We follow the parameters d, N, k, R, ℓ, t from the protocol in [Theorem 3.3](#).

Set

$$c := \lceil \log \tau \rceil, \quad h := \lceil \log \ell \rceil, \quad Q := tR.$$

The parameter bounds above give $h \leq d - 2$ for sufficiently large n and $k - dc \geq d - 1$. Therefore we have

$$k - dc - h \geq 1. \quad (2)$$

Construction: the Avoid instance. We construct the circuit D_x with Q output bits. Its input contains a shift list

$$\Delta_1, \dots, \Delta_\ell \in \{0, 1\}^R$$

and, for every $i \in [t]$,

$$j_i \in [\ell], \quad u_{i,1}, \dots, u_{i,d} \in \{0, 1\}^c, \quad A'_i \in \mathbb{F}_2^{k \times N}.$$

The index j_i is encoded using h bits, with invalid encodings interpreted as the index 1.

For every $i \in [t]$, the circuit computes

$$\mathbf{s}_i := (C(u_{i,1}), \dots, C(u_{i,d})) \in \{0, 1\}^N, \quad b'_i := A'_i \mathbf{s}_i,$$

and outputs the challenge

$$\rho_i := (A'_i, b'_i) \oplus \Delta_{j_i}.$$

The output of D_x is interpreted as Arthur's public random challenge strings $(\rho_1, \dots, \rho_t) \in \{0, 1\}^Q$.

The input length of D_x is $\ell R + t(h + dc + kN)$. Recall that

$$\begin{aligned} Q - (\ell R + t(h + dc + kN)) &= tR - (\ell R + t(h + dc + kN)) \\ &= t(R - (h + dc + kN)) - \ell R \\ &= t(k - (h + dc)) - \ell R \\ &\geq t - \ell R = 2. \end{aligned}$$

Hence D_x is expanding as desired. The circuit has polynomial size which follows from the construction above.

Correctness. Call an Arthur string in $\{0, 1\}^Q$ non-hitting if some Merlin response makes the verifier accept. Fix such a string $\boldsymbol{\rho}$ and an accepting response. It suffices to show that $\boldsymbol{\rho} \in \text{Im}(D_x)$, since then every non-hitting string is not a solution to the **Avoid** instance.

The accepting response supplies a shift list, indices j_i , tuples $\mathbf{s}_i = (s_{i,1}, \dots, s_{i,d}) \in S^d$, and membership witnesses.

Since this is a promised NO instance, $S \subseteq \text{Im}(C)$, so for every $i \in [t]$ and $a \in [d]$ there is a string $u_{i,a} \in \{0, 1\}^c$ such that $C(u_{i,a}) = s_{i,a}$.

Write

$$\rho_i \oplus \Delta_{j_i} = (A'_i, b'_i).$$

The verifier's hash check implies

$$b'_i = A'_i \mathbf{s}_i.$$

Therefore, feeding the shifts, indices, strings $u_{i,a}$, and matrices A'_i to D_x reproduces $\boldsymbol{\rho}$. Hence every non-hitting string belongs to $\text{Im}(D_x)$. Any solution to the resulting **Avoid** instance is therefore a targeted hitting string. $\square$

Remark 3.6. Consider the **Avoid** instance $D_x : \{0, 1\}^{\ell_{in}} \rightarrow \{0, 1\}^{\ell_{out}}$ constructed in the above proof. We note that the additive stretch $\ell_{out} - \ell_{in}$ can be made an arbitrarily large polynomial by increasing the number of challenges t in the verifier. However, the multiplicative stretch ℓ_{out}/ℓ_{in} is always bounded by $1 + o(1)$.

We now show a ‘converse’ statement: Consider any promise problem $\Pi \in \text{prAM}$. If Π -THS reduces to solving Range Avoidance, then Π reduces to Structured Set Lower Bound.

Theorem 3.7. *Let $\Pi \in \text{prAM}_{1,1/3}$ be a promise problem, and fix a verifier V witnessing this membership. If the V -Targeted Hitting String problem reduces to Avoid, then Π reduces to Structured Set Lower Bound.*

Proof. Let (f, g) be the assumed reduction. I.e. on input x ,

$$f(x) = D_x, \quad D_x : \{0, 1\}^m \rightarrow \{0, 1\}^{m+1}.$$

And given an Avoid solution $y \notin \text{Im}(D_x)$, $g(x, y)$ outputs a targeted hitting string for which the verifier V rejects on input x .

We define the following set

$$T_x := \{y \in \{0, 1\}^{m+1} : \exists w, V(x, g(x, y), w) = 1\}.$$

The predicate P for deciding membership in T_x follows by the above definition.

If $x \in \Pi_{\text{YES}}$, perfect completeness implies $T_x = \{0, 1\}^{m+1}$, and hence $|T_x| = 2^{m+1}$.

If $x \in \Pi_{\text{NO}}$, correctness of the reduction implies

$$T_x \subseteq \text{Im}(D_x).$$

Indeed, every $y \notin \text{Im}(D_x)$ must be mapped by g to a targeted hitting string, on which V rejects.

Define

$$S_x := T_x \times T_x,$$

and set the threshold to $\tau := 2^{2m}$. Define the circuit C_x covering S_x as:

$$C_x(u_1, u_2) := (D_x(u_1), D_x(u_2)), \quad C_x : \{0, 1\}^{2m} \rightarrow \{0, 1\}^{2(m+1)}.$$

If $x \in \Pi_{\text{YES}}$, then

$$|S_x| = 2^{2(m+1)} = 4\tau.$$

If $x \in \Pi_{\text{NO}}$, then

$$|S_x| \leq |\text{Im}(D_x)|^2 \leq 2^{2m} = \tau \quad \text{and} \quad S_x \subseteq \text{Im}(C_x).$$

□

Corollary 3.8 (Equivalence). *For a promise problem $\Pi \in \text{prAM}_{1,1/3}$, the following are equivalent:*

- (i) Π reduces to Structured Set Lower Bound.
- (ii) There exists a verifier V witnessing $\Pi \in \text{prAM}_{1,1/3}$ such that the V -Targeted Hitting String problem reduces to Avoid.

