The Hardest Explicit Construction

Oliver Korten
Department of Computer Science
Columbia University
New York, NY
oliver.korten@columbia.edu

2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS) | 978-1-6654-2055-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/FOCS52979.2021.00051

***Abstract*—We investigate the complexity of explicit construction problems, where the goal is to produce a particular object possessing some pseudorandom property in time polynomial in the size of that object. We give overwhelming evidence that APEPP, defined originally by Kleinberg et al. [12], is the natural complexity class associated with explicit constructions of objects whose existence follows from the probabilistic method, by placing a variety of such construction problems in this class. We then demonstrate that a result of Jeřábek [10] on provability in Bounded Arithmetic, when reinterpreted as a reduction between search problems, shows that constructing a truth table of high circuit complexity is complete for APEPP under NP-oracle reductions. This illustrates that Shannon's classical proof of the existence of hard boolean functions is in fact a *universal* probabilistic existence argument: derandomizing his proof implies a generic derandomization of the probabilistic method. As a corollary, we prove that $\mathbf{EXP}^{\mathbf{NP}}$ contains a language of mildly-exponential circuit complexity if and only if it contains a language of nearly maximum circuit complexity. Finally, for several of the problems shown to lie in APEPP, we demonstrate direct polynomial time reductions to the explicit construction of hard truth tables.**

***Keywords*-circuit complexity; psuedorandomness; total function complexity;**

Note: Various proofs are either abridged or omitted in this extended abstract; proofs of all stated results can be found in the full version: https://arxiv.org/abs/2106.00875

I. Introduction

Explicit construction — the task of replacing a nonconstructive argument for the existence of a certain type of object with a deterministic algorithm that outputs one — is an important genre of computational problems, one whose history is intertwined with the most fundamental questions in complexity and derandomization. The primary method of non-constructive argument for these sorts of problems is to show that a random object has the desired property with high probability. This technique, initiated by Erdös [5] and since dubbed the "probabilistic method," has proven immensely useful across disparate subfields of combinatorics and computer science. Indeed, the probabilistic method is currently our sole source of certainty that there exist hard Boolean functions, pseudorandom generators, rigid matrices, and optimal randomness extractors, among a variety of other combinatorial objects.

Explicit construction problems can be phrased, in complexity terms, as sparse search problems: given the input 1^n, output some object of size n satisfying a certain property. In the interesting case, such problems are also *total*: we have a reason to believe that for all n, at least one object with this property exists. In contrast to the fundamental importance of explicit constructions, there has been surprisingly little work attempting to systematically study their complexity. This gap was pointed out previously by Santhanam [19], who investigated the complexity of explicit construction problems from the following perspective: say we have some property Π which is *promised* to hold for almost all strings of length n. Based on the complexity of testing the property Π, what can be said about the complexity of producing an n-bit string with property Π? Though some interesting reductions can be shown in this framework, Santhanam notes that this approach does not seem to yield robust complexity classes with *complete* explicit construction problems.

This issue is familiar in the study of the class **TFNP**: when we have only a *promise* that a search problem is total, it is seemingly impossible to reduce it to a problem of similar complexity which has a *syntactic* guarantee of totality. This led to the study, initiated by Papadimitriou [16], of characterizing total search problems based on the combinatorial lemma which guarantees the existence of a solution.

So what is the basic combinatorial lemma guaranteeing the soundness of the probabilistic method of construction? Generally speaking, most constructions using the probabilistic method can be rephrased as *encoding arguments*: they demonstrate that whenever an object x of size n fails to possess a desired property (i.e. x is "bad"), this implies a succinct encoding of x using fewer than n bits, from which we can then recover x. The existence of a "good" x thus follows from the fact that there is no encoding scheme for arbitrary n-bit strings using fewer than n bits – phrased differently, there is no surjective function from $\{0,1\}^{n-1}$ to $\{0,1\}^n$. In recent work of Kleinberg et al. [12], a natural class capturing the complexity of this sort of encoding argument was presented. In particular they define the class **APEPP**, which consists of the $\mathbf{\Sigma_2^P}$ total search problems whose totality follows from the "Abundant Empty Pigeonhole Principle," which tells us that for any

2575-8454/21/$31.00 ©2021 IEEE
DOI 10.1109/FOCS52979.2021.00051

$f : \{0,1\}^{n-1} \to \{0,1\}^n$, there must exist some $y \in \{0,1\}^n$ such that for all $x \in \{0,1\}^{n-1}$, $f(x) \neq y$. In this paper, we show that **APEPP** is the natural *syntactic* class into which we can place a vast range of explicit construction problems where a solution is guaranteed by the probabilistic method.

