Efficient Construction of Rigid Matrices Using an NP Oracle

Josh Alman
MIT
jalman@mit.edu

Lijie Chen
MIT
lijieche@mit.edu

October 1, 2019

Abstract

For a matrix H over a field $\mathbb{F}$, its rank-r rigidity, denoted $\mathscr{R}_H(r)$, is the minimum Hamming distance from H to a matrix of rank at most r over $\mathbb{F}$. A central open challenge in complexity theory is to give explicit constructions of rigid matrices for a variety of parameter settings. In this work, building on Williams' seminal connection between circuit-analysis algorithms and lower bounds [Williams, J. ACM 2014], we give a construction of rigid matrices in P^{NP}. Letting $q = p^r$ be a prime power, we show:

- There is an absolute constant $\delta > 0$ such that, for all constants $\varepsilon > 0$, there is a P^{NP} machine M such that, for infinitely many N's, $M(1^N)$ outputs a matrix $H_N \in \{0,1\}^{N \times N}$ with $\mathscr{R}_{H_N}(2^{(\log N)^{1/4-\varepsilon}}) \ge \delta \cdot N^2$ over $\mathbb{F}_q$.

Using known connections between matrix rigidity and other topics in complexity theory, we derive several consequences of our constructions, including:

- There is a function $f \in \mathsf{TIME}\left[2^{(\log n)^{\omega(1)}}\right]^{\mathsf{NP}}$ such that $f \notin \mathsf{PH}^{\mathsf{cc}}$. Previously, it was open whether $\mathsf{E}^{\mathsf{NP}} \subset \mathsf{PH}^{\mathsf{cc}}$.
- For all $\varepsilon > 0$, there is a P^{NP} machine M such that, for infinitely many N's, $M(1^N)$ outputs an $N \times N$ matrix $H_N \in \{0,1\}^{N \times N}$ whose linear transformation requires depth-2 $\mathbb{F}_q$-linear circuits of size $\Omega(N \cdot 2^{(\log N)^{1/4-\varepsilon}})$. The previous best lower bound for an explicit family of $N \times N$ matrices over $\mathbb{F}_q$ was only $\Omega(N \log^2 N/(\log\log N)^2)$, for asymptotically good error-correcting codes.

1 Introduction

Let $\mathbb{F}$ be any field. The rank-r rigidity of a matrix $H \in \mathbb{F}^{N\times N}$, denoted $\mathscr{R}_H(r)$, is the minimum Hamming distance between H and any matrix of rank at most r. Ever since Leslie Valiant introduced the notion of matrix rigidity [Val77], it has been a major challenge to construct interesting rigid matrices.

Valiant showed that if $\{H_N\}_{N\in\mathbb{N}}$ is a family of matrices where H_N is an $N \times N$ matrix with rigidity $\mathscr{R}_{H_N}(N/\log\log N) \geq N^{1+\varepsilon}$ for any $\varepsilon > 0$, then the linear transformation defined by H_N cannot be computed by circuits of size $O(N)$ and depth $O(\log N)$. It remains an open problem to prove that any explicit family of matrices does not have such circuits. Since Valiant's result, connections have been drawn between rigid matrices (for many different rank parameters) and lower bounds in a number of areas including in arithmetic circuit complexity, Boolean circuit complexity, communication complexity, and error-correcting codes; see [Lok09] for a survey of these connections.

Valiant also showed that there exists an $N \times N$ matrix R_N over a finite field $\mathbb{F}_q$ with $\mathscr{R}_{R_N}(r) \geq \Omega\left(N^2\right)$ for all $r = o(N)$, and a random such matrix R_N has $\mathscr{R}_{R_N}(r) \geq \Omega\left(\frac{(N-r)^2}{\log N}\right)$ for all r with high probability[1]. However, this is not particularly exciting in the context of proving circuit lower bounds, since it is not hard to see that a *random* linear transformation cannot be computed by small circuits with high probability. It is thus most interesting to search for *explicit* rigid matrices: We say $\{H_N\}_{N\in\mathbb{N}}$ is explicit if there is a deterministic polynomial-time algorithm which, on input 1^N, outputs the $N \times N$ matrix H_N.

Despite decades of work and many known applications of rigid matrices, there has not been much success in actually constructing rigid matrices for almost any interesting rank parameter. There are essentially only three known deterministic constructions:

- For all ranks r, there is a family of $N \times N$ matrices M_N constructible in P with $\mathscr{R}_{M_N}(r) \geq \Omega\left(\frac{N^2}{r}\log(N/r)\right)$ [Fri93, SSS97]. This is proved via a combinatorial argument ("untouched minor argument"), and it is known that this type of approach cannot be further improved [Lok00].
- By combining a brute-force search for very rigid $r \times r$ matrices with a padding argument (see Lemma 2.7 below), we can construct, for any rank r, an $N \times N$ matrix L_N in $\mathsf{TIME}[\exp(r^2)]$ with $\mathscr{R}_{L_N}(r) \geq \Omega(N^2)$.
- Goldreich and Tal [GT16] show that random $N \times N$ Toeplitz matrices T_N over a finite field $\mathbb{F}_q$ have $\mathscr{R}_{T_N}(r) \geq \Omega\left(\frac{N^3}{r^2\log N}\right)$ for all $r \geq \sqrt{N}$ with high probability. Their proof is primarily combinatorial and linear algebraic. Using their construction, such rigid matrices can be constructed in E.[2]

Over large fields $\mathbb{F}$, there are also approaches to constructing matrices which are rigid by virtue of having very large entries. For instance, an 'algebraic dimension' approach [SS96] can be used to construct rigid matrices over $\mathbb{C}$ with algebraically independent entries [Lok00, Lok06]. In this paper, we focus on matrices over constant-size finite fields $\mathbb{F}_{p^r}$ where such techniques cannot work.

1.1 Our Results

In this paper, we give a new construction of rigid matrices. Unlike previous constructions, which primarily use combinatorial and algebraic techniques, our construction primarily uses *complexity-theoretic* ideas. Our

[1] By comparison, it is not hard to see that $\mathscr{R}_{R_N}(r) \leq (N-r)^2$ for all r and all $N \times N$ matrices R_N.

[2] [GT16] gives a test computable in time $2^{O(N)}$, such that: (1) a random $N \times N$ Toeplitz matrix passes the test with high probability, and (2) all matrices passing the test are rigid with the stated parameters. Therefore, one can construct such an explicit matrix in E by enumerating all possible random bits defining a Toeplitz matrix, and outputting the first matrix passing the test.

matrices are rigid for rank parameters which are smaller than what is needed for Valiant's program, but are still high enough that we can prove new lower bounds in communication complexity, Boolean circuit complexity, and arithmetic circuit complexity.

1.1.1 Construction of Rigid Matrices in P^{NP}

Our main result is a construction of a rigid matrix in P^{NP}:

Theorem 1.1 (An Infinitely Often Rigid Matrix Construction in P^{NP})**.** *There is an absolute constant $\delta > 0$ such for all prime powers $q = p^r$ and all constants $\varepsilon > 0$:*

- *There is a P^{NP} machine M such that, for infinitely many N, on input 1^N, M outputs an $N \times N$ matrix $H_N \in \{0,1\}^{N\times N}$ such that $\mathscr{R}_{H_N}(2^{(\log N)^{1/4-\varepsilon}}) \geq \delta \cdot N^2$ over $\mathbb{F}_q$.*

By comparison, applying previously known techniques to construct rigid $N \times N$ matrices M_N for this rank $r = 2^{(\log N)^{1/4-\varepsilon}}$, one either obtains:

- M_N constructible in P with only $\mathscr{R}_{M_N}(r) \geq \Omega\left(\frac{N^2}{2^{(\log N)^{1/4-\varepsilon}}}\right)$, or
- M_N only constructible in $\mathsf{TIME}[\exp(\exp((\log N)^{1/4-\varepsilon}))]$ with $\mathscr{R}_{M_N}(r) \geq \Omega\left(N^2\right)$. Note that the time bound here is larger than any quasi-polynomial in N, which can be written as $\exp(\exp(\log\log N))$.

Our construction in Theorem 1.1 is in P^{NP}, and achieves $\mathscr{R}_{M_N}(r) \geq \Omega\left(N^2\right)$.

1.1.2 Either $\mathsf{NQP} \not\subset \mathsf{P}_{/\,\mathrm{poly}}$ or a Better Construction of Rigid Matrices

It is natural to ask whether one can improve the constant $1/4 - \varepsilon$ in the rank in Theorem 1.1. We show an interesting "win-win" theorem: either the constant can be improved from $1/4-\varepsilon$ to $1-\varepsilon$, or $\mathsf{NQP} \not\subset \mathsf{P}_{/\,\mathrm{poly}}$ follows.

Theorem 1.2 (Either a Better Construction in P^{NP} or $\mathsf{NQP} \not\subset \mathsf{P}_{/\,\mathrm{poly}}$)**.** *There is an absolute constant $\delta > 0$ such that for all prime powers $q = p^r$ and all constants $\varepsilon > 0$, at least one of the following holds:*

- *$\mathsf{NQP} \not\subset \mathsf{P}_{/\,\mathrm{poly}}$.*
- *There is a P^{NP} machine M such that, for infinitely many N, on input 1^N, M outputs an $N \times N$ matrix $H_N \in \{0,1\}^{N\times N}$ such that $\mathscr{R}_{H_N}(2^{(\log N)^{1-\varepsilon}}) \geq \delta \cdot N^2$ over $\mathbb{F}_q$.*

Theorem 1.2 is interesting from the perspective of proving circuit lower bounds. Recall that a main motivation for constructing rigid matrices is to construct an explicit function which cannot be computed by $O(n)$-size $O(\log n)$-depth circuits [Val77]. If we aim to show that NE (or E^{NP}) does not admit such circuits (which is still open), then we can safely assume $\mathsf{NEXP} \subset \mathsf{P}_{/\,\mathrm{poly}}$ before constructing the required rigid matrices. Therefore, if one could further improve the construction in the second bullet of the above Theorem 1.2 to match the rigidity required by [Val77] (which would require $N \times N$ matrices H_N with $\mathscr{R}_{H_N}(N/\log\log N) \geq N^{1+\varepsilon}$ for any $\varepsilon > 0$, i.e. an improved rank parameter in exchange for a worsened rigidity parameter), it would imply that E^{NP} does not have $O(n)$-size $O(\log n)$-depth circuits.

1.1.3 Applications

Application: $\mathsf{PH}^{\mathsf{cc}}$ Lower Bound for $\mathsf{NTIME}[2^{(\log n)^{\omega(1)}}]^{\mathsf{NP}}$. A longstanding open problem in communication complexity is to prove a $\mathsf{PH}^{\mathsf{cc}}$ (the communication complexity analogue of the polynomial hierarchy) lower bound for an explicit function [BFS86] (see [GPW18] for a recent reference). In fact, even for the much weaker subclass $\mathsf{AM}^{\mathsf{cc}}$, it is a notoriously open question to prove an $\omega(\log n)$ lower bound for any explicit function [GPW16, CW19a]. Prior to this paper, it was even open whether $\mathsf{E}^{\mathsf{NP}} \subset \mathsf{AM}^{\mathsf{cc}}$, i.e., does every function in E^{NP} have an efficient AM communication protocol?

Razborov showed a rigidity upper bound for the truth-table matrix of any function in $\mathsf{PH}^{\mathsf{cc}}$:

Lemma 1.3 ([Raz89], see also [Wun12]). *Letting f be a function in $\mathsf{PH}^{\mathsf{cc}}$, the $2^n \times 2^n$ communication matrix M_f of f has $\mathscr{R}_{M_f}(2^{(\log n/\varepsilon)^c}) \leq \varepsilon \cdot 4^n$, where $\varepsilon > 0$ is arbitrary and $c > 0$ is a constant depending only on f, but not n.*

Using this, our construction of rigid matrices in Theorem 1.1 immediately shows that $\mathsf{E}^{\mathsf{NP}} \not\subset \mathsf{PH}^{\mathsf{cc}}$, giving the first non-trivial lower bound against $\mathsf{PH}^{\mathsf{cc}}$. In fact, our rigidity bound is for a much higher rank than is necessary for applying Lemma 1.3 (setting $n = \log N$ in Theorem 1.1, we give a $2^n \times 2^n$ matrix M with $\mathscr{R}_M(2^{n^{1/4-\varepsilon}}) \geq \delta \cdot 4^n$ for infinitely many n). By a simple modification of our construction, we prove an even stronger lower bound:

Theorem 1.4. *For all functions $\alpha(n) = \omega(1)$ such that $n^{\alpha(n)}$ is time-constructible, there is a function $f \in \mathsf{TIME}[2^{(\log n)^{\alpha(n)}}]^{\mathsf{NP}}$ which is not in $\mathsf{PH}^{\mathsf{cc}}$.*

Of the three previously-known deterministic constructions of rigid matrices mentioned in the introduction, only the second constructs rigid enough matrices to apply Lemma 1.3. However, it only yields a $2^n \times 2^n$ matrix M with $\mathscr{R}_M(2^{(\log n)^{\omega(1)}}) \geq \Omega(4^n)$ in $\mathsf{TIME}[\exp(\exp((\log n)^{\omega(1)}))]$. We obtain an *exponential* time savings using an NP oracle.

Application: Depth-2 Arithmetic Circuit Lower Bounds Although the rank parameters in our rigidity lower bounds from Theorems 1.1 are not high enough to give log-depth arithmetic circuit lower bounds via Valiant's approach, the *rigidity* parameters are high enough that we can prove lower bounds against constant-depth arithmetic circuits. We consider a variant on rigidity which is useful for studying depth-2 arithmetic circuits:

Definition 1.5. For a field $\mathbb{F}$ and a matrix $A \in \mathbb{F}^{N \times N}$, let

$$w_2(A) := \min\{\mathrm{nnz}(B) + \mathrm{nnz}(C) \mid A = BC\},$$

where the min is over all pairs B, C of matrices of any dimensions over $\mathbb{F}$ whose product is A, and $\mathrm{nnz}(X)$ denotes the number of nonzero entries in the matrix X.

It is not hard to see that $w_2(A)$ equals, up to an additive[3] n, the minimum size (number of wires) of a depth-2 linear circuit over $\mathbb{F}$ which computes A, i.e. a depth-2 circuit which takes as input the N entries of a vector $x \in \mathbb{F}^N$ and outputs the N entries of the vector Ax, and whose gates compute $\mathbb{F}$-linear combinations of their inputs.

[3] $w_2(A)$ equals the minimum size of a depth-2 linear circuit for A when wires are not allowed to go directly from inputs to outputs. We can convert a circuit where wires do go from inputs to outputs to one where they do not, by adding in n middle-level gates which take the values of the n inputs. Hence, the minimum size of a depth-2 linear circuit for A differs from $w_2(A)$ by a negligible additive $\leq n$.

Every matrix $M \in \{0,1\}^{N \times N}$ has $w_2(M) \leq O(N^2/\log N)$ over any field, and similar to the situation for rigidity, for any fixed prime power $q = p^r$, a random matrix $A \in \mathbb{F}_q^{N \times N}$ has $w_2(A) \geq \Omega(N^2/\log N)$ with high probability [Lup56]. However, the best known lower bounds on w_2 for explicit families of $N \times N$ matrices over constant-size finite fields are only:

- $\Omega(N \log N)$ for Boolean Hadamard matrices [AKW90], and
- $\Omega(N \log^2 N/(\log\log N)^2)$ for asymptotically good error-correcting codes [GHK$^+$12].

A lower bound of $\Omega(N \log^2 N/\log\log N)$ is also known for matrices based on super-concentrator graphs [RTS00], but these constructions require larger fields; see [Lok09, Section 2.3] for further discussion.

Connections between rigidity lower bounds and w_2 lower bounds for a number of different parameter settings are known [Pud94]. We apply our rigidity lower bounds using a similar connection in the high rigidity setting to show higher w_2 lower bounds for matrices constructible in P^{NP}:

Theorem 1.6. *For all prime powers $q = p^r$ and constants $\varepsilon > 0$, it holds:*

- *There is a P^{NP} machine M such that, for infinitely many N, on input 1^N, M outputs an $N \times N$ matrix $H_N \in \{0,1\}^{N \times N}$ such that $w_2(H_N) \geq \Omega(N \cdot 2^{(\log N)^{1/4-\varepsilon}})$ over $\mathbb{F}_q$.*

Application: $\mathsf{AC}^0[p] \circ \mathsf{LTF} \circ \mathsf{AC}^0[p] \circ \mathsf{LTF}$ Circuit Lower Bounds. We next give an application of our construction to Boolean circuit complexity. Building off of a known connection between rigid matrices and threshold circuits [Lok01, AW17] we give a lower bound against a powerful class of circuits with threshold gates:

Theorem 1.7. *For every $\delta > 0$ and prime p, there is an $a > 0$ such that the class E^{NP} does not have non-uniform $\mathsf{AC}^0[p] \circ \mathsf{LTF} \circ \mathsf{AC}^0[p] \circ \mathsf{LTF}$ circuits of depth $o(\log n/\log\log n)$ where the bottom LTF layer has $2^{O(n^a)}$ gates, the rest of the circuit has polynomial size, and the middle layer LTF gates have fan-in $O(n^{1/2-\delta})$.*

We briefly compare with some prior lower bounds for threshold circuits:

- It is known [ACW16] that E^{NP} does not have non-uniform $\mathsf{ACC}^0 \circ \mathsf{LTF} \circ \mathsf{LTF}$ circuits where the bottom LTF layer has $n^{2-\varepsilon}$ gates and the remaining $\mathsf{ACC}^0 \circ \mathsf{LTF}$ subcircuit has $2^{n^{o(1)}}$ size. Tamaki [Tam16] also showed similar results for depth-2 circuits with symmetric and threshold gates. Our new lower bound is incomparable to these: we allow for many more LTF gates in the bottom layer, and unbounded depth, but the prior result allowed for larger size above the bottom layer, as well as ACC^0 circuitry rather than just $\mathsf{AC}^0[p]$ circuitry.
- Kane and Williams [KW16] previously showed there is a function in P which requires $\mathsf{MAJ} \circ \mathsf{LTF} \circ \mathsf{LTF}$ circuits of size $\Omega(n^{3/2}/\log^3 n)$. Our lower bound is for much larger circuits than this, but without a MAJ gate on top, and for a function in E^{NP} instead of P.