All reductions are deterministic polynomial-time many-one reductions.

4 Applications

We now demonstrate that the class represented by Structured Set Lower Bound is rich and contains many interesting problems. In particular, it contains all of prMA . Moreover, two natural promise problems in prAM reduce to Structured Set Lower Bound, and hence the task of derandomizing them reduces to Range Avoidance.

4.1 Merlin–Arthur

Given the equivalence from [Corollary 3.8](#), we can show that every promise problem in prMA reduces to SSLB . This is not too surprising in retrospect, since the derandomization of prBPP through [Avoid](#) from [\[Kor21\]](#) also applies to prMA . In particular, we need the following theorem.

Theorem 4.1 ([\[Kor21, Theorem 3\]](#)). *Consider the search problem PRG: given 1^n , output a polynomial-length list $(r_1, \dots, r_t)$ of n -bit strings such that every Boolean circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}$ of size at most n satisfies*

$$\left| \frac{1}{t} \sum_{j=1}^t C(r_j) - \Pr_{r \sim \{0,1\}^n} [C(r) = 1] \right| \leq \frac{1}{n}.$$

The problem PRG has a deterministic polynomial-time many-one reduction to Avoid with $t = n^6$.

Theorem 4.2. *Every promise problem in prMA has a deterministic polynomial-time many-one reduction to SSLB .*

Proof. By perfect completeness for prMA [\[ZF87\]](#), fix a verifier $V(x, w, r)$ for Π with perfect completeness and soundness $1/3$, where w has length W and r has length R . Choose a sufficiently large polynomially bounded N such that every test $C_{x,w}(r) := V(x, w, r)$ has circuit size at most N and $W \leq N^6$. Set $t := N^6$. Define an Arthur–Merlin verifier $\hat{V}$ as follows: Arthur sends t independent uniform strings $r_1, \dots, r_t \in \{0, 1\}^R$, Merlin responds with a single witness w , and Arthur accepts iff $V(x, w, r_j) = 1$ for every j . Perfect completeness is preserved, and on a NO instance a union bound over witnesses gives soundness at most $2^W 3^{-t} \leq 1/3$ for sufficiently large t .

Apply the reduction in [Theorem 4.1](#) to 1^N . Any solution to the resulting [Avoid](#) instance yields a list $(r_1, \dots, r_t)$ that fools every size- N circuit with error $1/N$. If $x \in \Pi_{\text{NO}}$, then for every witness w ,

$$\frac{1}{t} \sum_{j=1}^t C_{x,w}(r_j) \leq \Pr_{r \sim \{0,1\}^R} [V(x, w, r) = 1] + \frac{1}{N} \leq \frac{1}{3} + \frac{1}{N} \leq \frac{2}{3}.$$

Thus no witness makes $\hat{V}$ accept this challenge. This reduces the $\hat{V}$ -Targeted Hitting String problem to [Avoid](#), so [Corollary 3.8](#) implies that Π reduces to SSLB . $\square$

4.2 Graph Non-Isomorphism

Theorem 4.3 ([\[GMW86, GS86, BM88\]](#)). *Graph Non-Isomorphism is in prAM .*

Theorem 4.4. *Graph Non-Isomorphism reduces to SSLB .*

Proof. Let (G_0, G_1) be a pair of graphs on n vertices.

The set. For $b \in \{0, 1\}$, define

$$K_b := \{(H, \alpha) : H \cong G_b, \alpha \in \text{Aut}(H)\}, \quad K := K_0 \cup K_1.$$

All graphs in these sets have vertex set $[n]$. Set

$$M := n!, \quad m := \lceil \log M \rceil, \quad L := m^2, \quad d := L + 2.$$

The set and threshold in the SSLB instance are

$$S := K^d, \quad \tau := 2^L M^d.$$

Membership for K_b can be checked by a non-deterministic predicate by guessing an isomorphism from G_b to H and checking that α is an automorphism of H . Membership for S follows similarly.

Cardinalities in the YES and NO cases. Let $a_b := |\text{Aut}(G_b)|$. Note that G_b has M/a_b distinctly labeled isomorphic copies, each with a_b automorphisms. Thus $|K_b| = M$ for both $b \in \{0, 1\}$.

In the YES case, $G_0 \not\cong G_1$, so K_0 and K_1 are disjoint and

$$|S| = (2M)^d = 2^{L+2}M^d = 4\tau.$$

In the NO case, $G_0 \cong G_1$, so $K = K_0 = K_1$ and

$$|S| = M^d \leq \tau.$$

The structure circuit. We construct the structure circuit C as follows: its input consists of a candidate generating set for $\text{Aut}(G_0)$, and d permutations. Each permutation will be encoded by its rank⁵, which requires $\lceil \log M^d \rceil$ bits jointly. By Lemma 2.5, every subgroup of $\mathfrak{S}_n$ has at most m generators. Thus $L = m^2$ bits suffice to encode such a list. The circuit has exactly

$$\lceil \log \tau \rceil = L + \lceil \log M^d \rceil$$

input bits. The first L bits encode a list Γ of m permutations. The remaining bits encode a tuple of d permutations $(\pi_1, \dots, \pi_d) \in (\mathfrak{S}_n)^d$.

Invalid encodings or lists containing a non-automorphism of G_0 are sent to a fixed output.

On a valid input, let $A := \langle \Gamma \rangle$. For each $i \in [d]$, apply Theorem 2.6 to (A, π_i) to compute the canonical representative r_i of $\pi_i A$ and the element $\gamma_i \in A$ satisfying $\pi_i = r_i \gamma_i$. The circuit outputs

$$C(\Gamma, \pi_1, \dots, \pi_d) := (r_i(G_0), r_i \gamma_i r_i^{-1})_{i \in [d]}.$$

Coverage in the NO case. Suppose $G_0 \cong G_1$, so $S = K_0^d$. Fix a generating list Γ^* for $A = \text{Aut}(G_0)$ that fits in the first L input bits. We show that the slice with $\Gamma = \Gamma^*$ covers S . Let $((H_i, \alpha_i))_{i \in [d]}$ be any element of K_0^d . For each i , the permutations mapping G_0 to H_i form a single left coset of A ; let r_i be its canonical representative. Define

$$\gamma_i := r_i^{-1} \alpha_i r_i \in A, \quad \pi_i := r_i \gamma_i.$$

Then $\pi_i A = r_i A$, so the circuit recovers exactly r_i and γ_i . Its i th output pair is therefore

$$(r_i(G_0), r_i \gamma_i r_i^{-1}) = (H_i, \alpha_i).$$

Every tuple $(\pi_1, \dots, \pi_d)$ has a valid joint encoding, proving $S \subseteq \text{Im}(C)$ in the NO case. $\square$

4.3 Goldreich–Goldwasser for GapSVP

Theorem 4.5 ([GG00]). *There is an approximation factor $\gamma(n) = \Theta(\sqrt{n/\log n})$ such that $\overline{\text{GapSVP}}_\gamma$ is in prAM.*

Theorem 4.6. *There is a $\gamma(n) = \Theta(\sqrt{n/\log n})$ such that $\overline{\text{GapSVP}}_\gamma$ reduces to SSLB.*

Proof. Let $\mathbf{B} \in \mathbb{Q}^{n \times n}$ be a nonsingular lattice basis and let $d \in \mathbb{Q}_{>0}$. By scaling $\mathbf{B}$ to $\mathbf{B}/d$, we may assume without loss of generality that $d = 1$.

⁵Encoding from and decoding into the one-line representation can be done in polynomial time.

Discretization. Fix positive integer constants c_0 and $c_1 \geq 2$ as specified below. Let $\gamma(n) := 400 \lceil \sqrt{\frac{n}{\log n}} \rceil$.

Set $Q := n^{c_1}$ to be the scaling factor. For the purpose of discretization, we will scale the space (and the lattice) by Q and work with vectors in $\mathbb{Z}^n$.

Define

$$R := \lfloor \frac{\gamma(n)Q}{100} \rfloor, \quad E := \{\mathbf{e} \in \mathbb{Z}^n : \|\mathbf{e}\|_2 \leq R\}, \quad M := |E|.$$

The set E discretizes an ambient ball of radius roughly $\gamma(n)/100$ (before scaling by Q); note that its points need not be lattice vectors. For $\mathbf{z} \in \mathbb{Z}^n$, define the rounded lattice vectors

$$\phi(\mathbf{z}) := \text{round}(Q\mathbf{B}\mathbf{z}) \in \mathbb{Z}^n,$$

using the rounding convention from the preliminaries. To choose a finite box of lattice vectors, set

$$V := \max \left\{ 1, \lceil \max_i \sum_j |(\mathbf{B}^{-1})_{ij}| \rceil \right\},$$

so that $\|\mathbf{v}\|_\infty \leq V$ whenever $\mathbf{v} \in \mathbb{Z}^n$ and $\|\mathbf{B}\mathbf{v}\|_2 \leq 1$. Let N be the least power of two at least $64n(V+1)$, and put

$$A := \{0, \dots, N-1\}^n, \quad H := |A| = N^n.$$

The numbers V and N have polynomial bit length, and M is computable in polynomial time by [Lemma 2.11](#) with $\mathbf{w} = \mathbf{0}$, since R is polynomially bounded.

The set. We construct $S := K^k$, where K consists of rounded lattice vectors plus short ambient errors. Define

$$f : A \times E \rightarrow \mathbb{Z}^n, \quad f(\mathbf{z}, \mathbf{e}) := \phi(\mathbf{z}) + \mathbf{e},$$

and let $K := f(A \times E)$. We encode elements of K using a fixed polynomial-length bit string, since $\|\phi(\mathbf{z}) + \mathbf{e}\|_\infty \leq QN \max_i \sum_j |\mathbf{B}_{ij}| + R + 1$ can be bounded by some fixed polynomial-size description.

To specify the set and the threshold, let

$$T := \lfloor M/n^{c_0} \rfloor, \quad U := HM - \frac{HT}{4},$$

and

$$a := n \lceil \log(2V+1) \rceil, \quad k := 8n^{c_0}(a+2).$$

Choose sufficiently large c_1 such that $M/n^{c_0} \geq 2$ and hence $T \geq M/(2n^{c_0})$.

The SSLB instance uses

$$S := K^k, \quad \tau := 2^a U^k.$$

The membership circuit P for S checks a tuple of k points: for each point, its witness supplies $(\mathbf{z}, \mathbf{e}) \in A \times E$, and P checks that the point equals $f(\mathbf{z}, \mathbf{e})$. This gives an NP predicate for S .