In fact, in order to prove Theorem 1.7, we actually prove a more general lower bound about polynomials over linear threshold functions which must be correct on all but a constant fraction of inputs. For a field $\mathbb{F}$ and positive integers d and m, let $\mathsf{POLY}[d, m, \mathbb{F}]$ denote the set of functions $\{0,1\}^m \to \mathbb{F}$ (with fan-in m) which can be computed[4] by degree d polynomials over $\mathbb{F}$.

[4] Such a polynomial may output any values over $\mathbb{F}$. However, in order to correctly compute a Boolean function $f : \{0,1\}^m \to \{0,1\}$ on input $x \in \{0,1\}^m$, the polynomial must correctly output 0 or 1 on x. Over a constant-sized finite field, one may map all nonzero values in $\mathbb{F}$ to 1 with only a constant factor increase in the degree.

Theorem 1.8. *There is a universal $\delta > 0$ such that for all prime powers $q = p^r$ and all constants $\varepsilon > 0$, there is an $a > 0$ such that:*

- *There is an $f \in \mathsf{E}^{\mathsf{NP}}$ such that every (non-uniform) function in $\mathsf{POLY}[O(n^{1/4-\varepsilon}), 2^{O(n^a)}, \mathbb{F}_q] \circ \mathsf{LTF}$ disagrees with f on a δ-fraction of inputs for infinitely many n. In other words, E^{NP} cannot be approximated by degree-$O(n^{1/4-\varepsilon})$ $\mathbb{F}_q$ polynomials over any $2^{O(n^a)}$ linear threshold functions of the inputs.*

By comparison, Williams [Wil18b] recently showed that, for every unbounded $\alpha(n)$ such that $n^{\alpha(n)}$ is time constructible, there is a function in $\mathsf{NTIME}[n^{\alpha(n)}]$ which does not have a (non-uniform) representation in $\mathsf{POLY}[1, n^{O(1)}, \mathbb{R}] \circ \mathsf{LTF}$.

1.2 Proof Overview

In this subsection we give an overview of our construction of rigid matrices in P^{NP}. For simplicity, we only consider the field $\mathbb{F}_2$ in this overview.

1.2.1 Either $\mathsf{NE} \not\subset \mathsf{P}_{/\,\mathrm{poly}}$ or a Construction of Rigid Matrices

We begin with a proof overview of Theorem 1.2. Note that Theorem 1.2 is equivalent to saying that there is a rigid matrix construction in P^{NP} under the assumption $\mathsf{NQP} \subset \mathsf{P}_{/\,\mathrm{poly}}$. Here we outline a conditional construction under the stronger assumption $\mathsf{NE} \subset \mathsf{P}_{/\,\mathrm{poly}}$ for simplicity. We will then show how to get rid of the assumption using an additional bootstrapping argument.

Theorem 1.9 (Either a Better Construction in P^{NP} or $\mathsf{NE} \not\subset \mathsf{P}_{/\,\mathrm{poly}}$). *There is an absolute constant $\delta > 0$ such that for all constants $\varepsilon > 0$, at least one of the following holds:*

- *$\mathsf{NE} \not\subset \mathsf{P}_{/\,\mathrm{poly}}$.*
- *There is a P^{NP} machine M such that, for infinitely many N, on input 1^N, M outputs an $N \times N$ matrix $H_N \in \{0,1\}^{N \times N}$ such that $\mathscr{R}_{H_N}(2^{(\log N)^{1-\varepsilon}}) \ge \delta \cdot N^2$ over $\mathbb{F}_q$.*

Low-Rank Matrices as a Circuit Class, and Corresponding Circuit Analysis Algorithms. We begin with the observation that we can view low-rank matrices over $\mathbb{F}_2$ as a special type of 'circuit' defined by a pair of matrices. That is, supposing $M \in \mathbb{F}_2^{N \times N}$ is a matrix with rank r (think of $r \ll N$), then there are matrices $A \in \mathbb{F}_2^{N \times r}$ and $B \in \mathbb{F}_2^{r \times N}$ such that $M = A \cdot B$. Assuming N is a power of 2 for simplicity, M can be interpreted as (the truth-table of) a Boolean function $f : \{0,1\}^{2\log N} \to \{0,1\}$, which has a special type of circuit of size $O(N \cdot r)$ defined by A and B.

In this way, our task of constructing rigid matrices can equivalently be viewed as the task of proving a certain average-case lower bound against this special class of circuits. This is how Williams' algorithmic approach [Wil13, Wil14b], which exploits circuit analysis algorithms to prove such lower bounds, comes into play. When given the matrices A, B, the corresponding circuit analysis questions are:

1. *Satisfiability (SAT)*, which asks whether $A \cdot B$ is the all zero matrix,
2. *Derandomization* (CAPP), which asks for an estimate of the probability that a random entry of $A \cdot B$ is 1, and
3. *Counting* ($\#\mathsf{SAT}$), which asks for the exact number of ones in $A \cdot B$.

In fact, we observe that given the pair (A, B), we can solve the hardest of these three problems, $\#\mathsf{SAT}$, in better-than-2^n time (note $n = 2 \log N$). More formally, let a_i denote the i-th row of A, and let b_j denote the j-th column of B. The goal of $\#\mathsf{SAT}$ is to count the number of pairs such that $\langle a_i, b_j \rangle = 0$ (the number of ones is N^2 minus the number of zeros). This is exactly an instance of *Counting* OV *over* $\mathbb{F}_2$ ($\mathbb{F}_2$-#OV), with N vectors of r dimensions; compared to the usual OV problem, our inner product here is over $\mathbb{F}_2$ instead of $\mathbb{Z}$. An algorithm by Chan and Williams [CW16] solves this problem in *deterministic* $N^{2-\Omega(1/\log(r/\log N))}$ time, for all $r \leq N^{o(1)}$ (see Subsection 2.4 and Appendix A for the details). This algorithm will play a crucial part in our construction.

Williams' Algorithmic Approach to Circuit Lower Bounds, and a First Attempt. In a seminal work, Williams [Wil13] demonstrated an algorithmic approach to proving circuit lower bounds. At a high level, the approach works as follows: Assuming a circuit lower bound is false, one combines the resulting small circuits with other algorithmic ideas to get a better-than-2^n non-deterministic algorithm for $\mathsf{NTIME}[2^n]$, therefore contradicting the non-deterministic time hierarchy theorem [Zák83].

A first attempt at using this approach in our situation proceeds as follows. Let L be a unary language in $\mathsf{NTIME}[2^n] \setminus \mathsf{NTIME}[2^n/n]$ [Zák83]. Fix an efficient PCP verifier V for L (such as [BV14]). That is, for a function $\ell := \ell(n) = n + O(\log n)$, $V(1^n)$ takes ℓ random inputs, runs in $\mathrm{poly}(n)$ time, and is given access to an oracle $O : \{0,1\}^\ell \to \{0,1\}$ (O corresponds to the length-2^ℓ proof for V, but we will interpret it as an ℓ-bit Boolean function to help with intuition later on), and satisfies the following conditions:

1. (PCP Completeness) if $1^n \in L$, then there exists an oracle O such that $V(1^n)^O$ always accepts;
2. (PCP Soundness) if $1^n \notin L$, then for all possible oracles O, the probability $V(1^n)^O$ accepts is $\leq 1/3$.

Intuitively, we are going to show that the truth table of the oracle O which makes V always accept (in the PCP Completeness case) has to be a rigid matrix. More precisely, letting $N = 2^{\ell/2}$, we can fix a P^{NP} machine M_{rigid} such that, on input 1^N, $M_{\mathsf{rigid}}(1^N)$ outputs the lexicographically first oracle O_n which makes $V(1^n)$ always accept. M_{rigid} runs in P^{NP} (on input 1^N, which has length $2^{\Omega(n)}$), since it can guess the oracle outputs bit by bit, using its NP oracle to verify its guesses. The output of $M_{\mathsf{rigid}}(1^N)$, and hence O_n itself, can be viewed as a matrix from $\{0,1\}^{N \times N}$ which we want to show is rigid.

Assume toward a contradiction that $\mathscr{R}_{M_{\mathsf{rigid}}(1^N)}(r) \leq \delta \cdot N^2$ for a small constant δ (one can think of $r := 2^{(\log N)^{1-\varepsilon}}$ for a small constant $\varepsilon > 0$) for all N. It follows that O_n can be $(1-\delta)$-approximated by a matrix of rank at most r. We can thus attempt to solve L as follows:

- Given an input 1^n, we guess matrices $A \in \mathbb{F}_2^{N \times r}$ and $B \in \mathbb{F}_2^{r \times N}$ in $\widetilde{O}(r \cdot 2^{n/2})$ time, with the hope that $M := A \cdot B$ approximates O_n.
- We estimate
$$p_{\mathsf{acc}}(M) = \Pr_{\tau \in \{0,1\}^\ell}[V(1^n)^M(\tau) = 1],$$
and accept only if $p_{\mathsf{acc}}(M) \geq 2/3$.

Following Williams' approach, the hope is that we can estimate $p_{\mathsf{acc}}(M)$ in $2^n/n$ time (i.e. faster than iterating over all choices of the randomness τ) by taking advantage of the given low-rank approximation of M, combined with the $\#\mathsf{SAT}$ algorithm for low rank matrices. If this were possible, it would put L in $\mathsf{NTIME}[2^n/n]$, and contradict the non-deterministic time hierarchy theorem, completing our proof.

Two Issues with the First Attempt. Unfortunately, there are two main issues with this attempt. The first issue is that $V(1^n)^M(\cdot)$ can no longer be written as a low-rank matrix, even if M can. Ideally we would like $V(1^n)^M(\cdot)$ to be a low-rank matrix so that our $\#\mathsf{SAT}$ algorithm applies to estimate $p_{\mathsf{acc}}(M)$; without this condition, it's unclear how the low-rank matrix M is helpful. From [BV14], one can actually take $V(1^n)$ to be a *3-CNF*, but this is still not enough, since a 3-CNF of low-rank matrices is not necessarily a low-rank matrix.

The second issue is more subtle. If we can estimate $p_{\mathsf{acc}}(M)$ with a high enough accuracy, clearly we will always reject when $1^n \notin L$, by the soundness of the PCP. But, in order to accept when $1^n \in L$, even if we have guessed an M which $(1-\delta)$-approximates O_n, it still could be the case that $p_{\mathsf{acc}}(M)$ is small. For instance, what if $V(1^n)$ always queries positions on which M and O_n differ?

We will ultimately resolve the second issue by making the verifier *smooth* (meaning each query is uniformly distributed), which we will explain later. To resolve the first issue, we use a recent idea from Chen and Williams [CW19b], together with easy-witness lemmas [IKW02, MW18].

The Easy Witness Lemma. Assuming $\mathsf{NE} \subset \mathsf{P}_{/\,\mathrm{poly}}$, by [IKW02], we know that all NE verifiers have polynomial-size witness circuits, including the verifier $V(1^n)$ discussed above. In other words, when $1^n \in L$, before we were only able to assume there is an oracle $O : \{0,1\}^\ell \to \{0,1\}$ such that $V(1^n)^O$ always accepts, but now we can further assume that there is such an oracle which is computed by a circuit $C : \{0,1\}^\ell \to \{0,1\}$ of size n^k for a constant k. Let us set C_{best} to be the lexicographically first circuit having this property. Now we can modify our algorithm from the first attempt by guessing C, and trying to estimate $p_{\mathsf{acc}}(C)$ instead. Notice that with this modification, there are no longer any low-rank matrices involved in our current approach. We will instead use low-rank approximations of the proof for a different PCP, which we describe next.

Smooth PCP of Proximity (PCPP). We are now going to make use of a very recent construction of a smooth PCPP [Par19]. Using PCPPs in conjunction with Williams' algorithmic approach to circuit lower bounds in this way was a key idea from [CW19b]. For a polynomial-size circuit $F : \{0,1\}^n \to \{0,1\}$ (we are eventually going to pick F to be a modification of the circuit C from above), a smooth PCPP verifier $V_{\mathsf{C\text{-}EVAL}}(F)$ for F[5] takes as input a proof π of length $\mathrm{poly}(n)$ and $O(\log n)$ random bits, and makes a constant number of uniformly distributed, non-adaptive queries to the proof and the input (i.e. which bits are queries depend only on the random bits, and each bit has an equal probability of being queried). Moreover, for some small constant $\delta_p > 0$:

- (PCPP Completeness) If $F(\tau) = 1$, then there is a proof π such that $V_{\mathsf{C\text{-}EVAL}}(F)^{\tau\circ\pi}$ always accepts. Moreover, there is a polynomial-time algorithm which computes π given F and τ.
- (PCPP Soundness) If $F(\tau') = 0$ for every $\tau' \in \{0,1\}^n$ which differs from τ in at most a δ_p fraction of entries, then $\Pr_{u\in\{0,1\}^{O(\log n)}}[V_{\mathsf{C\text{-}EVAL}}(F)^{\tau\circ\pi}(u) = 1] \le 1/3$ for all possible proofs $\pi \in \{0,1\}^{\ell_{\mathsf{proof}}}$.

The fact that the queries are both smooth and non-adaptive will be crucial to our construction later on. This 'proximity' aspect of the soundness condition is necessary for these properties to hold. For instance, if one fixed F to be the parity function, then it is not hard to see that such a construction without the 'proximity' aspect (i.e. with $\delta_p = 0$) is impossible. Our goal is to apply such a smooth PCPP to C, but since we don't have any guarantees about which inputs C should reject, we will first need to make some modifications to C to deal with the 'proximity' aspect.

[5] More precisely, $V_{\mathsf{C\text{-}EVAL}}(F)$ is a smooth PCPP for the $\mathsf{Circuit\text{-}Eval}$ problem, in which we have fixed the circuit to be F.

Defining $D_C(\tau) := V(1^n)^C(\tau)$, which is still a polynomial-size circuit, our goal is to design a fast algorithm to estimate

$$p_{\mathsf{acc}}(C) := \Pr_{\tau \in \{0,1\}^\ell}[V(1^n)^C(\tau) = 1] = \Pr_{\tau \in \{0,1\}^\ell}[D_C(\tau) = 1].$$

In preparation for using the PCP of proximity, we next apply an *error correcting code* to τ. Specifically, fix a constant-rate $\mathbb{F}_2$-linear error correcting code ECC with efficient encoder $\mathsf{Enc} : \{0,1\}^\ell \to \{0,1\}^{c_1 \cdot \ell}$ and decoder $\mathsf{Dec} : \{0,1\}^{c_1 \cdot \ell} \to \{0,1\}^\ell$ which can recover error up to a δ_{dec} fraction. We define another circuit $E_C : \{0,1\}^{c_1 \cdot \ell} \to \{0,1\}$, as $E_C(w) := D_C(\mathsf{Dec}(w))$. That is, E_C treats the input as a codeword of ECC, decodes it, and feeds the result into the circuit D_C. Now our goal is to estimate

$$p_{\mathsf{acc}}(C) = \Pr_{\tau \in \{0,1\}^\ell}[E_C(\mathsf{Enc}(\tau)) = 1].$$

Now we will apply the PCP of Proximity to simplify the estimation of $p_{\mathsf{acc}}(C)$. More precisely, we use a $q = O(1)$ query smooth PCPP, $V_{\mathsf{C\text{-}EVAL}}(E_C)$, for the circuit E_C, which has proximity parameter $< \delta_{\mathsf{dec}}$, proof length $\ell_{\mathsf{proof}} = \mathrm{poly}(\mathsf{SIZE}(E_C)) = \mathrm{poly}(n)$, and number of random bits $m = O(\log \ell_{\mathsf{proof}}) = O(\log n)$. The crucial observation here is that we have dealt with the 'proximity' aspect of the smooth PCPP by using the error correcting code: if $D_C(\tau) = 0$, then $\mathsf{Enc}(\tau)$ is δ_{dec}-far from any yes-inputs to E_C. This is because, for any $w \in \{0,1\}^{c_1 \cdot \ell}$ which is δ_{dec}-close to $\mathsf{Enc}(\tau)$, w decodes to τ and $E_C(w) = D_C(\tau) = 0$.

Summarizing, so far we have the following:

- (PCPP Completeness) If $D_C(\tau) = 1$, then there is a proof $\pi \in \{0,1\}^{\ell_{\mathsf{proof}}}$ such that $V_{\mathsf{C\text{-}EVAL}}(E_C)^{\mathsf{Enc}(\tau) \circ \pi}$ always accepts. Moreover, given E_C and τ, there is a polynomial-time computable function $\pi(E_C, \tau) \in \{0,1\}^{\ell_{\mathsf{proof}}}$ to compute the proof π.
- (PCPP Soundness) If $D_C(\tau) = 0$, then $\Pr_{u \in \{0,1\}^m}[V_{\mathsf{C\text{-}EVAL}}(E_C)^{\mathsf{Enc}(\tau) \circ \pi}(u) = 1] \le 1/3$ for all possible proofs $\pi \in \{0,1\}^{\ell_{\mathsf{proof}}}$.

The P^{NP} Machine M_{rigid}. Finally, we are ready to define our rigid matrix. It will be the concatenation, over all $\tau \in \{0,1\}^\ell$, of the proof $\pi(E_{C_{\mathsf{best}}}, \tau)$ from the PCPP Completeness condition above. More precisely, let $\pi_{C_{\mathsf{best}}}(\tau, j)$ be the j-th bit of $\pi(E_{C_{\mathsf{best}}}, \tau)$. Note that $\pi_{C_{\mathsf{best}}}$ is a Boolean function on $n_\pi := n + O(\log n)$ bits. Letting $N = 2^{n_\pi/2}$, we define our P^{NP} machine M_{rigid} as the function which, on input 1^N, outputs the truth-table of $\pi_{C_{\mathsf{best}}}$, which we interpret as a matrix in $\{0,1\}^{N \times N}$. M_{rigid} runs in P^{NP} since, similar to before, one can guess C_{best} bit-by-bit and verify with the NP oracle.

Again, assume toward a contradiction that $\mathscr{R}_{M_{\mathsf{rigid}}(1^N)}(r) \le \delta \cdot N^2$ for a small constant δ (recall that one can think of $r := 2^{(\log N)^{1-\varepsilon}}$ for a small constant $\varepsilon > 0$) for all N. That is, we know $\pi_{C_{\mathsf{best}}}(\cdot, \cdot)$ can be $(1-\delta)$-approximated by a matrix M of rank at most r. We guess a low-rank decomposition of that matrix $M = A \cdot B$, in $O(N \cdot r)$ time, and now we wish to estimate

$$p_{\mathsf{acc}}(M) := \Pr_{u \in \{0,1\}^m, \tau \in \{0,1\}^\ell}[V_{\mathsf{C\text{-}EVAL}}(E_C)^{\mathsf{Enc}(\tau) \circ M(\tau, \cdot)}(u) = 1].$$

Recall that τ is the randomness to the old PCP verifier V, of length $\ell = n + O(\log n)$, and u is the randomness to the new smooth PCPP verifier $V_{\mathsf{C\text{-}EVAL}}(E_C)$, of length $m = O(\log n)$.

Fast Algorithm for Computing $p_{\mathsf{acc}}(M)$. We now use the fact that the queries made by $V_{\mathsf{C\text{-}EVAL}}(E_C)$ *only depend on* u. Our algorithm will simply iterate over all $\mathrm{poly}(n)$ choices of u. Hence, fix $u \in \{0,1\}^m$, and suppose V queries $M(\tau, j_1), M(\tau, j_2), \ldots, M(\tau, j_{q_1})$ in $M(\tau, \cdot)$, and $e_1, e_2, \ldots, e_{q_2}$ in $\mathsf{Enc}(\tau)$. Now we want to estimate

$$\Pr_{\tau \in \{0,1\}^\ell}[F_u(M(\tau, j_1), M(\tau, j_2), \ldots, M(\tau, j_{q_1}), \mathsf{Enc}(\tau)_{e_1}, \mathsf{Enc}(\tau)_{e_2}, \ldots, \mathsf{Enc}(\tau)_{e_{q_2}}) = 1]$$

for a Boolean function F_u on $q = q_1 + q_2 = O(1)$ inputs. Next, using a standard trick from the analysis of Boolean functions, we observe that since we are aiming to compute the expected value of F_u, we can assume that F_u is a *parity* function. In other words, it is sufficient to quickly estimate

$$\Pr_{\tau \in \{0,1\}^\ell}[M(\tau, j_1) + M(\tau, j_2) + \ldots + M(\tau, j_{q_1}) + \mathsf{Enc}(\tau)_{e_1} + \mathsf{Enc}(\tau)_{e_2} + \ldots + \mathsf{Enc}(\tau)_{e_{q_2}} = 1],$$

where the sum is taken mod 2. The parity of $M(\tau, j_1) + M(\tau, j_2) + \ldots + M(\tau, j_{q_1})$, which is a sum of a constant number of low-rank matrices, can itself be written as a low rank matrix. Since Enc is a linear function over $\mathbb{F}_2$, incorporating $\mathsf{Enc}(\tau)_{e_1} + \mathsf{Enc}(\tau)_{e_2} + \ldots + \mathsf{Enc}(\tau)_{e_{q_2}}$, which is a linear function of the *indices* of the matrix, can only increase the rank by an additive constant. Hence, our goal is exactly to compute the number of 1s in a low rank matrix. This is an instance of the previously discussed $\#\mathsf{SAT}$ problem for low-rank matrices which, as discussed, can can be solved in $N^{2-\Omega(1/\log r)}$ time as described by [CW16].

Notice that:

- If $1^n \in L$, and we guessed the circuit C_{best} and a matrix M which $(1-\delta)$-approximates $\pi_{C_{\mathsf{best}}}$, then $p_{\mathsf{acc}}(M) \geq 1 - q \cdot \delta$ since $V_{\mathsf{C\text{-}EVAL}}(E_C)$'s queries are *smooth* (meaning, uniformly distributed over the proof).
- If $1^n \notin L$, then for all possibles guesses, $p_{\mathsf{acc}}(M) \leq 1/2$, by the soundness of PCPP and PCP.

Putting everything together, it follows that L is in non-deterministic time

$$\mathrm{poly}(n) \cdot N^{2-\Omega(1/\log r)} = 2^{n-\Omega(n/\log r)} = 2^{n-\Omega(n^{\varepsilon})},$$

contradicting the non-deterministic time hierarchy. This completes the proof overview for Theorem 1.9.

1.3 Unconditional Construction of Rigid Matrices

Getting Rid of the Easy-Witness Assumption: A Boot-Strapping Scheme. We now move on to a proof overview of Theorem 1.1. Note that in the above argument, the only consequence of $\mathsf{NE} \subset \mathsf{P}_{/\mathrm{poly}}$ used is the fact that $V(1^n)$ has a succinct witness circuit. In order to get rid of the assumption $\mathsf{NE} \subset \mathsf{P}_{/\mathrm{poly}}$, we next show how to construct a succinct witness for $V(1^n)$ solely based on the assumption that all P^{NP} machines have non-rigid output matrices.

The key idea is based on a *bootstrapping* argument. Observe that an $N^{o(1)}$-rank decomposition of a matrix $M \in \{0,1\}^{N \times N}$ actually compresses the N^2 bits of M into an $N^{1+o(1)}$ bit representation. If we can further treat those bits after the compression as a low-rank matrix, and compress it again, and so on, we can further reduce the number of bits required to represent the matrix.

A key property of low-rank decompositions we will use is that they are *locally decodable*. That is, if A, B are the two matrices of a rank-r expression for M, then one can compute a particular entry $M_{i,j}$, by looking at only $O(r)$ entries of the matrices A and B (the ith row of A and the jth column of B).

High-Level Idea. Recall from the proof above that O_n is the lexicographically first oracle which makes $V(1^n)$ always accept. The high level idea for constructing a succinct witness for O_n is as follows. We first interpret O_n as a matrix $M_1 \in \{0,1\}^{N_1 \times N_1}$. Letting (A_1, B_1) be its low-rank decomposition, we then interpret the concatenation (A_1, B_1) as a matrix $M_2 \in \{0,1\}^{N_2 \times N_2}$. We will show that M_2 also has a low-rank decomposition (A_2, B_2). We then interpret this as a matrix $M_3 \in \{0,1\}^{N_3 \times N_3}$, and repeat until we have a small enough matrix $M_k \in \{0,1\}^{N_k \times N_k}$. Note that for all i, we have $N_i = N_{i-1}^{1/2+o(1)}$; that is, each time we compress the bits by about a square-root.

Why do all these matrices have low-rank approximations? This follows from our assumption that all P^{NP} machines' output matrices are non-rigid, and hence have low-rank approximations. First, similar to before, we know that there is a P^{NP} algorithm M that, on input $1^{\sqrt{|O_n|}}$, outputs $O_n = M_1$. Then, we can recursively show that each of the matrices $M_2, \dots, M_k$ can be constructed by an NP oracle machine: for each i, to construct M_i, we use the oracle to find the lexicographically first low-rank approximation of M_{i-1}.

Our succinct witness for O_n is M_k for a large constant k. M_k is small enough that we can construct a circuit for it by brute-force. The idea is to then repeatedly use the local decoding scheme we discussed earlier to construct circuits for $M_{k-1}, M_{k-2}, \dots, M_1$, since each corresponds to a low-rank approximation of the next. However, having a low-rank approximation of M_i is not enough to recover M_i exactly. To circumvent this issue, we apply *locally-decodable codes* to the matrices. Indeed, if our low-rank decomposition $A_i \cdot B_i$ gives a $(1-\delta)$-approximation to the matrix $\mathsf{Enc}(M_i)$ (the encoding of M_i using a suitable locally-decodable code), rather than to M_i, then we can use the local decoder to compute M_i exactly.

Locally-Decodable Codes and the Actual Compression Scheme $f_i(\cdot)$. We now give more details of the construction. We fix a locally-decodable code $\mathsf{ECC}_{\mathsf{local}}$, with message length $n^{1+\varepsilon_{\mathsf{enc}}}$ ($\varepsilon_{\mathsf{enc}}$ can be made an arbitrarily small constant), and a $\operatorname{polylog}(n)$-time local decoder. Let the encoder be $\mathsf{Enc} : \{0,1\}^n \to \{0,1\}^{n^{1+\varepsilon_{\mathsf{enc}}}}$. The local decoder implies that, for $S \in \{0,1\}^n$, if we have a T-size circuit which approximates the string $\mathsf{Enc}(S)$, then there is a $\operatorname{polylog}(n) \cdot T$-size circuit which computes S exactly.

We now define two functions to describe how to go from a matrix to its low-rank decomposition. First define the function $\mathsf{rk}(N) = 2^{(\log N)^b}$ for a constant $b > 0$. Then, for a string S, define $\mathsf{comp}(S)$ as follows: Let $N = \sqrt{|S|}$ (we will pretend here that $|S|$ is the square of an integer; in the real proof we use a slight padding to make sure of this), and let A, B be two matrices in $\{0,1\}^{N \times \mathsf{rk}(N)}$ and $\{0,1\}^{\mathsf{rk}(N) \times N}$, respectively, such that $A \cdot B$ equals S on the most possible positions (viewing S as a matrix in $\{0,1\}^{N \times N}$). If there are multiple equally good options for A, B, then pick the lexicographically first one. We then define $\mathsf{comp}(S) = A \circ B$, as the concatenation of matrices A and B.

Next, we define a series of functions which recursively give compressions of a given string S:

$$f_i(S) := \begin{cases} \mathsf{Enc}(S) & i = 1, \\ \mathsf{Enc}(\mathsf{comp}(f_{i-1}(S))) & i \geq 2. \end{cases}$$

Note that A and B (the outputs of $\mathsf{comp}(S)$) can be computed from S in $\mathsf{TIME}[\operatorname{poly}(|S|)]^{\mathsf{NP}}$. Now, we set $\ell_{n,i} = \sqrt{|f_i(O_n)|}$. We can then pick our NP oracle machine to, on input $1^{\ell_{n,i}}$, output the corresponding matrix for $f_i(O_n)$. (In the full proof below we use some simple tricks to make sure the $\ell_{n,i}$'s are all distinct.) Therefore, by assumption, we know that each $f_i(O_n)$ can be approximated by a $\mathsf{rk}(\ell_{n,i})$-rank matrix.

Finally, we are ready to implement our bootstrapping. We know that, for a parameter j, $f_j(O_n)$ has an $\ell_{n,j}$-size circuit which computes it *exactly*. Suppose we have a T-size circuit C_j which $(1-\delta)$-approximates $f_j(O_n)$. From this we can construct a circuit C_{j-1} which $(1-\delta)$-approximates $f_{j-1}(O_n)$ as follows:

- First, applying the local decoder for $\mathsf{ECC}_{\mathsf{local}}$, we can construct a $\operatorname{polylog}(\ell_{n,j-1}) \cdot T$-size circuit C_{comp} which exactly computes $\mathsf{comp}(f_{j-1}(O_n))$.
- Let A, B be the matrices corresponding to $\mathsf{comp}(f_{j-1}(O_n))$. By our assumption, $A \cdot B$ is a $(1-\delta)$-approximation for $f_{j-1}(O_n)$. We know that $(A \cdot B)_{x,y}$ can be computed in $\mathsf{rk}(\ell_{n,j-1})$ time, given oracle access to C_{comp}. It follows from the locally-decodable property of $\mathsf{ECC}_{\mathsf{local}}$ that we get a circuit of size $\mathsf{rk}(\ell_{n,j-1}) \cdot \operatorname{polylog}(\ell_{n,j-1}) \cdot T$ which approximates $f_{j-1}(O_n)$.

From this construction, we can show that $f_1(O_n) = \mathsf{Enc}(O_n)$ can be approximated by a small circuit, which in turn shows O_n has a small exact circuit. It is not hard to see, in particular, that

$$\mathsf{SIZE}(O_n) \le \prod_{i=1}^{j-1} \mathsf{rk}(\ell_{n,j-1})^{1+o(1)} \cdot |\ell_{n,j}| = \operatorname{poly}(\mathsf{rk}(|O_n|)) \cdot |\ell_{n,j}| = 2^{O(n^b)} \cdot |\ell_{n,j}|.$$

The Constant $1/4 - \varepsilon$. Supposing that O_n has a 2^{n^a}-size witness, and the rank we consider is $\mathsf{rk}(N) = 2^{(\log N)^b}$, the running time of our algorithm is

$$2^{O(n^a)} \cdot 2^{n-\Omega(n^{1-b})} = 2^{n+O(n^a)-\Omega(n^{1-b})}.$$

In order to make the above faster than 2^n and get a contradiction, we want to pick $b < 1-a$. From the bound on $\mathsf{SIZE}(O_n)$, we can see that the bootstrapping scheme can only achieve $a > b$. Therefore, we set $a = 1/2 + \varepsilon$ and $b = 1/2 - 2\varepsilon$, for a small constant $\varepsilon > 0$.

We now consider the running time of M_{comp}. Since we only aim to compress O_n to a witness of size $2^{O(n^a)}$, we can stop if we find $\ell_{n,j} \le 2^{n^a}$, as there is no need to further compress. Let $M := \ell_{n,j}$. On input 1^M, M_{comp} needs $\operatorname{poly}(\ell_{n,1}) = 2^{O(n)} \le M^{\log M}$ time to compute $f_i(O_n)$. Therefore, M_{comp} runs in $\mathsf{TIME}[n^{\log n}]^{\mathsf{NP}}$, and hence yields a rigid matrix constructible in $\mathsf{TIME}[n^{\log n}]^{\mathsf{NP}}$ for rank $2^{(\log N)^{1/2-2\varepsilon}}$. In other words, the time is slower than we hoped for, but the rank is higher than we hoped for.

Finally, we use a tensor product argument (Lemma 2.7 below) to transform this into a P^{NP} construction, which is rigid for a worse rank of $2^{(\log N)^{1/4-\varepsilon}}$. The idea is to take the tensor product of our rigid matrix with a large all-1s matrix. The resulting matrix is still rigid for the same rank, but has larger dimensions. Equivalently, in terms of the dimension N of the matrix, the complexity to compute the matrix has gone down, but it is also rigid for a lower rank.

1.4 Other Related Work

Explicit Construction Based on Complexity-Theoretical Ideas. In a recent breakthrough work, Oliveira and Santhanam [OS17] gave an infinite often *pseudodeterministic* construction of primes in *sub-exponential time*. (That is, given an input 1^N, the (randomized) algorithm outputs a fixed prime P_N of N bits with high probability, for infinite number of N's, and it runs in sub-exponential time). Their results are similar to ours in that they construct algebraic objects by building on complexity-theoretic ideas.

Our approach differs from theirs in several ways. First, [OS17] make crucial use of the fact that primes can be recognized in polynomial-time [AKS04], while in contrast, testing whether a matrix is rigid is coNP-complete (cf, Proposition 29 of [Des07]). Second, their results build on *hardness vs randomness*, and a crucial component of their arguments is to use special pseudo-random generators to hit the set of all N-bit primes, while our results build on Williams' algorithmic approach to lower bounds [Wil13, Wil14b]: we show one can contradict the non-deterministic time hierarchy theorem, assuming there is no P^{NP} construction of rigid matrices.