Cardinality in the YES case. Suppose $\lambda_1(\mathcal{L}(\mathbf{B})) > \gamma(n)$. If $f(\mathbf{z}, \mathbf{e}) = f(\mathbf{z}', \mathbf{e}')$, the rounding error gives

$$Q\|\mathbf{B}(\mathbf{z} - \mathbf{z}')\|_2 \leq 2R + \sqrt{n} < \gamma(n)Q.$$

Consequently, $\mathbf{z} = \mathbf{z}'$, since otherwise $\mathbf{B}(\mathbf{z} - \mathbf{z}')$ would be a non-zero lattice vector shorter than $\gamma(n)$. It then follows that $\mathbf{e} = \mathbf{e}'$. Thus f is injective and

$$|K| = HM. \tag{3}$$

Our choice of k gives

$$\left(\frac{HM}{U} \right)^k = \left(1 - \frac{T}{4M} \right)^{-k} \geq \exp\left(\frac{kT}{4M} \right) \geq e^{a+2} \geq 2^{a+2}. \tag{4}$$

Combining (3) and (4), we obtain

$$|S| = (HM)^k \geq 4 \cdot 2^a U^k = 4\tau.$$

Cardinality in the NO case. Suppose $\lambda_1(\mathcal{L}(\mathbf{B})) \leq 1$. Choose a non-zero $\mathbf{v} \in \mathbb{Z}^n$ with $\|\mathbf{B}\mathbf{v}\|_2 \leq 1$. Without loss of generality, assume that its first non-zero coordinate v_j is positive, for otherwise we could consider $-\mathbf{v}$ instead.

For every $\mathbf{z}$ such that $(\mathbf{z}, \mathbf{z} + \mathbf{v}) \in A \times A$, let

$$\mathbf{w}_{\mathbf{z}} := \phi(\mathbf{z} + \mathbf{v}) - \phi(\mathbf{z}).$$

Since $Q\|\mathbf{B}\mathbf{v}\|_2 \leq Q$, the rounding error gives $\|\mathbf{w}_{\mathbf{z}}\|_2 \leq Q + \sqrt{n}$.

We now establish the overlap estimate and fix c_0 . There is a positive integer constant c_0 such that, for sufficiently large c_1 , every $\mathbf{w} \in \mathbb{Z}^n$ with $\|\mathbf{w}\|_2 \leq Q + \sqrt{n}$ satisfies

$$|E \cap (E + \mathbf{w})| \geq \frac{M}{n^{c_0}}. \quad (5)$$

Indeed, set $r_- := R - \sqrt{n}/2$ and $r_+ := R + \sqrt{n}/2$. Our choice of R and Q satisfies $R \geq 4Q$ and $Q \geq 2\sqrt{n}$. Thus $Q + \sqrt{n} \leq r_-/2$. By Lemma 2.10,

$$\frac{|E \cap (E + \mathbf{w})|}{M} \geq \frac{1}{\sqrt{2\pi n}} \left(\frac{r_-}{r_+}\right)^n \left(1 - \frac{\|\mathbf{w}\|_2^2}{4r_-^2}\right)^{(n+1)/2}.$$

We bound the two loss terms as follows, by taking the logarithms of their reciprocals:

$$\begin{aligned} -\frac{n+1}{2} \ln\left(1 - \frac{\|\mathbf{w}\|_2^2}{4r_-^2}\right) &\leq \frac{(n+1)\|\mathbf{w}\|_2^2}{4r_-^2} = O\left(\frac{n}{\gamma(n)^2}\right) = O(\log n), \\ n \ln\left(\frac{r_+}{r_-}\right) &\leq \frac{n\sqrt{n}}{r_-} = o(1), \end{aligned}$$

where the first inequality uses $-\ln(1-x) \leq 2x$ for $0 \leq x \leq 1/16$, and the second inequality uses $\ln(1+x) \leq x$ for $x \geq 0$. And therefore (5) holds for some constant c_0 .

By (5), the lens $E \cap (E + \mathbf{w}_{\mathbf{z}})$ has at least T elements, each satisfying

$$f(\mathbf{z}, \mathbf{e}) = f(\mathbf{z} + \mathbf{v}, \mathbf{e} - \mathbf{w}_{\mathbf{z}}). \quad (6)$$

Hence, we gain T elements given one overlap (witnessed by $(\mathbf{z}, \mathbf{z} + \mathbf{v}) \in A \times A$). We next show that there are many overlaps, by lower bounding the number of disjoint pairs $(\mathbf{z}, \mathbf{z} + \mathbf{v}) \in A \times A$. Define a subset $A_{\mathbf{v}} \subseteq A$ as:

$$A_{\mathbf{v}} := \{\mathbf{z} \in A : \mathbf{z} + \mathbf{v} \in A, [z_j/v_j] \equiv 0 \pmod{2}\}.$$

Notice that by the parity condition, if $\mathbf{z} \in A_{\mathbf{v}}$, then $\mathbf{z} + \mathbf{v} \notin A_{\mathbf{v}}$. We therefore proceed to lower bound $|A_{\mathbf{v}}|$.

Let I_i consist of the integers $\max\{0, -v_i\} \leq t < \min\{N, N - v_i\}$, with the additional restriction $[t/v_j] \equiv 0 \pmod{2}$ when $i = j$. Then $A_{\mathbf{v}} = I_1 \times \cdots \times I_n$. For $i \neq j$, we have $|I_i| = N - |v_i|$. In coordinate j , the interval is $\{0, \dots, N - v_j - 1\}$, and the parity condition keeps at least half the entries and $|I_j| \geq (N - v_j)/2$. Consequently,

$$|A_{\mathbf{v}}| = \prod_{i=1}^n |I_i| \geq \frac{1}{2} \prod_{i=1}^n (N - |v_i|) \geq \frac{H}{4},$$

where the last inequality uses $|v_i| \leq V$ and $N \geq 64n(V+1)$.

Recall that HM is an overestimate of $|K|$ assuming $E + \phi(\mathbf{z})$ are disjoint for every $\mathbf{z} \in A$, and for every $\mathbf{z} \in A_{\mathbf{v}}$, at least T elements in $(E + \phi(\mathbf{z})) \cap E + \phi(\mathbf{z} + \mathbf{v})$ are double counted. Therefore we can upper bound $|K|$ by

$$|K| \leq HM - |A_{\mathbf{v}}|T \leq HM - \frac{HT}{4} = U.$$

And hence $|S| \leq U^k \leq \tau$.