Conditional Explicit Construction of Rigid Matrices. There are several works achieving P-time construction of rigid matrices under strong complexity assumptions. They are all based on the hardness-vs-randomness paradigm [NW94]. The observation is that since checking rigidity is in coNP, the ability of fooling a non-deterministic algorithm implies the ability to construct rigid matrices.

In [KvM02], it is shown that under the assumption that E has no $2^{o(n)}$-size SAT-oracle circuits, there is a P-time construction of matrices M_N over $\mathbb{Z}_{p(N)}$ such that $\mathscr{R}(M_N)(r) \geq \Omega((n-r)^2/\log n)$, where $p(N)$ is prime bounded by a polynomial of N. [MV05] give the same construction under the weaker assumption that E has no $2^{o(n)}$-size non-deterministic circuits[6]. In [GST03], the same construction is achieved with a uniform assumption that E has no $2^{o(n)}$-time Arther-Merlin protocols.

Lower Bounds on w_2. Recently announced concurrent work by Kumar and Volk [KV19] also constructs matrices with high w_2. Among other results, they show that there are constants $a, b, c > 0$ and a family $\{A_N\}_{N\in\mathbb{N}}$ such that A_N is an $N \times N$ matrix over an extension of $\mathbb{F}_2$ of degree $\exp(N^{1-a})$ which can be computed in time $\exp(N^{1-b})$ and with $w_2(A_N) > N^{1+c}$. By comparison, our Theorem 1.6 constructs $N \times N$ matrices H_N in P^{NP} with the worse lower bound $w_2(H_N) \geq \Omega(N \cdot 2^{(\log N)^{1/4-\varepsilon}})$, but our matrices are over $\mathbb{F}_2$ instead of a large extension field. Their techniques seem very different from ours, although they also use a padding trick, similar to our Lemma 2.7, of taking the Kronecker product of a rigid matrix with a large simple matrix to decrease its computational complexity in terms of the matrix size.

Circuit Lower Bounds via PCPP. In a recent work, Chen and Williams [CW19b] applied PCPP to show that in order to prove $\mathscr{C}$ lower bound for various non-deterministic time classes such as NEXP or NQP, it suffices to derandomize $\oplus_2 \circ \mathscr{C}$ circuits (an XOR of two $\mathscr{C}$ circuits) in a better-than-2^n time. The proof crucially combines the forgoing derandomization algorithms and PCPP to obtain a non-trivial derandomization of *general circuits*. Here, our proof for Theorem 1.2 makes similar, but more sophisticated use of PCPP. In particular, we actually require the PCPP to be *smooth*, which is not required in [CW19b]. Our proof for Theorem 1.1 also relies on a completely different bootstrapping argument, which is specific for our task of constructing rigid matrices.

Rigidity and Data Structure Lower Bounds. Recent work by Dvir, Golovnev, and Weinstein [DGW19] showed connections between rigidity and static data structure lower bounds. In particular, they posed the challenge of constructing rigid matrices in P^{NP} or E^{NP} as an avenue toward proving new data structure lower bounds. Unfortunately, the parameters of our new P^{NP} construction do not seem to yield any new bounds using their approach.

2 Preliminaries

Our construction of rigid matrices makes use of a number of tools from the complexity theory literature; in this Section we precisely define the tools from prior work which we will use.

The *Circuit Evaluation Problem* ($\mathsf{Circuit\text{-}Eval}$) is the language of pairs (C, w) where C is a general fan-in-2 circuit, and w is an input such that $C(w) = 1$. For two strings a, b, we use $a \circ b$ to denote their concatenation.[7]

[6] Indeed, the requirement is E has no $2^{o(n)}$-size SV-nondeterministic circuits, which is the non-uniform analogue of $\mathsf{NP}\cap\mathsf{coNP}$; see [MV05] for details.

[7] The symbol $\circ$ is also used for circuit composition; its meaning will always be clear from context.

2.1 Probabilistic Checkable Proofs of Proximity

Our proof will make heavy use of probabilistically checkable proofs of proximity.

Definition 2.1 (Probabilistic Checkable Proofs of Proximity (PCP of proximity, or PCPP))**.** For $s, \delta : \mathbb{N} \to [0,1]$ and $r, q : \mathbb{N} \to \mathbb{N}$, a verifier V is a PCP of proximity system for a pair language L with proximity parameter δ, soundness parameter s, number of random bits r and query complexity q if the following holds for all x, y:

- (Completeness) If $(x, y) \in L$, then there is a proof π such that $V(x)$ accepts oracle $y \circ \pi$ with probability 1.
- (Soundness) If y is $\delta(|x|)$-far from $L(x) := \{z : (x, z) \in L\}$, then for all proofs π, $V(x)$ accepts oracle $y \circ \pi$ with probability at most $s(|x|)$.
- $V(x)$ tosses $r(|x|)$ random coins, and makes at most $q(|x|)$ non-adaptive queries.

Lemma 2.2 ([BGH$^+$06, Theorem 3.3])**.** *For any constants $0 < \delta, s < 1$, there is a PCP of proximity system for Circuit-Eval with proximity δ, soundness s, number of random bits $r = O(\log n)$ and query complexity $q = O(1)$. Moreover, given the pair $(C, w) \in$ Circuit-Eval, a proof π which makes $V(C)$ always accept can be constructed in $\mathrm{poly}(|C| + |w|)$ time.*

Remark 2.3. *The last ('Moreover') sentence is not explicitly stated in [BGH$^+$06], but it is evident from their construction.*

In this paper, we need a stronger PCPP construction which is additionally *smooth*, meaning, every position in the proof π is queried with equal probability (assuming without loss of generality that all queries are non-adaptive and distinct). Such a construction can be found in [Par19].[8]

Lemma 2.4 ([Par19])**.** *For any constants $0 < \delta, s < 1$, there is a smooth PCP of proximity system for Circuit-Eval with proximity δ, soundness s, number of random bits $r = O(\log n)$ and query complexity $q = O(1)$. Moreover, given the pair $(C, w) \in$ Circuit-Eval, a proof π making $V(C)$ always accepts can be constructed in $\mathrm{poly}(|C| + |w|)$ time.*

2.2 Error Correcting Codes

We also need standard constructions of two different types of codes: constant-rate linear error correcting codes, and n^{ε}-rate codes with $\mathrm{polylog}(n)$ time local decoders.

Lemma 2.5 ([Spi96])**.** *There is a constant-rate linear error correcting code ECC with a linear-time encoder Enc and a linear-time decoder Dec recovering error up to a universal constant δ.*

Lemma 2.6 (cf, Section 2.3 of [Yek12])**.** *For any constant $\varepsilon > 0$, there is a n^{ε}-rate error correcting code ECC with a $\mathrm{poly}(n)$-time encoder Enc and a $\mathrm{polylog}(n)$-time local-decoder Dec which recovers up to a 0.01 fraction of errors.*

[8] [Par19]'s construction actually ensures that this holds for every query position in the second input y as well. This additional property is not required by our proof.

2.3 A Simple Fact About Matrix Rigidity

We use $\mathbf{1}_N$ to denote the all-ones matrix of size $N \times N$, and $\otimes$ to denote the Kronecker product of matrices.

Lemma 2.7. *For any field $\mathbb{F}$ and any matrix $A \in \mathbb{F}^{M \times M}$, we have*

$$\mathscr{R}_{\mathbf{1}_N \otimes A}(r) = \mathscr{R}_A(r) \cdot N^2.$$

Proof. We first show $\mathscr{R}_{\mathbf{1}_N \otimes A}(r) \geq \mathscr{R}_A(r) \cdot N^2$. Assume to the contrary that there is a way to change $k < \mathscr{R}_A(r) \cdot N^2$ entries of $\mathbf{1}_N \otimes A$ to make its rank r. The matrix $\mathbf{1}_N \otimes A$ consists of N^2 disjoint copies of A, so by the pigeonhole principle, there were at most $k/N^2 < \mathscr{R}_A(r)$ entries changed in one of those copies of A. Thus, that submatrix still has rank greater than r after the change, a contradiction.

We next show that $\mathscr{R}_{\mathbf{1}_N \otimes A}(r) \leq \mathscr{R}_A(r) \cdot N^2$. Let B a matrix of rank r whose Hamming distance from A is $\mathscr{R}_A(r)$. Thus, $\mathbf{1}_N \otimes B$ has rank $\operatorname{rank}(\mathbf{1}_N) \cdot \operatorname{rank}(B) = r$, and its Hamming distance from $\mathbf{1}_N \otimes A$ is $\mathscr{R}_A(r) \cdot N^2$. □

2.4 $\mathbb{F}_{p^r}$-#OV

One crucial component of our construction is the algorithm for $\mathbb{F}_{p^r}$-#OV from [CW16].

Definition 2.8. For a prime power $q = p^r$, in an $\mathbb{F}_q$-#$\mathsf{OV}_{n,d}$ instance, we are given two collections of vectors from $\mathbb{F}_q^d$, $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$, and want to compute the number of pairs such that $\langle a_i, b_j \rangle = 0$ over $\mathbb{F}_q$.

We use the following algorithm for $\mathbb{F}_q$-#$\mathsf{OV}_{n,d}$.

Theorem 2.9 ([CW16])**.** *For all fixed prime powers $q = p^r$, there is an $n^{2-\Omega(1/\log(d/\log n))}$ time deterministic algorithm for $\mathbb{F}_q$-#$\mathsf{OV}_{n,d}$, when $d = n^{o(1)}$.*

The original paper [CW16] only states an algorithm for $\#\mathsf{OV}$ (the problem when A and B are collections of vectors from $\{0,1\}^d$ and the inner product is over $\mathbb{Z}$). We make two small modifications to their algorithm to get the result stated in Theorem 2.9 above; see Appendix A for details.

3 Easy-Witness and Rigidity Matrix Construction in P^{NP}

In this section we prove Theorem 1.2.

We say an algorithm is a *matrix-constructing algorithm* if on input 1^N, it outputs a matrix in $\{0,1\}^{N \times N}$. We say a function $f : \mathbb{N} \to \mathbb{N}$ is a *typical resource bound function* if it is strictly increasing, and satisfies $f(n) = \omega(f(n+1)/(n+1))$. We first prove the following lemma, which says that if certain nondeterministic time classes have easy witnesses, then there is a P^{NP} construction of rigid matrices.

Lemma 3.1. *There is an absolute constant $\delta > 0$ such that, for all prime powers $q = p^r$, and any three typical resource bound functions $T, S, R : \mathbb{N} \to \mathbb{N}$ with $T(n), S(n) \geq n$ for all n, the following three conditions cannot hold simultaneously.*

- *All polynomial-time verifiers[9] for unary $\mathsf{NTIME}[T(n)]$ languages have $S(n)$-size witness circuits.*

[9] That is, for $L \in \mathsf{NTIME}[T(n)]$, the verifier V takes two inputs x, y with $|x| = n$ and $|y| = \operatorname{poly}(T(n))$, runs in $\operatorname{poly}(|x|+|y|)$ time, and has the property that $x \in L$ if and only if there is a y such that $V(x,y) = 1$.

- *For all* P^{NP} *matrix-constructing algorithms* M*,* $\mathscr{R}_{M(1^N)}(R(N)) \leq \delta \cdot N^2$ *for almost all* N*.*

- $\log T(n)/\log R(N) = \omega(\log\log T(n) + \log S(n))$*, where* $N = 2^{n_\pi/2}$*, for* $n_\pi = \log T(n) + O(\log\log T(n)) + O(\log S(n))$*, and* $R(N) = N^{o(1)}$*.*

Remark 3.2. *In the following proof, we will actually only need the first assumption to hold for the special* PCP *verifier* $V(1^n)$ *of the language* L *we consider. This remark will be useful in the proof in the next Section.*

Proof. Let $\delta > 0$ be a constant to be decided later. We first only consider the case when the field is $\mathbb{F}_2$, and then show how to generalize the argument for other finite fields. We will assume that all three items are true, and derive a contradiction.

Unary Language L and PCP. Let L be a unary language in $\mathsf{NTIME}[T(n)] \setminus \mathsf{NTIME}[T(n)/n]$. Using the non-deterministic time hierarchy theorem [Zák83], such an L exists because $T(n)$ is a typical resource function. Let V be an efficient PCP verifier for L from [BV14]. That is, there is a function $\ell = \ell(n) = \log T(n) + O(\log\log T(n))$, such that $V(1^n)$ takes an oracle $O : \{0,1\}^\ell \to \{0,1\}$ and ℓ random bits as input, runs in $\mathrm{poly}(n)$ time, and:

1. (PCP Completeness) If $1^n \in L$, then there exists a circuit $C : \{0,1\}^\ell \to \{0,1\}$ of size $S(n)$ such that $\Pr_{r\in\{0,1\}^\ell}[V(1^n)^C(r) = 1] = 1$. (This follows from the first assumption of the Lemma.)

2. (PCP Soundness) If $1^n \notin L$, then for all oracles $O : \{0,1\}^\ell \to \{0,1\}$, we have $\Pr_{r\in\{0,1\}^\ell}[V(1^n)^O(r) = 1] \leq 1/n$.

We will next show how to put $L \in \mathsf{NTIME}[T(n)/n]$ by using the second and the third assumptions of the Lemma, which will give us the contradiction we want.

The Plan. Let C_{best} be the circuit of size $S(n)$ such that $\Pr_{r\in\{0,1\}^\ell}[V(1^n)^{C_{\mathsf{best}}}(r) = 1] = 1$, and if there are multiple such circuits, we break the tie by choosing the lexicographically first one. Note that such a circuit doesn't exist when $1^n \notin L$, and in that case we set C_{best} to be a trivial circuit which always outputs 0.

In our non-deterministic algorithm to solve L, given an input 1^n, we first guess a circuit C of size at most $S(n)$, and wish to ensure that the following two conditions hold:

1. When $1^n \in L$ and $C = C_{\mathsf{best}}$, we accept, and

2. When $1^n \notin L$, we always reject.

If our algorithm satisfies these two conditions and runs in $T(n)/n$ non-deterministic time, then we have put $L \in \mathsf{NTIME}[T(n)/n]$ and arrived at the desired contradiction.

Implementation. Now suppose we have guessed a circuit C of size at most $S(n)$. Toward achieving the tweo conditions above, we want to estimate

$$p_{\mathsf{acc}}(C) := \Pr_{r\in\{0,1\}^\ell}[V(1^n)^C(r) = 1].$$

Define another circuit $D_C : \{0,1\}^\ell \to \{0,1\}$ as $D_C(r) := V(1^n)^C(r)$. We thus equivalently have that

$$p_{\mathsf{acc}}(C) = \Pr_{r\in\{0,1\}^\ell}[(D_C, r) \in \textsf{Circuit-Eval}],$$

by the definition of Circuit-Eval.

Applying Error Correcting Codes. Fix an $\mathbb{F}_2$-linear error correcting code ECC with rate c_1 and recovering error δ_1, whose existence is guaranteed by Lemma 2.5. Let $\mathsf{Enc} : \{0,1\}^\ell \to \{0,1\}^{c_1\cdot\ell}$ and $\mathsf{Dec} : \{0,1\}^{c_1\cdot\ell} \to \{0,1\}^\ell$ be the corresponding linear-time encoder and decoder.

We now define yet another circuit $E_C : \{0,1\}^{c_1\cdot\ell} \to \{0,1\}$ as $E_C(w) := D_C(\mathsf{Dec}(w))$. Then it suffices to estimate

$$p_{\mathsf{acc}}(C) = \Pr_{r\in\{0,1\}^\ell}[(E_C, \mathsf{Enc}(r)) \in \textsf{Circuit-Eval}].$$

Notice that $\mathsf{SIZE}(E_C) \le \mathrm{poly}(n)\cdot S(n)$, since the verifier $V(1^n)$ runs in $\mathrm{poly}(n)$ time, and the decoder Dec runs in linear time.

Applying the PCPP. Now we use a $q_{\mathsf{PCPP}} = O(1)$-query smooth PCPP for Circuit-Eval from Lemma 2.4 with constant soundness s_{PCPP} and proximity parameter δ_{PCPP} to be specified later. Let $V_{\mathsf{C\text{-}EVAL}}(E_C)$ be the verifier for this smooth PCPP with the circuit fixed to E_C. Hence, $V_{\mathsf{C\text{-}EVAL}}(E_C)$ uses proof length $\ell_{\mathsf{proof}} = \mathrm{poly}(\mathsf{SIZE}(E_C)) = \mathrm{poly}(S(n))$ and $m = O(\log \ell_{\mathsf{proof}})$ random bits.