The structure circuit. The circuit C has exactly

$$\lceil \log \tau \rceil = a + \lceil \log U^k \rceil$$

input bits.

Its first a bits encode a candidate vector $\mathbf{v} \in [-V, V]^n$, expected to represent a short non-zero lattice vector as in the cardinality analysis above. This vector will be used in all k coordinates. The circuit checks that $\mathbf{v} \neq \mathbf{0}$, its first non-zero coordinate is positive, and $\|\mathbf{B}\mathbf{v}\|_2 \leq 1$. Invalid encodings and failed checks are sent to an arbitrarily fixed output.

For a valid $\mathbf{v}$, we remove the repeated descriptions counted above from $A \times E$. For each $\mathbf{z} \in A_{\mathbf{v}}$, take the first T points $\mathbf{e}$ of $E \cap (E + \mathbf{w}_{\mathbf{z}})$ in lexicographic order, and delete $(\mathbf{z}, \mathbf{e})$ from $A \times E$. Let $\mathcal{R}_{\mathbf{v}} \subseteq A \times E$ be the set of remaining descriptions. We have

$$|\mathcal{R}_{\mathbf{v}}| = HM - |A_{\mathbf{v}}|T \leq U.$$

The remaining $\lceil \log U^k \rceil$ input bits suffice to encode $\mathcal{R}_{\mathbf{v}}^k$.

It remains to show that $\mathcal{R}_{\mathbf{v}}$ can be unranked (using lexicographic order) in polynomial time. For every $\mathbf{z} \in A$, the block of descriptions with coefficient $\mathbf{z}$ has size

$$|\{\mathbf{e} \in E : (\mathbf{z}, \mathbf{e}) \in \mathcal{R}_{\mathbf{v}}\}| = \begin{cases} M - T, & \mathbf{z} \in A_{\mathbf{v}}, \\ M, & \mathbf{z} \notin A_{\mathbf{v}}. \end{cases}$$

For any fixed description $(\mathbf{z}, \mathbf{e})$, we show how to count the number of descriptions preceding $(\mathbf{z}, \mathbf{e})$ in $\mathcal{R}_{\mathbf{v}}$. In particular, the number of coefficients preceding $\mathbf{z}$ can be computed as

$$|\{\mathbf{z}' \in A : \mathbf{z}' <_{\text{lex}} \mathbf{z}\}| = \sum_{i=1}^n z_i N^{n-i},$$

where the i th summand counts coefficients whose first difference from $\mathbf{z}$ is at coordinate i . Using $A_{\mathbf{v}} = I_1 \times \cdots \times I_n$, the corresponding count is

$$|\{\mathbf{z}' \in A_{\mathbf{v}} : \mathbf{z}' <_{\text{lex}} \mathbf{z}\}| = \sum_{i=1}^n \left(\prod_{h < i} \mathbb{1}_{\{z_h \in I_h\}} \right) |\{t \in I_i : t < z_i\}| \prod_{h > i} |I_h|.$$

It remains to count the number of descriptions with coefficient $\mathbf{z}$ that precede $(\mathbf{z}, \mathbf{e})$. If $\mathbf{z} \notin A_{\mathbf{v}}$, the number of preceding descriptions is simply the rank of $\mathbf{e}$ in E . Otherwise, we subtract from this rank

$$\min\{T, |\{\mathbf{e}' \in E \cap (E + \mathbf{w}_{\mathbf{z}}) : \mathbf{e}' <_{\text{lex}} \mathbf{e}\}|\},$$

which can be computed via [Lemma 2.11](#).

Now to compute the q th description in $\mathcal{R}_{\mathbf{v}}$, one can perform a binary search over $A \times E$ by using $O(\log(HM))$ queries.

Coverage in the NO case. Suppose $\lambda_1(\mathcal{L}(\mathbf{B})) \leq 1$, and fix a valid short vector $\mathbf{v}$. Every deleted description $(\mathbf{z}, \mathbf{e})$ has the same image as $(\mathbf{z} + \mathbf{v}, \mathbf{e} - \mathbf{w}_{\mathbf{z}})$ by (6). The latter remains in $\mathcal{R}_{\mathbf{v}}$, since the parity condition makes $A_{\mathbf{v}}$ and $A_{\mathbf{v}} + \mathbf{v}$ disjoint. Consequently,

$$f(\mathcal{R}_{\mathbf{v}}) = K, \quad |\mathcal{R}_{\mathbf{v}}| \leq U. \tag{7}$$

By (7), fixing this $\mathbf{v}$ and varying the joint rank covers $f(\mathcal{R}_{\mathbf{v}})^k = K^k = S$. Thus $S \subseteq \text{Im}(C)$, as desired. $\square$

Remark 4.7. The argument does not directly extend to $\overline{\text{GapCVP}}_{\gamma}$: a YES instance guarantees that the target is far from the lattice, but gives no lower bound on the lengths of non-zero lattice vectors. Short lattice vectors can therefore cause heavy overlap among the translates of E forming K , so the injectivity argument above need not apply. We do not know how to estimate $|K|$ accurately enough to establish the required cardinality gap.

5 Consequences of SearchNP Solvability

We now record a consequence of solving only the particular **Avoid** families obtained through our reductions. Following the convention in the Range Avoidance literature [CL24, LT26], a total search problem is in **SearchNP** if there is a non-deterministic polynomial-time machine that has an accepting computation path on every input and such that every accepting path outputs a valid solution.

Let f be an algorithm that produces an expanding circuit

$$f(x) = D_x : \{0, 1\}^{m(x)} \rightarrow \{0, 1\}^{m(x)+1}.$$

Write Avoid_f for the family-specific total search problem whose input is x and whose valid outputs are the strings outside $\text{Im}(D_x)$. Thus the hypothesis $\text{Avoid}_f \in \text{SearchNP}$ concerns only the circuit family generated by f , not arbitrary instances of **Avoid**.