Claim 1. *$V_{\mathsf{C\text{-}EVAL}}(E_C)$ satisfies the following three properties by setting $\delta_{\mathsf{PCPP}} < \delta_{\mathsf{dec}}$, and $s_{\mathsf{PCPP}} = 1/3$.*

1. *(PCPP Completeness) If $(D_C, r) \in$ Circuit-Eval, there is a proof $\pi \in \{0,1\}^{\ell_{\mathsf{proof}}}$ that*

$$\Pr_{u\in\{0,1\}^m}[V_{\mathsf{C\text{-}EVAL}}(E_C)^{\mathsf{Enc}(r)\circ\pi}(u)] = 1.$$

2. *(From PCPP Smoothness) Suppose $(D_C, r) \in$ Circuit-Eval, and let π be a proof satisfying the previous property. If proof $\widetilde{\pi} \in \{0,1\}^{\ell_{\mathsf{proof}}}$ is a $(1-\delta)$-approximation to π for some $\delta \in [0,1]$, then*

$$\Pr_{u\in\{0,1\}^m}[V_{\mathsf{C\text{-}EVAL}}(E_C)^{\mathsf{Enc}(r)\circ\widetilde{\pi}}(u)] \ge 1 - q_{\mathsf{PCPP}}\cdot\delta.$$

3. *(PCPP Soundness) If $(D_C, r) \notin$ Circuit-Eval, then for all proofs $\pi \in \{0,1\}^{\ell_{\mathsf{proof}}}$, we have*

$$\Pr_{u\in\{0,1\}^m}[V_{\mathsf{C\text{-}EVAL}}(E_C)^{\mathsf{Enc}(r)\circ\pi}(u)] \le 1/3.$$

Property (1) of Claim 1 follows from the completeness property of the PCPP system, and property (2) follows from the smoothness of the PCPP system combined with a simple union bound.

For property (3), note that if $(D_C, r) \notin$ Circuit-Eval, then $\mathsf{Enc}(r)$ is δ_{dec}-far from the set $\{w \in \{0,1\}^{c_1\cdot\ell} : (E_C, w) \in \textsf{Circuit-Eval}\}$. This is because for any string $w \in \{0,1\}^{c_1\cdot\ell}$ which is $< \delta_{\mathsf{dec}}$-close to $\mathsf{Enc}(r)$, we know $\mathsf{Dec}(w) = r$ and hence $(E_C, w) \notin$ Circuit-Eval. Therefore, by setting $\delta_{\mathsf{PCPP}} < \delta_{\mathsf{dec}}$, and $s_{\mathsf{PCPP}} = 1/3$, property (3) follows from the soundness of the PCPP system.

The Function $\pi_{C_{\mathsf{best}}}(r,j)$. Note that by Lemma 2.4, there is a polynomial time computable function $\pi(E_C, \mathsf{Enc}(r)) \in \{0,1\}^{\ell_{\mathsf{proof}}}$, such that when $(E_C, \mathsf{Enc}(r)) \in \mathsf{Circuit\text{-}Eval}$, we have

$$\Pr_{u \in \{0,1\}^m}[V_{\mathsf{C\text{-}EVAL}}(E_C)^{\mathsf{Enc}(r) \circ \pi(E_C, \mathsf{Enc}(r))}(u)] = 1.$$

Define the Boolean function $\pi_C(r,j)$, for $j \in [\ell_{\mathsf{proof}}]$, to be the j-th bit of $\pi(E_C, r)$ (suppose ℓ_{proof} is a power of 2 for simplicity).

The function $\pi_{C_{\mathsf{best}}}(r,j)$ is computable in E^{NP}, by using the following procedure. First we show how to compute the circuit C_{best} in E^{NP}. We are given two inputs r, j with length $|r| = \ell$ and $|j| = \log \ell_{\mathsf{proof}} = O(\log S(n))$. In $O(2^\ell)$ time with an NP oracle, we can first decide whether $V(1^n)$ always accepts a circuit of size at most $S(n)$. If not, then we just output a trivial circuit. If so, then we guess that circuit bit by bit to construct the lexicographically first one, again using the NP oracle to check each guess. In this way, we can compute C_{best} in E^{NP}. We can then construct the circuit $E_{C_{\mathsf{best}}}$ from C_{best}, and then (using the fact that $\pi(E_C, \mathsf{Enc}(r))$ can be computed in polynomial time) compute $\pi_{C_{\mathsf{best}}}(r)$ and output its j-th bit. The whole procedure runs in E^{NP}.

The P^{NP} Machine M_{rigid}. Note that $\pi_{C_{\mathsf{best}}}$ has input length $n_\pi = \ell + O(\log S(n))$. We can thus construct a P^{NP} machine M_{rigid} such that, given an input $1^{2^{n_\pi/2}}$, it outputs the truth-table of $\pi_{C_{\mathsf{best}}}$ as a matrix. Therefore, by the second assumption of the Lemma, $\pi_{C_{\mathsf{best}}}$ as a matrix can be δ-approximated by a matrix of rank $R(2^{n_\pi/2})$.

Putting L in $\mathsf{NTIME}[T(n)/n]$. Finally, consider the following algorithm for solving L. We first guess a circuit C of size $S(n)$, with the hope that it is C_{best}. Then, letting $N = 2^{n_\pi/2}$, we guess a matrix $M : N \times N$ of rank $R(N)$, with the hope that it δ-approximates $\pi_{C_{\mathsf{best}}}$. More specifically, we guess two matrices U, V of size $N \times R(N)$ and $R(N) \times N$, and set (implicitly, without explicitly computing it) $M = UV$.

Now we try to calculate

$$p_{\mathsf{acc}}(M) := \Pr_{r \in \{0,1\}^\ell, u \in \{0,1\}^m}[V_{\mathsf{C\text{-}EVAL}}(E_C)^{\mathsf{Enc}(r) \circ M(r)}(u) = 1].$$

Fix u, and suppose that for randomness u, the verifier $V_{\mathsf{C\text{-}EVAL}}(E_C)$ queries $M(r,j_1), M(r,j_2), \ldots, M(r,j_{q_1})$ in $M(r,\cdot)$, and $e_1, e_2, \ldots, e_{q_2}$ in $\mathsf{Enc}(r)$ (note that $V_{\mathsf{C\text{-}EVAL}}(E_C)$'s query positions only depend on the randomness u). Now we want to estimate

$$\Pr_{r \in \{0,1\}^\ell}[F_u(M(r,j_1), M(r,j_2), \ldots, M(r,j_{q_1}), \mathsf{Enc}(r)_{e_1}, \mathsf{Enc}(r)_{e_2}, \ldots, \mathsf{Enc}(r)_{e_{q_2}}) = 1]$$

for a Boolean function F_u on $q_{\mathsf{PCPP}} = q_1 + q_2$ inputs. First, we can write F_u in the basis of XOR functions:

$$F_u(z_1, z_2, \ldots, z_{q_{\mathsf{PCPP}}}) = \sum_{S \subseteq [q_{\mathsf{PCPP}}]} \alpha_S \cdot \bigoplus_{i \in S} z_i.$$

(Here, we consider the XOR function $\oplus$ to be outputting a $\{0,1\}$ value, and the coefficients α_S and the sum Σ are over $\mathbb{R}$, not over $\mathbb{F}_2$.) Since our goal is to compute the expected value of F_u, by linearity of expectation, it suffices to separately compute the expected value of each of the (constant number of) parity functions. Therefore, it suffices to consider the case when F_u is just an XOR function.

Also, note that since ECC is a linear code, it follows that $\mathsf{Enc}(r)_k$ is an XOR function on a subset of coordinates of r. Thus, if $r = a \circ b$ where $|a| = n_\pi/2$ and $|b| = |r| - |a|$ (note that $\ell > n_\pi/2$ by the third assumption of the Lemma), we have $\mathsf{Enc}(r)_e = \mathsf{Enc}_L(a)_e \oplus \mathsf{Enc}_R(b)_e$, where $\mathsf{Enc}_L(a)_e$ and $\mathsf{Enc}_R(b)_e$ are the corresponding contributions of a and b to $\mathsf{Enc}(r)_e$.

Next, we define

$$E_L(a)_e := \begin{cases} (1,0) & \text{if } \mathsf{Enc}_L(a)_e = 0, \\ (0,1) & \text{if } \mathsf{Enc}_L(a)_e = 1, \end{cases} \quad \text{and} \quad E_R(b)_e := \begin{cases} (0,1) & \text{if } \mathsf{Enc}_L(b)_e = 0, \\ (1,0) & \text{if } \mathsf{Enc}_L(b)_e = 1. \end{cases}$$

It is easy to verify that $\langle E_L(a)_e, E_R(b)_e \rangle = \mathsf{Enc}_L(a)_e \oplus \mathsf{Enc}_R(b)_e = \mathsf{Enc}(r)_e$.

Constructing $\mathbb{F}_2$-#OV Instance. We can now simplify the quantity we want to compute as

$$\begin{aligned} &\Pr_{a \in \{0,1\}^{n_\pi/2}, b \in \{0,1\}^{\ell - n_\pi/2}} \left[\bigoplus_{i=1}^{q_1} \langle U_a, V_{b \circ j_i} \rangle \oplus \bigoplus_{i=1}^{q_2} \langle E_L(a)_{e_i}, E_R(b)_{e_i} \rangle = 1 \right] \\ = &\Pr_{a \in \{0,1\}^{n_\pi/2}, b \in \{0,1\}^{\ell - n_\pi/2}} \left[\left\langle \bigcirc_{i=1}^{q_1} U_a \circ \bigcirc_{i=1}^{q_2} E_L(a)_{e_i} \circ 1, \bigcirc_{i=1}^{q_1} V_{b \circ j_i} \circ \bigcirc_{i=1}^{q_2} E_R(b)_{e_i} \circ 1 \right\rangle = 0 \right]. \end{aligned}$$

In above, we use U_i and V_j to denote the i-th row of U and j-th column of V respectively, so that $\langle U_i, V_j \rangle = M_{i,j}$. By duplicating each of the ‘b’s $2^{n_\pi - \ell}$ times, the above can be reduced to a counting $\mathbb{F}_2$-$\#\mathsf{OV}_{N,d}$ instance, with $N = 2^{n_\pi/2}$ vectors of $d = O(R(N))$ dimensions. By Theorem 2.9, this can be solved in time

$$\begin{aligned} N^{2-\Omega(1/\log d)} &= N^{2-\Omega(1/\log R(N))} \\ &= 2^{n_\pi - \Omega(n_\pi/\log R(N))} \\ &\le 2^{\log T(n) + O(\log\log T(n)) + O(\log S(n)) - \Omega(\log T(n)/\log R(N))}. \\ & \qquad (n_\pi = \ell + O(\log S(n)) = \log T(n) + \log\log T(n) + S(n)) \end{aligned}$$

Since we also need $\operatorname{poly}(S(n))$ time for enumerating all possible $u \in \{0,1\}^m$, the overall running time for calculating $p_{\mathsf{acc}}(M)$ is

$$2^{\log T(n) + O(\log\log T(n)) + O(\log S(n)) - \Omega(\log T(n)/\log R(N))}.$$

By our third assumption, we know the above running time is $\le 2^{\log T(n) - \omega(\log S(n))} \le T(n)/n$, since $S(n) \ge n$.

Analysis of the Algorithm. Consider first when $1^n \in L$. We know that on the correct guess of $C = C_{\mathsf{best}}$ and the appropriate $M \approx \pi_{C_{\mathsf{best}}}$, we have that M $(1-\delta)$-approximates $\pi_{C_{\mathsf{best}}}$. That is, for a random $r \in \{0,1\}^\ell$, the average relative distance between $M(r,\cdot)$ and $\pi_{C_{\mathsf{best}}}(r,\cdot)$ is at most δ. Hence, by Property (2) of Claim 1 and by linearity of expectation, we know that $p_{\mathsf{acc}}(M) > 1 - q_{\mathsf{PCPP}} \cdot \delta$ in this case.

Otherwise, if $1^n \notin L$, then for every guess of C and M, by the soundness property of PCP, we know that

$$\Pr_{r \in \{0,1\}^\ell} [(D_C, r) \in \textsf{Circuit-Eval}] \le 1/n.$$

Then by Property (3) of Claim 1, we have that

$$p_{\mathsf{acc}}(M) \le 1/n + 1/3 \le 1/2.$$

Therefore, when we set δ to be small enough so that $1 - q_{\mathsf{PCPP}} \cdot \delta > 1/2$, we can distinguish the above two cases. By the above argument, this puts $L \in \mathsf{NTIME}[T(n)/n]$, a contradiction.

Adaptation for the field $\mathbb{F}_q$. Let $q = p^r$ be a prime power. In the following, we sketch the adaptation to deal with $\mathbb{F}_q$. The only thing we need to modify is how to reduce the computation of $p_{\textsf{acc}}(M)$ to $\mathbb{F}_q$-#$\textsf{OV}$. Again, we guess a rank $R(N)$ matrix $M = UV$ over $\mathbb{F}_q$, and we want to calculate

$$\Pr_{r \in \{0,1\}^\ell}[F_u(M(r,j_1)^{q-1}, M(r,j_2)^{q-1}, \dots, M(r,j_{q_1})^{q-1}, \textsf{Enc}(r)_{e_1}, \textsf{Enc}(r)_{e_2}, \dots, \textsf{Enc}(r)_{e_{q_2}}) = 1]$$

for a Boolean function F_u on $q_{\textsf{PCPP}} = q_1 + q_2$ inputs. Note that in the above, we raise all the $M(r{,}j_i)$ inputs to the $(q-1)$-th power to make them Boolean. Now, we can write F_u as a real sum of $2^{q_{\textsf{PCPP}}}$ $\textsf{AND}$ functions, each one for a subset of the inputs of F_u. Hence, like before, it suffices to consider the case when F_u is an $\textsf{AND}$ function, and in this case we want to calculate

$$\Pr_{r \in \{0,1\}^\ell}\left[\prod_{i=1}^{q_1} M(r,j_i)^{q-1} \cdot \prod_{i=1}^{q_2} \textsf{Enc}(r)_{e_i} = 1\right],$$

which is equivalent to

$$\begin{aligned}
&\Pr_{a \in \{0,1\}^{n_\pi/2}} \Pr_{b \in \{0,1\}^{\ell - n_\pi/2}}\left[\prod_{i=1}^{q_1}\langle U_a, V_{b \circ j_i}\rangle^{q-1} \cdot \prod_{i=1}^{q_2}\langle E_L(a)_{e_i}, E_R(b)_{e_i}\rangle = 1\right]\\
=&\Pr_{a \in \{0,1\}^{n_\pi/2}} \Pr_{b \in \{0,1\}^{\ell - n_\pi/2}}\left[\left\langle\left(\bigotimes_{i=1}^{q_1} U_a^{\otimes(q-1)} \otimes \bigotimes_{i=1}^{q_2} E_L(a)_{e_i}\right) \circ 1, \left(\bigotimes_{i=1}^{q_1} V_{b \circ j_i} \otimes \bigotimes_{i=1}^{q_2} E_R(b)_{e_i}\right) \circ -1\right\rangle = 0\right].
\end{aligned}$$

That last line follows from the fact that for vectors a_1, b_1, a_2, b_2, we always have $\langle a_1, b_1\rangle \cdot \langle a_2, b_2\rangle = \langle a_1 \otimes a_2, b_1 \otimes b_2\rangle$. Finally, the above can be reduced to an $\mathbb{F}_q$-#$\textsf{OV}$ instance with $2^{n_\pi/2}$ vectors of $R(N)^{O(1)}$ dimensions. One can see that this polynomial blowup in the dimension is acceptable, and we can still proceed as in the case of $\mathbb{F}_2$. □

Now we are ready to prove Theorem 1.2 (restated below). Notice that here we use the stronger condition $\textsf{NQP} \not\subset \textsf{P}_{/\text{poly}}$ instead of $\textsf{NE} \not\subset \textsf{P}_{/\text{poly}}$.

Reminder of Theorem 1.2 *There is an absolute constant $\delta > 0$ such that, for all prime powers $q = p^r$ and all $\varepsilon > 0$ at least one of the following holds:*

- $\textsf{NQP} \not\subset \textsf{P}_{/\text{poly}}$.
- *There is a $\textsf{P}^{\textsf{NP}}$ machine M such that, for infinitely many N's, on input 1^N, M outputs an $N \times N$ matrix $H_N \in \{0,1\}^{N \times N}$ such that $\mathscr{R}_{H_N}(2^{(\log N)^{1-\varepsilon}}) \geq \delta \cdot N^2$ over $\mathbb{F}_q$.*

Proof of Theorem 1.2. Let $\delta > 0$ be a constant to be chosen later.

Assume that $\textsf{NQP} \subset \textsf{P}_{/\text{poly}}$. By [MW18], this in particular implies that for a constant b to be specified later and $T(n) := 2^{\log^b n}$, all polynomial-time verifiers for unary languages in $\textsf{NTIME}[2^{\log^b n}]$ have $S(n) := n^k$-size witness circuits, for a constant $k = k(b)$.

Set $R(N) := 2^{(\log N)^{1-\varepsilon}}$. We will now apply Lemma 3.1 with R, S, T as above. Note that $n_\pi = \log T(n) + O(\log\log T(n)) + O(\log S(n)) = \log^b n + O(\log n)$ and $N = 2^{n_\pi/2} = 2^{\log^b n/2 + O(\log n)}$. We thus calculate that

$$\log T(n)/\log R(N) \geq \log^b n/\log^{b(1-\varepsilon)} n \geq \log^{b \cdot \varepsilon} n = \omega(\log\log T(n) + \log S(n)) = \omega(\log n),$$

if we set $b = 2/\varepsilon$.