Theorem 5.1. *Let V be a fixed verifier witnessing $\Pi \in \text{prAM}_{1,1/3}$, and suppose that the V -Targeted Hitting String problem reduces to **Avoid** via reduction (f, g) . If $\text{Avoid}_f \in \text{SearchNP}$, then $\Pi \in \text{prNP}$.*

Proof. Let M be a non-deterministic polynomial-time machine solving Avoid_f . We construct an NP verifier for Π . On input x , it guesses an accepting computation path of $M(x)$ and a string w . It checks the computation path, lets y be its output, computes

$$r := g(x, y),$$

and accepts iff $V(x, r, w) = 1$.

Suppose first that $x \in \Pi_{\text{YES}}$. The totality of the **SearchNP** algorithm gives an accepting path of $M(x)$ and hence an output y . Perfect completeness then gives a response w with $V(x, g(x, y), w) = 1$. Thus the NP verifier has an accepting witness.

Now suppose that $x \in \Pi_{\text{NO}}$. Every accepting path of $M(x)$ outputs some $y \notin \text{Im}(D_x)$. Correctness of the targeted hitting-string reduction therefore implies that $g(x, y)$ is rejected for every Merlin response. Hence no guessed pair can make the NP verifier accept. $\square$

Let f_{GNI} and f_{SVP} denote the two **Avoid** families obtained by composing the reductions in Theorems 4.4 and 4.6 with the reduction in Theorem 3.5.

Corollary 5.2 (Graph Non-Isomorphism). *If $\text{Avoid}_{f_{\text{GNI}}} \in \text{SearchNP}$, then Graph Non-Isomorphism is in $\text{NP} \cap \text{coNP}$.*

Corollary 5.3 (GapSVP). *Let $\gamma(n) := 400 \lceil \sqrt{n/\log n} \rceil$, as in Theorem 4.6. If $\text{Avoid}_{f_{\text{SVP}}} \in \text{SearchNP}$, then $\overline{\text{GapSVP}}_{\gamma} \in \text{prNP}$ and $\text{GapSVP}_{\gamma} \in \text{prNP} \cap \text{prcoNP}$.*

Acknowledgements

The author is grateful to Noah Stephens-Davidowitz and Roei Tell for useful comments and feedback about an early draft of this paper.

AI Usage

This paragraph is included as part of the initial collective adaptation of TCS to AI usage, and in particular following the recommendations of the report “AI and TCS: The Next Six Months” from September 2026.

This work is hatched through the author’s continued effort to relate **Avoid** and lattice problems. The starting point is a coupled proof idea that reduces the targeted hitting string for $\overline{\text{GapSVP}}$ to **Avoid**. The proof idea is then decoupled into proof ideas underlying Theorem 3.5 and Theorem 4.6. Definition 3.1 is then reverse-engineered into its current form, fitting the proof of Theorem 3.5. This stage is done without the usage of AI.

The author then asks GPT 6 Astra to expand the proof idea of [Theorem 4.6](#) into a full proof. In particular, the concrete discretization scheme was proposed by GPT 6 Astra. Next, believing that [Theorem 4.4](#) is true but not knowing how to prove it (without the knowledge of [Theorem 2.6](#)), the author asks GPT 6 Astra to prove it. GPT 6 Astra points out [Theorem 2.6](#) and proposes a proof which is essentially the proof of [Theorem 4.4](#) in the paper.

GPT 6 Astra was also used in the preparation of other parts of the manuscript, drafting preliminary versions of proof texts, performing a literature review and copy-editing. The author takes full responsibility for everything in the paper.

References

- [ABB⁺23] Divesh Aggarwal, Huck Bennett, Zvika Brakerski, Alexander Golovnev, Rajendra Kumar, Zeyong Li, Spencer Peters, Noah Stephens-Davidowitz, and Vinod Vaikuntanathan. Lattice problems beyond polynomial time. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1516–1526. ACM, 2023. Full version: arXiv:2211.11693.
- [AK01] Vikraman Arvind and Johannes Köbler. On pseudorandomness and resource-bounded measure. *Theoretical Computer Science*, 255(1–2):205–221, 2001.
- [AK02] Vikraman Arvind and Piyush P. Kurur. Graph isomorphism is in SPP. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 743–750. IEEE Computer Society, 2002.
- [AR05] Dorit Aharonov and Oded Regev. Lattice problems in $\text{NP} \cap \text{coNP}$. *Journal of the ACM*, 52(5):749–765, 2005.
- [Bab16] László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, pages 684–697. ACM, 2016.
- [BCT25] Marshall Ball, Lijie Chen, and Roei Tell. Towards free lunch derandomization from necessary assumptions (and OWFs). In *40th Computational Complexity Conference*, volume 339 of *Leibniz International Proceedings in Informatics*, pages 31:1–31:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025.
- [Ben23] Huck Bennett. The complexity of the shortest vector problem. *ACM SIGACT News*, 54(1):37–61, 2023.
- [BHZ87] Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25(2):127–132, 1987.
- [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.
- [CGL⁺26] Eldon Chung, Alexander Golovnev, Zeyong Li, Maciej Obremski, Sidhant Saraogi, and Noah Stephens-Davidowitz. The hardness of range avoidance for randomized algorithms implies minicrypt. In *46th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, Leibniz International Proceedings in Informatics. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2026. To appear.
- [CH22] Kuan Cheng and William M. Hoza. Hitting sets give two-sided derandomization of small space. *Theory of Computing*, 18(21):1–32, 2022.