Therefore, since Conditions (1) and (3) of Lemma 3.1 hold, we conclude that Condition (2) of Lemma 3.1 does not hold, and this completes the proof. □

4 Rigid Matrix Construction in P^{NP}

In this section, we prove Theorem 1.1 by using an additional bootstrapping argument.

For an integer $n \in \mathbb{N}$, we write $n_{[k]}$ to denote the smallest integer $m \geq n$ such that $m \equiv 2^k - 1 \pmod{2^{k+1}}$. Notice that $n_{[k]}$ satisfies $|n - n_{[k]}| \leq 2^{k+1}$. Moreover, for all integers $n, m, i, j \in \mathbb{N}$ with $i \neq j$, we have that $n_{[i]} \neq m_{[j]}$.

We first prove the following lemma, which gives an (unconditional) construction of a matrix which is rigid for a higher rank than the construction in Theorem 1.1, but with a slower construction time of $\mathsf{TIME}[n^{\log n}]^{\mathsf{NP}}$.

Lemma 4.1. *There is an absolute constant $\delta > 0$ such for all prime powers $q = p^r$ and all constants $\varepsilon > 0$:*

- *There is a $\mathsf{TIME}[n^{\log n}]^{\mathsf{NP}}$ machine M such that, for infinitely many N's, on input 1^N, M outputs an $N \times N$ matrix $H_N \in \{0,1\}^{N \times N}$ such that $\mathscr{R}_{H_N}(2^{(\log N)^{1/2-\varepsilon}}) \geq \delta \cdot N^2$ over $\mathbb{F}_q$.*

Proof. Let δ be a constant to be specified later. For simplicity, we only consider the finite field $\mathbb{F}_2$ in the following. It is not hard to see that our proof also works for all finite fields $\mathbb{F}_{p^r}$ with a straightforward modification.

Assume toward a contradiction that for all $\mathsf{TIME}[n^{\log n}]^{\mathsf{NP}}$ machines M, and for almost all input lengths N, the output matrix $H_N \in \{0,1\}^{N \times N}$ of M satisfies $\mathscr{R}_{H_N}(2^{(\log N)^{1/2-\varepsilon}}) < \delta \cdot N^2$. (By padding with zeros or only keeping the first N^2 output bits, we can always assume that M outputs exactly N^2 bits on inputs of length N.)

Notation. Throughout the proof, we will often identify a matrix from $\{0,1\}^{N \times N}$ with a string from $\{0,1\}^{N^2}$ (reading the matrix from top row to bottom row, and from leftmost column to rightmost column to construct the corresponding string).

Define the functions $\mathsf{rk}(N) := 2^{(\log N)^{1/2-\varepsilon}}$ and $\ell_{\mathsf{comp}}(N) := 2 \cdot \sqrt{N} \cdot \mathsf{rk}(\sqrt{N})$.

Set $\varepsilon_{\mathsf{enc}} = 0.01$, and $\ell_{\mathsf{enc}}(N) := N^{1+\varepsilon_{\mathsf{enc}}}$. Define the function $\ell_{\mathsf{PCP}}(N) := N \cdot \log^{C_{\mathsf{PCP}}} N$ for a constant C_{PCP} to be specified later.

Applying Lemma 2.6, we fix a locally-decodable error correcting code $\mathsf{ECC}_{\mathsf{local}}$ with a $\mathrm{poly}(N)$-time encoder $\mathsf{Enc} : \{0,1\}^N \to \{0,1\}^{\ell_{\mathsf{enc}}(N)}$, which has a $(\log N)^{C_{\mathsf{enc}}}$-time randomized decoder that decodes any position with probability at least 0.99 when given oracle access to a codeword which is corrupted in less than a 0.01 fraction of its entries.

The Compression Scheme $f_i(\cdot)$. Now, given a string $S \in \{0,1\}^N$, we define the function $\mathsf{comp}(S)$ as follows. Let N' be the smallest square number $\geq N$ and let $A, B \in \{0,1\}^{\sqrt{N'} \times \mathsf{rk}(\sqrt{N'})}$ be the two matrices such that $S \circ 0^{N'-N}$ (interpreted as a $\{0,1\}^{\sqrt{N'} \times \sqrt{N'}}$ matrix) agrees with AB^T (over $\mathbb{F}_2$) on the greatest number of positions. In case of a tie, make the choice resulting in $A \circ B$ being the lexicographically earliest string. We define $\mathsf{comp}(S) := A \circ B$.

Given a string $S \in \{0,1\}^N$, we further define the sequence of functions $f_1(S) := \mathsf{Enc}(S)$ and $f_i(S) := \mathsf{Enc}(\mathsf{comp}(f_{i-1}(S)))$ for $i > 1$.

We now aim to apply Lemma 3.1 from the previous section. Fix a unary language $L \in \mathsf{NTIME}[2^n]$ such that $L \notin \mathsf{NTIME}[2^n/n]$ [Zák83]. Fix an efficient PCP verifier V for L from [BV14], such that $V(1^n)$ takes $\log \ell_{\mathsf{PCP}}(2^n)$ random bits and oracle access to a string of length $\ell_{\mathsf{PCP}}(2^n)$. In order to apply Lemma 3.1, we need to show $V(1^n)$ has small witness circuits.

The Construction of the $\mathsf{TIME}[n^{\log n}]^{\mathsf{NP}}$ Machine M_{comp}: A Bootstrapping Argument. Let $O_n \in \{0,1\}^{\ell_{\mathsf{PCP}}(2^n)}$ be the lexicographically first string which makes $V(1^n)$ always accept, if such a string exists, and $0^{\ell_{\mathsf{PCP}}(2^n)}$ otherwise.

Our $\mathsf{TIME}[n^{\log n}]^{\mathsf{NP}}$ machine M_{comp} works as follows. For n and $1 \le i \le 2/3 \cdot \log n$, let $\ell_{n,i} := \left\lceil \sqrt{|f_i(O_n)|} \right\rceil_{[i]}$. If $\ell_{n,i} \ge 2^{n^{1/2+\varepsilon_1}}$ for a constant ε_1 to be specified later, then M_{comp} on input $1^{\ell_{n,i}}$ computes $f_i(O_n)$, padded with $\ell_{n,i}^2 - |f_i(O_n)|$ zeros. Otherwise it outputs an all-zero matrix.

We first claim that M_{comp} is well-defined, meaning there exists a constant N_0 such that for all $n \ge N_0$ and $1 \le i \le 2/3 \log n$, the $\ell_{n,i}$'s are distinct. To prove this, it suffices to show that $\ell_{n,i} < \ell_{m,i}$ whenever $i \le 2/3 \log n$ and $n < m$, but this follows from the definitions of the function $f_i(\cdot)$'s.

Next, we note that M_{comp} indeed runs in $\mathsf{TIME}[n^{\log n}]^{\mathsf{NP}}$: on input $1^{\ell_{n,i}}$ of length $m = \ell_{n,i} \ge 2^{n^{1/2+\varepsilon_1}}$, the algorithm runs in time $\operatorname{poly}(\ell_{n,1}) = 2^{O(n)} \le m^{\log m}$.

$V(1^n)$ Has Succinct Witness. We first show from our assumption (that $\mathsf{TIME}[n^{\log n}]^{\mathsf{NP}}$ does not have rigid matrices) that $V(1^n)$ has a succinct witness circuit if there is an oracle which always satisfies it.

When this is the case, notice that for all $1 \le i \le 2/3 \log n$, the output of $M_{\mathsf{comp}}(1^{\ell_{n,i}})$ can be δ-approximated by a matrix of rank $\mathsf{rk}(\ell_{n,i})$. We can calculate that $\ell_{n,2/3 \log n} < 2^{n^{1/2}}$; let j be the largest integer such that $\ell_{n,j} \ge 2^{n^{1/2+\varepsilon_1}}$, and note that $\ell_{n,j} \le 2^{3n^{1/2+\varepsilon_1}}$. Hence, $M_{\mathsf{comp}}(1^{\ell_{n,j}})$ can be implemented as a circuit of size $2^{O(n^{1/2+\varepsilon_1})}$.

Next, if there is a size-S circuit which $(1-\delta)$-approximates $M_{\mathsf{comp}}(1^{\ell_{n,i}})$, then $M_{\mathsf{comp}}(1^{\ell_{n,i-1}})$ can be $(1-\delta)$-approximated by a $\mathsf{rk}(\ell_{n,i-1}) \cdot \operatorname{poly}(n) \cdot S$ size circuit by using the local decoder of the corresponding locally decodeable codes. Therefore, $M_{\mathsf{comp}}(1^{\ell_{n,1}})$ can be $(1-\delta)$-approximated by a circuit of size

$$\prod_{i=1}^{j-1} \mathsf{rk}(\ell_{n,i}) \cdot n^{O(\log n)} \cdot 2^{O(n^{1/2+\varepsilon_1})} = 2^{O(n^{1/2+\varepsilon_1})}.$$

Since $M_{\mathsf{comp}}(1^{\ell_{n,1}}) = \mathsf{Enc}(O_n)$, it follows that O_n can be computed *exactly* by a $2^{O(n^{1/2+\varepsilon_1})}$-size circuit.

Applying Lemma 3.1. Toward applying Lemma 3.1, we set $T(n) = 2^n$, $S(n) = 2^{O(n^{1/2+\varepsilon_1})}$ and $R(N) = 2^{(\log N)^{1/2-\varepsilon}}$, where $\varepsilon_1 := \varepsilon/2 > 0$. The two parameters in Condition (3) of Lemma 3.1 are bounded by $n_\pi = \log T(n) + O(\log\log T(n)) + O(\log S(n)) = n + O(n^{1/2+\varepsilon_1})$ and $N = 2^{n/2+O(n^{1/2+\varepsilon_1})}$. We thus calculate that

$$\log T(n)/\log R(N) = \Omega(n/n^{1/2-\varepsilon}) = \Omega(n^{1/2+\varepsilon}) = \omega(n^{1/2+\varepsilon_1}) = \omega(\log\log T(n) + \log S(n)).$$

Therefore, Conditions (1) and (3) of Lemma 3.1 are satisfied, and it follows that Condition (2) must be violated, which completes the proof. □

Finally, we prove Theorem 1.1 (restated below) by using a simple padding argument.

Reminder of Theorem 1.1 *There is an absolute constant $\delta > 0$ such for all prime powers $q = p^r$ and all constants $\varepsilon > 0$:*

- *There is a P^{NP} machine M such that, for infinitely many N's, on input 1^N, M outputs an $N \times N$ matrix $H_N \in \{0,1\}^{N\times N}$ such that $\mathscr{R}_{H_N}(2^{(\log N)^{1/4-\varepsilon}}) \ge \delta \cdot N^2$ over $\mathbb{F}_q$.*

Proof of Theorem 1.1. We have shown, from Lemma 4.1, that there is an absolute constant $\delta > 0$ such that for all constants $\varepsilon > 0$:

- There is a $\mathsf{TIME}[n^{\log n}]^{\mathsf{NP}}$ machine M such that, for infinitely many Ns, on input 1^N, M outputs an $N \times N$ matrix $H_N \in \{0,1\}^{N\times N}$ such that $\mathscr{R}_{H_N}(2^{(\log N)^{1/2-\varepsilon}}) \geq \delta \cdot N^2$ over $\mathbb{F}_2$.

Let $N' = N^{\log N}$, and consider the P^{NP} machine M' which, given an input $1^{N'}$, outputs a matrix $H'_{N'} := \mathbf{1}_{N^{\log N - 1}} \otimes H_N$. By Lemma 2.7, we have

$$\mathscr{R}_{H'_{N'}}(2^{(\log N)^{1/2-\varepsilon}}) \geq \delta \cdot N'^2$$

for infinitely many N'. This rigidity bound is equivalent to

$$\mathscr{R}_{H'_{N'}}(2^{(\log N')^{1/4-\varepsilon/2}}) \geq \delta \cdot N'^2,$$

as desired. □

5 $\mathsf{PH}^{\mathsf{cc}}$ Communication Lower Bound for $\mathsf{TIME}\left[2^{(\log n)^{\omega(1)}}\right]^{\mathsf{NP}}$

In this section we apply our construction of rigid matrices to prove a $\mathsf{PH}^{\mathsf{cc}}$ communication lower bound for functions in $\mathsf{TIME}\left[2^{(\log n)^{\omega(1)}}\right]^{\mathsf{NP}}$. Our main tool will be a known connection between rigid matrices and $\mathsf{PH}^{\mathsf{cc}}$:

Lemma 5.1 ([Raz89], see also [Wun12])**.** *Letting f be a function in $\mathsf{PH}^{\mathsf{cc}}$, the $2^n \times 2^n$ communication matrix M_f of f has $\mathscr{R}_{M_f}(2^{(\log n/\varepsilon)^c}) \leq \varepsilon \cdot 4^n$ over $\mathbb{F}_2$, where $\varepsilon > 0$ is arbitrary and $c > 0$ is a constant depending only on f, but not n.*

We will also use the following simple Lemma.

Lemma 5.2. *For any field $\mathbb{F}$ and any matrix $A \in \mathbb{F}^{N\times N}$, and for $M > N$, define $P_{A,M} \in \mathbb{F}^{M\times M}$ to be the matrix such that the top-left $N \times N$ sub-matrix is A, and the rest of entries are all zeros. For all r, we have*

$$\mathscr{R}_{P_{A,M}}(r) \geq \mathscr{R}_A(r).$$

Reminder of Theorem 1.4 *For all functions $\alpha(n) = \omega(1)$ such that $n^{\alpha(n)}$ is time-constructible, there is a function $f \in \mathsf{TIME}[2^{(\log n)^{\alpha(n)}}]^{\mathsf{NP}}$ which is not in $\mathsf{PH}^{\mathsf{cc}}$.*

Proof. By Theorem 1.1, we know that there is a P^{NP} machine M such that $\mathscr{R}_{M(1^N)}(2^{(\log N)^{1/5}}) \geq \delta \cdot N^2$ over $\mathbb{F}_2$, for a constant $\delta > 0$ and infinitely many N's. For simplicity, we can assume $\alpha(n) \leq \log n$ (e.g., by setting $\alpha'(n) = \min(\alpha(n), \log n)$).

The Definition of f. Now we define a function $f \in \mathsf{TIME}[2^{(\log n)^{\alpha(n)}}]^{\mathsf{NP}}$ as follows:

- Given as input $x \in \{0,1\}^n$, the function f outputs zero immediately if 4 does not divide n. Otherwise let $m = n/4$.
- It treats the first $2m$ bits of the input as an integer N in $[2^{2m}]$, and if $N > 2^{(\log m)^{\alpha(n)}}$, it outputs zero.
- Otherwise, it constructs the matrix $H = M(1^N)$. Let $S = 2^m$, and $Q = P_{\mathbf{1}_{\lfloor S/N \rfloor}\otimes H, S}$. It treats the next $2m$ bits of the input as a pair of integers $(i,j) \in [S] \times [S]$, and outputs $Q_{i,j}$.

$Q_{i,j}$ can be computed easily given H, so f can be computed in $\mathsf{TIME}[2^{(\log n)^{\alpha(n)}}]^{\mathsf{NP}}$.

f **is not in in $\mathsf{PH}^{\mathsf{cc}}$.** We will now show that f, when interpreted as a communication problem, is not in $\mathsf{PH}^{\mathsf{cc}}$. We distribute the input bits of f among the two players as follows: When 4 divides n, setting $m = n/4$, then Alice holds the bits $x_1, x_2, \ldots, x_m$ and $x_{2m+1,\ldots,3m}$, and Bob holds the bits $x_{m+1}, x_{m+2}, \ldots, x_{2m}$ and $x_{3m+1,3m+2,\ldots,4m}$.

Assume to the contrary that $f \in \mathsf{PH}^{\mathsf{cc}}$. This means that for all assignments α to $x_1, x_2, \ldots, x_{2m}$, the restricted function $f_\alpha : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ is still in $\mathsf{PH}^{\mathsf{cc}}$. That is, there exists a constant c, such that for all $N \le 2^{(\log m)^{\alpha(n)}}$, $S = 2^m$, and $Q = P_{\mathbf{1}_{\lfloor S/N \rfloor} \otimes M(1^N),S}$, we have

$$\mathscr{R}_Q(2^{(\log m)^c}) \le \delta/2 \cdot S^2.$$

By Lemma 5.2, this implies

$$\mathscr{R}_{\mathbf{1}_{\lfloor S/N \rfloor} \otimes M(1^N)}(2^{(\log m)^c}) \le \delta/2 \cdot S^2 \le \delta \cdot 2/3 \cdot (\lfloor S/N \rfloor \cdot N)^2.$$

By Lemma 2.7, this further implies

$$\mathscr{R}_{M(1^N)}(2^{(\log m)^c}) \le 2/3 \cdot \delta \cdot N^2.$$

Now, let N be a sufficiently large integer such that

$$\mathscr{R}_{M(1^N)}(2^{(\log N)^{1/5}}) \ge \delta \cdot N^2.$$

Let m be the smallest integer such that $2^{(\log m)^{\alpha(4m)}} \ge N$. Since $\alpha(n)$ is unbounded, we can pick N to be large enough such that $\alpha(4m-4) \ge 20 \cdot c$. By definition of m, we have $2^{(\log(m-1))^{\alpha(4m-4)}} < N$, meaning $2^{(\log(m-1))^{20c}} < N$, and so $2^{(\log m)^{10c}} < N$. But then by the above discussion, we have

$$\mathscr{R}_{M(1^N)}(2^{(\log N)^{1/10}}) \le 2/3 \cdot \delta \cdot N^2,$$

a contradiction. □

6 Depth-2 Arithmetic Circuit Lower Bounds

In this section we prove Theorem 1.6 (restated below). Recall first the definition of w_2:

Definition 6.1. For a field $\mathbb{F}$ and a matrix $A \in \mathbb{F}^{N \times N}$, let

$$w_2(A) := \min\{\mathrm{nnz}(B) + \mathrm{nnz}(C) \mid A = BC\},$$

where the min is over all pairs B, C of matrices of any dimensions over $\mathbb{F}$ whose product is A, and $\mathrm{nnz}(X)$ denotes the number of nonzero entries in the matrix X.

Reminder of Theorem 1.6 *For all prime powers $q = p^r$ and constants $\varepsilon > 0$, it holds:*

- *There is a P^{NP} machine M such that, for infinitely many N, on input 1^N, M outputs an $N \times N$ matrix $H_N \in \{0,1\}^{N \times N}$ such that $w_2(H_N) \ge \Omega(N \cdot 2^{(\log N)^{1/4-\varepsilon}})$ over $\mathbb{F}_q$.*

We first prove the following folklore lemma.

Lemma 6.2. *For any field $\mathbb{F}$, and any matrix $A \in \mathbb{F}^{N\times N}$, let $r = w_2(A)/N$. Then, for any constant $\delta > 0$, we have*

$$\mathscr{R}_A(\rho_\delta \cdot r^2) \leq \delta \cdot N^2,$$

for some constant ρ_δ depending only on δ.

In other words, if $\mathscr{R}_A(\rho_\delta \cdot r^2) > \delta \cdot N^2$, then we have $w_2(A) \geq r \cdot N$.

Proof. For some integer M, let B and C be matrices over $\mathbb{F}$ of dimensions $N \times M$ and $M \times N$, respectively, such that $A = BC$ and $\operatorname{nnz}(B) + \operatorname{nnz}(C) = r \cdot N$. For $i, j \in [N]$, let $b_i \in \mathbb{F}^M$ be the i-th row of B, and $c_j \in F^M$ be the j-th column of C. Hence, $A_{i,j} = \langle b_i, c_j\rangle$.

Now, let ρ_δ be a function of δ to be specified later, and set $m = \rho_\delta \cdot r^2$. Pick a hash function $P : [M] \to [m]$ uniformly at random. Next, for each b_i, we define a vector $\widetilde{b}_i$ by setting, for each $j \in [m]$:

$$(\widetilde{b}_i)_j := \sum_{k \in P^{-1}(j)} (b_i)_k.$$

We similarly define $\widetilde{c}_j$. Now, let $\widetilde{B}$ be the $N \times m$ matrix with the $\widetilde{b}_i$'s as rows, and $\widetilde{C}$ be the $m \times N$ matrix with the $\widetilde{c}_j$'s as columns. We will now argue that $\widetilde{B}\widetilde{C}$ approximates A well.

First, from definition, we have

$$\mathop{\mathbb{E}}_{(i,j)\in[N]\times[N]} \operatorname{nnz}(b_i) + \operatorname{nnz}(c_j) = \frac{\operatorname{nnz}(A) + \operatorname{nnz}(B)}{N} = r.$$

Hence, by Markov's inequality, for at least a $1 - \delta/2$ fraction of the pairs $(i,j) \in [N] \times [N]$, we have $\operatorname{nnz}(b_i) + \operatorname{nnz}(c_j) \leq r \cdot \frac{2}{\delta}$.

Fix such a pair of (i, j), and let $I = \{k \in [M] : (b_i)_k \neq 0 \vee (c_j)_k \neq 0\}$, which has size $|I| \leq r \cdot \frac{2}{\delta}$. Note that if all the elements of I have distinct images under the mapping P, then $\langle \widetilde{b}_i, \widetilde{c}_j\rangle = \langle b_i, c_j\rangle = M_{i,j}$. By a union bound, this happens with probability at least $1 - |I|^2/m$ over the random choice of P.

Setting $\rho_\delta = (\frac{2}{\delta})^3$, we have $1 - |I|^2/m \geq 1 - \delta/2$. Thus, by the probabilistic method, there is a fixed P for which $\widetilde{B}\widetilde{C}$ agrees with A on a $1 - \delta$ fraction of inputs, and hence $\mathscr{R}_A(\rho_\delta \cdot r^2) \leq \delta \cdot N^2$. □

Theorem 1.6 then follows by combining Lemma 6.2, Theorem 1.1, and Theorem 1.2.

7 Threshold Circuit Lower Bound for E^{NP}

Definition 7.1. For a Boolean function $f : \{0,1\}^n \to \{0,1\}$, its *truth-table matrix* is the matrix $M_f \in \mathbb{F}_2^{2^{n/2}\times 2^{n/2}}$ which is given by $M_f[x,y] = f(x,y)$ for all $x, y \in \{0,1\}^{n/2}$.

In this section, we prove threshold circuit lower bounds for E^{NP}.

Reminder of Theorem 1.7 *For every $\delta > 0$ and prime p, there is an $a > 0$ such that the class E^{NP} does not have non-uniform $\mathsf{AC}^0[p] \circ \mathsf{LTF} \circ \mathsf{AC}^0[p] \circ \mathsf{LTF}$ circuits of depth $o(\log n/\log\log n)$ where the bottom LTF layer has $2^{O(n^a)}$ gates, the rest of the circuit has polynomial size, and the middle layer LTF gates have fan-in $O(n^{1/2-\delta})$.*

We prove Theorem 1.7 by combining our main results, rigidity lower bounds for matrices which can be computed in P^{NP} (and hence which are the truth tables of functions in E^{NP}), with a rigidity upper bound for such threshold circuits:

Lemma 7.2. *For any prime p and constants $c, \delta, a > 0$ such that $c > \delta$ and $\delta^2 > 2ca$, there is a constant $\gamma > 0$ such that every $\mathsf{AC}^0[p] \circ \mathsf{LTF} \circ \mathsf{AC}^0[p] \circ \mathsf{LTF}$ circuit C on n inputs, where the bottom layer has $2^{O(n^a)}$ LTF gates, the middle layer LTF gates have fan-in $O(n^{c-\delta})$, and the rest of the circuit has polynomial size and depth $o(\log n / \log\log n)$, has $\mathscr{R}_{M_C}(2^{n^{c/2-\gamma}}) \le \varepsilon 2^n$ for all $\varepsilon \ge 1/2^{n^{o(1)}}$ over $\mathbb{F}_p$.*

Lemma 7.2 is a variant on [AW17, Theorem C.1], and its proof is very similar. We nonetheless give the proof here as some important details are different. We begin with some definitions.

Definition 7.3. A *ε-error probabilistic polynomial* for a Boolean function $f : \{0,1\}^n \to \{0,1\}$ over a field $\mathbb{F}$ is a distribution Q on polynomials $q : \mathbb{F}^n \to \mathbb{F}$ such that, for all $x \in \{0,1\}^n$, we have $\Pr_{q \sim Q}[q(x) = f(x)] \ge 1 - \varepsilon$. The *degree* of Q is the maximum degree of the polynomials in its support. The *ε-error probabilistic degree* of Boolean function f over $\mathbb{F}$ is the minimum degree of an ε-error probabilistic polynomial for f over $\mathbb{F}$.

Definition 7.4. The ε-probabilistic rank of a Boolean function $f : \{0,1\}^{2n} \to \{0,1\}$ is the minimum r such that there is a distribution D on matrices in $\mathbb{F}^{2^n \times 2^n}$ of rank at most r such that for all $x, y \in \{0,1\}^n$ we have $\Pr_{M \sim D}[M[x,y] = f(x,y)] \ge 1 - \varepsilon$.

Rigidity, probabilistic rank, and probabilistic degree have a simple known relationship:

Proposition 7.5. *For a Boolean function $f : \{0,1\}^{2n} \to \{0,1\}$:*

- *If the ε-probabilistic degree of f is d, then the ε-probabilistic rank of f is at most $n^{O(d)}$.*
- *If the ε-probabilistic rank of f is r, then $\mathscr{R}_{M_f}(r) \le \varepsilon 2^{2n}$.*

Our proof will make use of a number of probabilistic polynomial and probabilistic rank constructions from past work:

Lemma 7.6 ([KS12] Lemma 10)**.** *For any prime p and $\varepsilon \in (0, 1/2)$, any $\mathsf{AC}^0[p]$ circuit C of size s and depth d has an ε-error probabilistic polynomial over $\mathbb{F}_p$ of degree at most $(\log s)^{O(d)} \cdot \log(1/\varepsilon)$.*

Theorem 7.7 ([AW15] Theorem 1.1)**.** *Every symmetric Boolean function on n variables has ε-probabilistic degree $O(\sqrt{n \log(1/\varepsilon)})$.*

Lemma 7.8 ([AW17] Theorem D.3)**.** *For every n, every linear threshold function on n inputs has ε-probabilistic rank $O(n^2/\varepsilon)$.*

Theorem 7.9 ([MT98] Theorem 3.3, [ACW16] Theorem 7.1)**.** *For every $\alpha > 0$, every LTF on n inputs can be computed by a polynomial-size $\mathsf{AC}^0 \circ \mathsf{MAJ}$ circuit where the fan-in of each MAJ gate is $n^{1+\alpha}$ and the circuit has depth $O(\log(1/\alpha))$.*

Proof of Lemma 7.2. Let $b \le 2^{O(n^a)}$ be the number of LTF gates in the bottom layer of C. We know by Lemma 7.8 that each of these b LTF gates has $\varepsilon/(2b)$-probabilistic rank $O(n^2 b/\varepsilon)$. We will next show that there is an $\varepsilon/2$-error probabilistic polynomial of degree $O(n^{c/2-a-\gamma'})$ for computing all the layers of C above the bottom LTF layer, for some $\gamma' > 0$. We can then view the terms of the probabilistic rank expressions for the bottom layer LTF gates as 'variables' that we substitute into this polynomial. Since there are b such gates, each with rank $O(n^2 b/\varepsilon)$, the resulting probabilistic rank expression for C has rank $O(n^2 b^2/\varepsilon)^{O(n^{c/2-a-\gamma})} \le 2^{O(n^{c/2-\gamma})}$ for any $\gamma < \gamma'$. (Here we used that $\varepsilon \ge 2^{n^{o(1)}}$.) By a union bound, the resulting error is at most ε.

It remains to bound the $\varepsilon/2$-probabilistic degree of a $\mathsf{AC}^0[\oplus] \circ \mathsf{LTF} \circ \mathsf{AC}^0[\oplus]$ circuit on n inputs of polynomial size and depth $d \leq o(\log n/\log\log n)$, where the LTF gates have fan-in $O(n^{c-\delta})$. First, applying Theorem 7.9 with $\alpha = \delta/c$ to the LTF gates, we reduce the circuit to a $\mathsf{AC}^0[\oplus] \circ \mathsf{MAJ} \circ \mathsf{AC}^0[\oplus]$ of size $s = n^{O(1)}$ and depth $O(d)$, where each MAJ gate has fan-in $O(n^{(c-\delta)(1+\delta/c)}) = O(n^{c-\delta^2/c})$. We now use Theorem 7.7 to replace each MAJ gate with a $\varepsilon/(4s)$-error probabilistic polynomial of degree $O(\sqrt{n^{c-\delta^2/c}\log(s/\varepsilon)}) \leq n^{c/2-\delta^2/(2c)+o(1)}$, and we use Lemma 7.6 to replace the $\mathsf{AC}^0[\oplus]$ circuitry with a $\varepsilon/4$-error probabilistic polynomial of degree $n^{o(1)}\log^{O(d)}(n) \leq n^{o(1)}$. Combined, by a union bound, the entire circuit has a $\varepsilon/2$-error probabilistic polynomial of degree $n^{c/2-\delta^2/(2c)+o(1)}$. This is of the desired form $O(n^{c/2-a-\gamma'})$ for any $\gamma' < \delta^2/(2c) - a$. Since we assumed that $\delta^2 > 2ac$, there are $\gamma' > 0$ satisfying this inequality, as desired. □

Proof of Theorem 1.7. Consider first this circuit class where the middle layer LTF gates have fan-in $O(n^{1/2-\delta})$. By Lemma 7.2 with $c = 1/2$, for every $\delta > 0$, there are $a, \gamma > 0$ such that every circuit C in this class has $\mathscr{R}_{M_C}(2^{n^{1/4-\gamma}}) \leq o(2^n)$ over $\mathbb{F}_p$. By comparison, Theorem 1.1 says that there is a E^{NP} machine H such that for all $\gamma > 0$, we have $\mathscr{R}_{M_H}(2^{n^{1/4-\gamma}}) \geq \Omega(2^n)$ over $\mathbb{F}_p$.

□

In fact, Theorem 1.8 follows from an almost identical argument, by noting that in the proof of Lemma 7.2, the only property of the upper $\mathsf{AC}^0[p] \circ \mathsf{LTF} \circ \mathsf{AC}^0[p]$ circuitry we used is that it has a ε-error probabilistic polynomial of degree $n^{c/2-\Omega(1)}$ for all constant $\varepsilon > 0$.

8 Acknowledgment

We thank our advisor, Ryan Williams, for his support and many inspiring discussions throughout this project. We are also grateful to Alexander Golovnev and Chi-Ning Chou for helpful discussions, and to Emanuele Viola and Avishay Tal for helpful comments on an earlier draft.

A Algorithm for Counting Orthogonal Vectors over Finite Fields

Finally, in this Section, we give a sketch of the algorithm for $\mathbb{F}_{p^r}$-#OV which we stated in Theorem 2.9 and which is needed by our construction above. The algorithm is a minor modification of the deterministic algorithm for #OV by Chan and Williams [CW16], which makes use of the 'polynomial method in algorithm design.'

A.1 Reduction to Prime Fields

We begin by sketching a reduction from $\mathbb{F}_{p^r}$-#OV to $\mathbb{F}_p$-#OV. More precisely, for a prime power $q = p^r$, we give a reduction from one instance of $\mathbb{F}_q$-#$\mathsf{OV}_{n,d}$ to a constant number of different instances of $\mathbb{F}_p$-#$\mathsf{OV}_{n,d\cdot O_r(1)}$. The reduction builds on ideas from [LPT$^+$17] and [Wil18a].

We first define an intermediate problem $\mathbb{F}_p$-#AND-$\mathsf{OV}_{n,d,r}$: given as input two size-n collections $A, B \subseteq (\mathbb{F}_q^d)^r$, with $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ (so, for instance, each a_i is an r-tuple of vectors from $\mathbb{F}_q^d$), the goal is to compute the number of pairs $(i, i') \in [n]^2$ such that $\langle a_{i,j}, b_{i',j}\rangle = 0$ for all $j \in [r]$.

$\mathbb{F}_q$-#OV $\Rightarrow$ $\mathbb{F}_p$-#AND-OV. We first show how to reduce an $\mathbb{F}_q$-#OV$_{n,d}$ instance to an $\mathbb{F}_p$-#AND-OV$_{n,dr^2,r}$ instance in nearly linear time. Pick a degree-r $\mathbb{F}_p$ irreducible polynomial P; we know that $\mathbb{F}_q$ isomorphic to $F_p[X]/(P)$. In the calculations below, we perform the arithmetic mod P.

Suppose we have two vectors $u, v \in \mathbb{F}_q^d$. Let $u_i = \sum_{j=0}^{r-1} \alpha_{i,j} \cdot X^j$, and $v_i = \sum_{j=0}^{r-1} \beta_{i,j} \cdot X^j$ for coefficients $\alpha_{i,j}, \beta_{i,j} \in \mathbb{F}_p$. We have that

$$\begin{aligned}\sum_{i=1}^{d} u_i \cdot v_i &= \sum_{i=1}^{d} \left(\sum_{j=0}^{r-1} \alpha_{i,j} \cdot X^j \right) \cdot \left(\sum_{j=0}^{r-1} \beta_{i,j} \cdot X^j \right) \\ &= \sum_{i=1}^{d} \sum_{j=0}^{r-1} \sum_{k=0}^{r-1} \alpha_{i,j} \cdot \beta_{i,k} X^{j+k}.\end{aligned}$$

Define the coefficients $\gamma_{j+k,\ell} \in \mathbb{F}_p$ so that $X^{j+k} = \sum_{\ell=0}^{r-1} \gamma_{j+k,\ell} \cdot X^\ell \pmod{P}$. The above simplifies to

$$\sum_{j=0}^{r-1} \sum_{k=0}^{r-1} X^{j+k} \cdot \sum_{i=1}^{d} \alpha_{i,j} \cdot \beta_{i,k} = \sum_{\ell=0}^{r-1} X^\ell \cdot \left(\sum_{j=0}^{r-1} \sum_{k=0}^{r-1} \sum_{i=1}^{d} \gamma_{j+k,\ell} \cdot \alpha_{i,j} \cdot \beta_{i,k} \right) \pmod{P}.$$

We therefore see that $\langle u_i, v_i \rangle = 0$ if and only if

$$\sum_{j=0}^{r-1} \sum_{k=0}^{r-1} \sum_{i=1}^{d} \gamma_{j+k,\ell} \cdot \alpha_{i,j} \cdot \beta_{i,k} = 0 \tag{1}$$

for all $0 \le \ell \le r-1$. For each ℓ, we can build vectors $u_i^{(\ell)}$ and $v_i^{(\ell)}$ in $\mathbb{F}_p^{r^2 \cdot d}$ so that $\langle u_i^{(\ell)}, v_i^{(\ell)} \rangle$ equals the left hand side of (1). This transformation reduces an $\mathbb{F}_q$-#OV$_{n,d}$ instance to an $\mathbb{F}_p$-#AND-OV$_{n,dr^2,r}$ instance as desired.