- [CHLR23] Yeyuan Chen, Yizhi Huang, Jiayu Li, and Hanlin Ren. Range avoidance, remote point, and hard partial truth table via satisfying-pairs algorithms. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1058–1066. ACM, 2023.
- [CHLR26] Lijie Chen, Shuichi Hirahara, Zeyong Li, and Hanlin Ren. Symmetric exponential time requires near-maximum circuit size. *Journal of the ACM*, 73(1):4:1–4:21, 2026.
- [CL24] Yilei Chen and Jiayu Li. Hardness of range avoidance and remote point for restricted circuits via cryptography. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 620–629. ACM, 2024.
- [CLL25] Lijie Chen, Jiayu Li, and Jingxun Liang. Maximum circuit lower bounds for exponential-time arthur merlin. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, pages 1348–1358. ACM, 2025.
- [Cov73] Thomas M. Cover. Enumerative source encoding. *IEEE Transactions on Information Theory*, 19(1):73–77, 1973.
- [CRT22] Lijie Chen, Ron D. Rothblum, and Roei Tell. Unstructured hardness to average-case randomness. In *Proceedings of the 63rd Annual IEEE Symposium on Foundations of Computer Science*, pages 429–437. IEEE, 2022.
- [CT21] Lijie Chen and Roei Tell. Hardness vs randomness, revised: Uniform, non-black-box, and instance-wise. In *Proceedings of the 62nd Annual IEEE Symposium on Foundations of Computer Science*, pages 125–136. IEEE, 2021.
- [CT23a] Lijie Chen and Roei Tell. Guest column: New ways of studying the $BPP = P$ conjecture. *ACM SIGACT News*, 54(2):44–69, 2023.
- [CT23b] Lijie Chen and Roei Tell. When Arthur has neither random coins nor time to spare: Superfast derandomization of proof systems. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 60–69. ACM, 2023.
- [CTW23] Lijie Chen, Roei Tell, and Ryan Williams. Derandomization vs refutation: A unified framework for characterizing derandomization. In *Proceedings of the 64th Annual IEEE Symposium on Foundations of Computer Science*, pages 1008–1047. IEEE, 2023.
- [DPT24] Dean Doron, Edward Pyne, and Roei Tell. Opening up the distinguisher: A hardness to randomness approach for $BPL = L$ that uses properties of BPL. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 2039–2049. ACM, 2024.
- [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.
- [GG00] Oded Goldreich and Shafi Goldwasser. On the limits of nonapproximability of lattice problems. *Journal of Computer and System Sciences*, 60(3):540–563, 2000.
- [GGLS26] Karthik Gajulapalli, Surendra Ghentiyala, Zeyong Li, and Sidhant Saraogi. Downward self-reducibility in the total function polynomial hierarchy. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 5117–5150. SIAM, 2026.
- [GGNS23] Karthik Gajulapalli, Alexander Golovnev, Satyajeet Nagargoje, and Sidhant Saraogi. Range avoidance for constant depth circuits: Hardness and algorithms. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 275 of *Leibniz International Proceedings in Informatics*, pages 65:1–65:18, 2023.

- [GLS26] Surendra Ghentiyala, Zeyong Li, and Noah Stephens-Davidowitz. Range avoidance, arthur-merlin, and TFNP. In *Proceedings of the 58th Annual ACM Symposium on Theory of Computing*, pages 1925–1936. ACM, 2026.
- [GLW22] Venkatesan Guruswami, Xin Lyu, and Xiuhan Wang. Range avoidance for low-depth circuits and connections to pseudorandomness. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 245 of *Leibniz International Proceedings in Informatics*, pages 20:1–20:21, 2022.
- [GLY26] Venkatesan Guruswami, Xin Lyu, and Weiqiang Yuan. Cell-probe lower bounds via semi-random CSP refutation: Simplified and the odd-locality case. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 372–387. SIAM, 2026.
- [GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 174–187. IEEE, 1986.
- [Gol11] Oded Goldreich. In a world of $P = BPP$. In *Studies in Complexity and Cryptography*, volume 6650 of *Lecture Notes in Computer Science*, pages 191–232. Springer, 2011.
- [Gol20] Oded Goldreich. Two comments on targeted canonical derandomizers. In *Computational Complexity and Property Testing*, volume 12050 of *Lecture Notes in Computer Science*, pages 24–35. Springer, 2020.
- [GS86] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 59–68. ACM, 1986.
- [GS20] Martin Grohe and Pascal Schweitzer. The graph isomorphism problem. *Communications of the ACM*, 63(11):128–134, 2020.
- [GST03] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. Uniform hardness versus randomness tradeoffs for arthur-merlin games. *Computational Complexity*, 12(3–4):85–130, 2003.
- [GvHKW20] Yunus Can Gültekin, Wim J. van Houtum, Arie Koppelaar, and Frans M. J. Willems. Enumerative sphere shaping for wireless communications with short packets. *IEEE Transactions on Wireless Communications*, 19(2):1098–1112, 2020.
- [HLZ26] Shengtang Huang, Xin Li, and Yan Zhong. Range avoidance and remote point: New algorithms and hardness. In *17th Innovations in Theoretical Computer Science Conference*, volume 362 of *Leibniz International Proceedings in Informatics*, pages 79:1–79:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2026.
- [Ila25] Rahul Ilango. The oracle derandomization hypothesis is false (and more) assuming no natural proofs. Technical Report TR25-190, Electronic Colloquium on Computational Complexity, 2025.
- [ILW23] Rahul Ilango, Jiayu Li, and R. Ryan Williams. Indistinguishability obfuscation, range avoidance, and bounded arithmetic. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1076–1089. ACM, 2023.
- [IW97] Russell Impagliazzo and Avi Wigderson. $p = bpp$ if e requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 220–229. ACM, 1997.
- [IW01] Russell Impagliazzo and Avi Wigderson. Randomness vs time: Derandomization under a uniform assumption. *Journal of Computer and System Sciences*, 63(4):672–688, 2001.