$\mathbb{F}_p$-#AND-OV $\Rightarrow$ $\mathbb{F}_p$-#OV. Now, given an $\mathbb{F}_p$-#AND-OV$_{n,d,r}$ instance with input collections A, B, we show how to reduce it to p^r different $\mathbb{F}_p$-#OV$_{n,dr+1}$ instances, again in nearly linear time.

Let $a, b \in (\mathbb{F}_p^d)^r$. For a random vector $u \in \mathbb{F}_p^r$, observe that:

- If $\langle a_i, b_i \rangle = 0$ for all $i \in [r]$, then $\sum_{i=1}^{r} u_i \cdot \langle a_i, b_i \rangle$ is always zero.
- Otherwise, $\sum_{i=1}^{r} u_i \cdot \langle a_i, b_i \rangle = 1$ with probability $1/p$.

For our reduction, we iterate over all vectors $u \in \mathbb{F}_p^r$, and sum the number of pairs $(a, b) \in A \times B$ such that

$$\sum_{i=1}^{r} u_i \cdot \langle a_i, b_i \rangle = \left\langle \bigcirc_{i=1}^{r} u_i a_i, \bigcirc_{i=1}^{r} b_i \right\rangle = 1.$$

For each u, this can be written as an $\mathbb{F}_p$-#OV$_{n,dr+1}$ instance (via $\langle a, b \rangle = 1 \Leftrightarrow \langle a \circ 1, b \circ -1 \rangle = 0$).

For a pair $(a, b) \in A \times B$, if $\langle a_i, b_i \rangle = 0$ for all $i \in [r]$, then (a, b) is never counted in the above sum. Otherwise, it is counted p^{r-1} times. Therefore, by summing up the results of all these $\mathbb{F}_p$-#OV instances after the reduction, dividing the result by p^{r-1}, and then finally subtracting the resulting number from $|A| \cdot |B|$, we can compute the answer to the given $\mathbb{F}_{n,d,r}$-#AND-OV instance.

A.2 Algorithm for Prime Fields

In this subsection, we give a self-contained exposition of the $\mathbb{F}_p$-#OV algorithm which is implicit in [CW16]. In [CW16], the deterministic $\#\mathsf{OV}$ algorithm works by combining two key technical tools: *small-biased sets*, and *modulus-amplifying polynomials*. We won't need small-biased sets here as we only aim to solve $\mathbb{F}_{p^r}$-#OV. We first recall the definition of modulus-amplifying polynomials.

Lemma A.1 (Modulus-Amplifying Polynomial [Yao90, BT94]). *For all integers $\ell \geq 1$, there is a polynomial F_ℓ over $\mathbb{Z}$ of degree $(2\ell - 1)$ with $O(\ell)$-bit coefficients such that for all integers $m \geq 1$ and all $a \in \mathbb{Z}$:*

(1) if $a \equiv 0 \pmod{m}$, then $F_\ell(a) \equiv 0 \pmod{m^\ell}$, and

(2) if $a \equiv 1 \pmod{m}$, $F_\ell(a) \equiv 1 \pmod{m^\ell}$.

We also need the following algorithm for fast rectangular matrix multiplication.

Theorem A.2 ([Cop82]; see also [Wil14a]). *There is an algorithm for multiplying matrices of dimensions $N \times N^{0.172}$ and $N^{0.172} \times N$ over any field using $N^2 \cdot \operatorname{polylog}(N)$ field operations.*

Now we are ready to prove Theorem 2.9 when the modulus q is a prime. The case when q is a prime power then follows using the reduction from Subsection A.1.

Theorem A.3. *For all primes p, there is an $n^{2-\Omega(1/\log(d/\log n))}$ time deterministic algorithm for $\mathbb{F}_p$-#$\mathsf{OV}_{n,d}$, when $d = n^{o(1)}$.*

Proof. Let ℓ be a parameter to be specified later. Let X, Y be two collections of $p^{\ell/4}$ vectors from $\mathbb{F}_p^d$. We define the polynomial

$$P(X,Y) := \sum_{(x,y)\in X\times Y} (1 - F_\ell(\langle x,y\rangle^{p-1})),$$

where F_ℓ is the modulus-amplifying polynomial from Lemma A.1. Hence,

$$1 - F_\ell(\langle x,y\rangle^{p-1}) \equiv \begin{cases} 1 \pmod{p^\ell} & \text{when } \langle x,y\rangle \equiv 0 \pmod{p}, \\ 0 \pmod{p^\ell} & \text{when } \langle x,y\rangle \not\equiv 0 \pmod{p}. \end{cases}$$

Let us count the number M of monomials in $F_\ell(\langle x,y\rangle^{p-1}) = F_\ell((x_1y_1 + x_2y_2 + \cdots + x_dy_d)^{p-1})$ when it is expanded and simplified. F_ℓ is a polynomial of degree $(2\ell - 1)\cdot(p-1)$ in $x, y \in \mathbb{F}_p^d$. In particular, since we are working over $\mathbb{F}_p$, we may simplify F_ℓ so that each of the $2d$ input variables has individual degree at most $p - 1$ in any given monomial. Thus, using the simple bound that no monomial depends on more variables than the degree of the polynomial, combined with the fact that the power of x_i in a given monomial is always equal to the power of y_i in that monomial, we get the bound

$$M \leq (p-1)^{2\ell\cdot p} \cdot \sum_{i=0}^{2\ell\cdot p} \binom{d}{i} \leq (p-1)^{2\ell\cdot p} \cdot O\left(\frac{d}{\ell\cdot p}\right)^{2\ell\cdot p} \leq O\left(\frac{d}{\ell}\right)^{2\ell\cdot p}.$$

Next, we will construct two mappings $\Phi_X, \Phi_Y : (\mathbb{F}_p^d)^{p^{\ell/4}} \to \mathbb{Z}^M$ such that for any $X, Y \in (\mathbb{F}_p^d)^{p^{\ell/4}}$,

$$P(X,Y) = \langle \Phi_X(X), \Phi_Y(Y)\rangle.$$

We construct Φ_X, Φ_Y as follows. For a set $S \subseteq [d]$, let x_S (resp. y_S) denote $\prod_{i \in S} x_i$ ($\prod_{i \in S} y_i$). Let $S_1, S_2, \dots, S_M$ be an enumeration of all subsets of $[d]$ of size no greater than $(2\ell - 1) \cdot (p - 1)$. There are corresponding coefficients $c_1, c_2, \dots, c_M \in \mathbb{Z}$ such that

$$1 - F_\ell(\langle x, y \rangle^{p-1}) = \sum_{i=1}^{M} c_i \cdot x_{S_i} \cdot y_{S_i}.$$

We can then define

$$\Phi_X(X) := \left(\sum_{x \in X} c_1 \cdot x_{S_1}, \sum_{x \in X} c_2 \cdot x_{S_2}, \dots, \sum_{x \in X} c_M \cdot x_{S_M} \right),$$

$$\Phi_Y(Y) := \left(\sum_{y \in Y} y_{S_1}, \sum_{y \in Y} y_{S_2}, \dots, \sum_{y \in Y} y_{S_M} \right),$$

and it follows that

$$\langle \Phi_X(X), \Phi_Y(Y) \rangle = \sum_{i=1}^{M} \sum_{(x,y) \in X \times Y} c_i \cdot x_{S_i} \cdot y_{S_i} = P(X, Y).$$

Picking $c = d/\log n$ and $\ell = \varepsilon/p \cdot \log n/\log c$ for a small enough constant ε, we have

$$M \le O\left(\frac{c \log n}{\ell}\right)^{2\ell \cdot p} = O\left(\frac{p \cdot c \log c}{\varepsilon}\right)^{2\varepsilon \log n/\log c} \le n^{0.01}.$$

Let $b = p^{\ell/4}$ (and set ε small enough so that $b \le n^{0.01}$ as well). We partition A (resp. B) into n/b blocks $A_1, A_2, \dots, A_{n/b}$ ($B_1, B_2, \dots, B_{n/b}$), each of size b. We then apply the algorithm from Theorem A.2 to evaluate $P(A_i, B_j)$ for each $(i, j) \in [n/b] \times [n/b]$ in $(n/b)^2 \cdot \mathrm{polylog}(n) = n^{2-1/O(\log c)}$ time by multiplying two matrices of dimensions $n/b \times n^{0.01}$ and $n^{0.01} \times n/b$ over $\mathbb{Z}$ whose entries are $\mathrm{polylog}(n)$-bit integers. Since

$$P(A_i, B_j) \equiv \sum_{(x,y) \in A_i \times B_j} [\langle x, y \rangle \equiv 0 \pmod{p}] \pmod{p^\ell},$$

this allows us to solve $\mathbb{F}_p$-#OV in $n^{2-1/O(\log c)}$ time. □

References

[ACW16] Josh Alman, Timothy M Chan, and Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In *FOCS*, pages 467–476. IEEE, 2016.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in P. *Annals of mathematics*, pages 781–793, 2004.

[AKW90] Noga Alon, Mauricio Karchmer, and Avi Wigderson. Linear circuits over gf(2). *SIAM Journal on Computing*, 19(6):1064–1067, 1990.

[AW15] Josh Alman and Ryan Williams. Probabilistic polynomials and hamming nearest neighbors. In *FOCS*, pages 136–150. IEEE, 2015.

[AW17] Josh Alman and R. Ryan Williams. Probabilistic rank and matrix rigidity. In *STOC*, pages 641–652. ACM, 2017.

[BFS86] László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory (preliminary version). In *FOCS*, pages 337–347, 1986.

[BGH$^+$06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM J. Comput.*, 36(4):889–974, 2006.

[BT94] Richard Beigel and Jun Tarui. On ACC. *Computational Complexity*, 4:350–366, 1994.

[BV14] Eli Ben-Sasson and Emanuele Viola. Short PCPs with projection queries. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 163–173, 2014.

[Cop82] Don Coppersmith. Rapid multiplication of rectangular matrices. *SIAM Journal on Computing*, 11(3):467–471, 1982.

[CW16] Timothy M Chan and Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing razborov-smolensky. In *SODA*, pages 1246–1255. SIAM, 2016.

[CW19a] Lijie Chen and Ruosong Wang. Classical algorithms from quantum and arthur-merlin communication protocols. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 23:1–23:20, 2019.

[CW19b] Lijie Chen and Ryan Williams. Stronger connections between circuit analysis and circuit lower bounds, via PCPs of proximity. 2019. CCC.

[Des07] Amit Jayant Deshpande. *Sampling-based algorithms for dimension reduction*. PhD thesis, Massachusetts Institute of Technology, 2007.

[DGW19] Zeev Dvir, Alexander Golovnev, and Omri Weinstein. Static data structure lower bounds imply rigidity. In *STOC*, pages 967–978. ACM, 2019.

[Fri93] Joel Friedman. A note on matrix rigidity. *Combinatorica*, 13(2):235–239, 1993.

[GHK$^+$12] Anna Gál, Kristoffer Arnsfelt Hansen, Michal Kouckỳ, Pavel Pudlák, and Emanuele Viola. Tight bounds on computing error-correcting codes by bounded-depth circuits with arbitrary gates. In *STOC*, pages 479–494. ACM, 2012.

[GPW16] Mika Göös, Toniann Pitassi, and Thomas Watson. Zero-information protocols and unambiguity in arthur-merlin communication. *Algorithmica*, 76(3):684–719, 2016.

[GPW18] Mika Göös, Toniann Pitassi, and Thomas Watson. The landscape of communication complexity classes. *Computational Complexity*, 27(2):245–304, 2018.

[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.

[GT16] Oded Goldreich and Avishay Tal. Matrix rigidity of random toeplitz matrices. In *STOC*, pages 91–104. ACM, 2016.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. Comput. Syst. Sci.*, 65(4):672–694, 2002.

[KS12] Swastik Kopparty and Srikanth Srinivasan. Certifying polynomials for AC^0 (parity) circuits, with applications. In *FSTTCS*, 2012.

[KV19] Mrinal Kumar and Ben Lee Volk. Lower bounds for matrix factorization, 2019.

[KvM02] Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput.*, 31(5):1501–1526, 2002.

[KW16] Daniel M. Kane and Ryan Williams. Super-linear gate and super-quadratic wire lower bounds for depth-two and depth-three threshold circuits. In *STOC*, pages 633–643. ACM, 2016.

[Lok00] Satyanarayana V Lokam. On the rigidity of vandermonde matrices. *Theoretical Computer Science*, 237(1-2):477–483, 2000.

[Lok01] Satyanarayana V Lokam. Spectral methods for matrix rigidity with applications to size–depth trade-offs and communication complexity. *Journal of Computer and System Sciences*, 63(3):449–473, 2001.

[Lok06] Satyanarayana V Lokam. Quadratic lower bounds on matrix rigidity. In *International Conference on Theory and Applications of Models of Computation*, pages 295–307. Springer, 2006.

[Lok09] Satyanarayana V Lokam. Complexity lower bounds using linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 4(1–2):1–155, 2009.

[LPT$^+$17] Daniel Lokshtanov, Ramamohan Paturi, Suguru Tamaki, R. Ryan Williams, and Huacheng Yu. Beating brute force for systems of polynomial equations over finite fields. In *SODA*, pages 2190–2202. SIAM, 2017.

[Lup56] Oleg B Lupanov. On rectifier and switching-and-rectifier schemes. In *Dokl. Akad. Nauk SSSR*, volume 111, pages 1171–1174, 1956.

[MT98] Alexis Maciel and Denis Therien. Threshold circuits of small majority-depth. *Information and Computation*, 146(1):55–83, 1998.

[MV05] Peter Bro Miltersen and N. V. Vinodchandran. Derandomizing arthur-merlin games using hitting sets. *Computational Complexity*, 14(3):256–279, 2005.

[MW18] Cody Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime: an easy witness lemma for NP and NQP. In *STOC*, pages 890–901. ACM, 2018.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[OS17] Igor Carboni Oliveira and Rahul Santhanam. Pseudodeterministic constructions in subexponential time. In *STOC*, pages 665–677. ACM, 2017.

[Par19] Orr Paradise. Smooth and strong PCPs, 2019.

[Pud94] Pavel Pudlak. Large communication in constant depth circuits. *Combinatorica*, 14(2):203–216, 1994.

[Raz89] Alexander A. Razborov. On rigid matrices (in Russian), 1989.

[RTS00] Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Discrete Mathematics*, 13(1):2–24, 2000.

[Spi96] Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Trans. Information Theory*, 42(6):1723–1731, 1996.

[SS96] Victor Shoup and Roman Smolensky. Lower bounds for polynomial evaluation and interpolation problems. *computational complexity*, 6(4):301–311, 1996.

[SSS97] Mohammad Amin Shokrollahi, Daniel A Spielman, and Volker Stemann. A remark on matrix rigidity. *Information Processing Letters*, 64(6):283–285, 1997.

[Tam16] Suguru Tamaki. A satisfiability algorithm for depth two circuits with a sub-quadratic number of symmetric and threshold gates. *Electronic Colloquium on Computational Complexity (ECCC)*, 23(100):100, 2016.

[Val77] Leslie G Valiant. Graph-theoretic arguments in low-level complexity. In *International Symposium on Mathematical Foundations of Computer Science*, pages 162–176. Springer, 1977.

[Wil13] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM Journal on Computing*, 42(3):1218–1244, 2013.

[Wil14a] Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. In *STOC*, pages 194–202. ACM, 2014.

[Wil14b] Ryan Williams. Nonuniform ACC circuit lower bounds. *Journal of the ACM (JACM)*, 61(1):2, 2014.

[Wil18a] R. Ryan Williams. Counting solutions to polynomial systems via reductions. In *SOSA*, pages 6:1–6:15, 2018.

[Wil18b] Richard Ryan Williams. Limits on representing boolean functions by linear combinations of simple functions: Thresholds, relus, and low-degree polynomials. In *CCC*, pages 6:1–6:24, 2018.

[Wun12] Henning Wunderlich. On a theorem of razborov. *Computational Complexity*, 21(3):431–477, 2012.

[Yao90] Andrew Chi-Chih Yao. On ACC and threshold circuits. In *FOCS*, pages 619–627. IEEE, 1990.

[Yek12] Sergey Yekhanin. Locally decodable codes. *Foundations and Trends in Theoretical Computer Science*, 6(3):139–255, 2012.

[Zák83] Stanislav Zák. A Turing machine time hierarchy. *Theor. Comput. Sci.*, 26:327–333, 1983.