- [Jeř04] Emil Jeřábek. Dual weak pigeonhole principle, Boolean complexity, and derandomization. *Annals of Pure and Applied Logic*, 129(1–3):1–37, 2004.
- [KKMP21] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos Papadimitriou. Total functions in the polynomial hierarchy. In *12th Innovations in Theoretical Computer Science Conference*, volume 185 of *Leibniz International Proceedings in Informatics*, pages 44:1–44:18, 2021.
- [Kor21] Oliver Korten. The hardest explicit construction. In *Proceedings of the 62nd Annual IEEE Symposium on Foundations of Computer Science*, pages 433–444. IEEE, 2021.
- [Kor22] Oliver Korten. Derandomization from time-space tradeoffs. In *37th Computational Complexity Conference*, volume 234 of *Leibniz International Proceedings in Informatics*, pages 37:1–37:26, 2022.
- [Kor25] Oliver Korten. Range avoidance and the complexity of explicit constructions. *Bulletin of the EATCS*, 145, 2025.
- [KPI25] Oliver Korten, Toniann Pitassi, and Russell Impagliazzo. Stronger cell probe lower bounds via local PRGs. In *Proceedings of the 66th Annual IEEE Symposium on Foundations of Computer Science*, pages 2480–2504. IEEE, 2025.
- [KS25a] Oliver Korten and Rahul Santhanam. How to construct random strings. In *40th Computational Complexity Conference*, volume 339 of *Leibniz International Proceedings in Informatics*, pages 35:1–35:32, 2025.
- [KS25b] Neha Kuntewar and Jayalal Sarma. Avoiding range via Turan-type bounds. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 353 of *Leibniz International Proceedings in Informatics*, pages 62:1–62:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025.
- [Kv02] Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, 2002.
- [LP22] Yanyi Liu and Rafael Pass. Characterizing derandomization through hardness of levin–kolmogorov complexity. In *37th Computational Complexity Conference*, volume 234 of *Leibniz International Proceedings in Informatics*, pages 35:1–35:17, 2022.
- [LP23] Yanyi Liu and Rafael Pass. Leakage-resilient hardness vs randomness. In *38th Computational Complexity Conference*, volume 264 of *Leibniz International Proceedings in Informatics*, pages 32:1–32:20, 2023.
- [LRZ26] Xin Li, Hanlin Ren, and Yan Zhong. Many proof complexity generators inside one demi-bits generator. In *Proceedings of the 67th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 2026. To appear.
- [LT26] Zeyong Li and Roei Tell. Trading time, space, and alternations: General plasticity for algorithms from hardness of range avoidance. Technical Report TR26-233, Electronic Colloquium on Computational Complexity, 2026.
- [Lu01] Chi-Jen Lu. Derandomizing arthur–merlin games under uniform assumptions. *Computational Complexity*, 10(3):247–259, 2001.
- [MV05] Peter Bro Miltersen and N. V. Vinodchandran. Derandomizing arthur–merlin games using hitting sets. *Computational Complexity*, 14(3):256–279, 2005.

- [NW94] Noam Nisan and Avi Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.
- [PP23] Aaron (Louie) Putterman and Edward Pyne. Near-optimal derandomization of medium-width branching programs. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 23–34. ACM, 2023.
- [PRZ23] Edward Pyne, Ran Raz, and Wei Zhan. Certified hardness vs. randomness for log-space. In *Proceedings of the 64th Annual IEEE Symposium on Foundations of Computer Science*, pages 989–1007. IEEE Computer Society, 2023.
- [Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):34:1–34:40, 2009.
- [RSW22] Hanlin Ren, Rahul Santhanam, and Zhikun Wang. On the range avoidance problem for circuits. In *Proceedings of the 63rd Annual IEEE Symposium on Foundations of Computer Science*, pages 640–650. IEEE, 2022.
- [RW26] Hanlin Ren and Ryan Williams. Near-maximum circuit lower bounds for exponential time with merlin-arthur queries. In *Proceedings of the 67th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 2026. To appear.
- [RWZ26] Hanlin Ren, Yichuan Wang, and Yan Zhong. Hardness of range avoidance and proof complexity generators from demi-bits. In *17th Innovations in Theoretical Computer Science Conference*, volume 362 of *Leibniz International Proceedings in Informatics*, pages 111:1–111:25, 2026.
- [STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.
- [SU05] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *Journal of the ACM*, 52(2):172–216, 2005.
- [SU09] Ronen Shaltiel and Christopher Umans. Low-end uniform hardness versus randomness tradeoffs for AM. *SIAM Journal on Computing*, 39(3):1006–1037, 2009.
- [SZ99] Michael Saks and Shiyu Zhou. $BP_HSPACE(S) \subseteq DSPACE(S^{3/2})$. *Journal of Computer and System Sciences*, 58(2):376–403, 1999.
- [TV07] Luca Trevisan and Salil Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007.
- [Uma03] Christopher Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67(2):419–440, 2003.
- [vMS23] Dieter van Melkebeek and Nicollas Mocelin Sdroievski. Leakage resilience, targeted pseudorandom generators, and mild derandomization of arthur–merlin protocols. In *43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 284 of *Leibniz International Proceedings in Informatics*, pages 29:1–29:22, 2023.
- [vMS25] Dieter van Melkebeek and Nicollas Mocelin Sdroievski. Instance-wise hardness and refutation versus derandomization for arthur–merlin protocols. *Computational Complexity*, 34(15):1–86, 2025.
- [Vol20] Ilya Volkovich. The untold story of SBP. In *Computer Science – Theory and Applications*, volume 12159 of *Lecture Notes in Computer Science*, pages 393–405. Springer, 2020.
- [ZF87] Stathis Zachos and Martin Fürer. Probabilistic quantifiers vs. distrustful adversaries. In *Foundations of Software Technology and Theoretical Computer Science*, volume 287 of *Lecture Notes in Computer Science*, pages 443–455. Springer, 1987.