Given that **APEPP** is a syntactic class, it is natural to ask whether some explicit construction problem is complete for it. As it turns out, the answer is positive: constructing a truth table of length 2^n with circuit complexity $2^{\epsilon n}$ is in fact *complete* for **APEPP** under $\mathbf{P^{NP}}$ reductions. Perhaps surprisingly, this important fact had been known for many years in the universe of Bounded Arithmetic, essentially proved in Emil Jeřábek's PhD thesis in 2004. Here Jeřábek shows that the theorem asserting the empty pigeonhole principle is *equivalent*, in a particular theory of Bounded Arithmetic, to the theorem asserting the existence of hard boolean functions. Although his result is phrased in terms of logical expressibility, we show that when translated to language of search problems his techniques give a $\mathbf{P^{NP}}$ reduction from any problem in **APEPP** to the problem of constructing a hard truth table. In Section IV we give a self-contained proof of this, and generalize the reduction to hold for arbitrary classes of circuits equipped with oracle gates. Combined with our results placing a wide range of explicit construction problems in **APEPP**, this shows that in a concrete sense, constructing a hard truth table is a *universal explicit construction problem*. We give further credence to this claim by showing in addition that several well known explicit construction problems in **APEPP,** including the explicit construction of rigid matrices, can be directly reduced to the problem of constructing a hard truth table via polynomial time reductions (as opposed to $\mathbf{P^{NP}}$ reductions).

A. Contributions

We investigate the complexity class **APEPP** introduced in [12], defined by the following complete problem EMPTY: given a circuit $C : \{0,1\}^n \to \{0,1\}^m$ with $m > n$, find an m-bit string outside the range of C. In Section III we give overwhelming evidence that **APEPP** is the natural class associated with explicit constructions from the probabilistic method, by placing a wide range of well-studied problems in this class. In particular, we show that the explicit construction problems associated with the following objects lie in **APEPP**:

- Truth tables of length 2^n with circuit complexity $\frac{2^n}{2n}$ (Theorem 1)
- Pseudorandom generators (Theorem 2)
- Strongly explicit two-source randomness extractors with 1 bit output for min-entropy $\log n + O(\log(1/\epsilon))$, and thus strongly explicit $O(\log n)$-Ramsey graphs in both the bipartite and non-bipartite case (Theorem 3)
- Matrices with high rigidity over any finite field (Theorem 4)
- Strings of time-bounded Kolmogorov complexity $n-1$ relative to any fixed polynomial time bound and any fixed Turing machine (Theorem 7)
- Communication problems outside of $\mathbf{PSPACE^{CC}}$
- Hard data structure problems in the-bit probe model

Since the work of Impagliazzo and Wigderson [9] implies that constructing pseudorandom generators reduces to constructing hard truth tables, **APEPP** constructions of PRGs follow immediately from **APEPP** constructions of hard truth tables. However, we provide a self-contained and simple proof that PRG construction can be reduced to EMPTY, without requiring the more involved techniques of Nisan, Wigderson, and Impagliazzo [15][9]. Together with the result in the following section that constructing hard truth tables is complete for **APEPP** under $\mathbf{P^{NP}}$ reductions, this gives an alternative and significantly simplified proof that worst-case-hard truth tables can be used to derandomize algorithms (although it proves a weaker result, that this derandomization can be accomplished with an **NP** oracle).

In Section IV we show that constructing a truth table of length 2^n with circuit complexity $2^{\epsilon n}$ is complete for **APEPP** under $\mathbf{P^{NP}}$ reductions (for any fixed $0 < \epsilon < 1$). As discussed earlier, the core argument behind this result was proven by Jeřábek in [10], where he shows that the theorem asserting the existence of hard boolean functions is equivalent to the theorem asserting the empty pigeonhole principle in a certain fragment of Bounded Arithmetic. We show that, when viewed through the lens of explicit construction problems, this technique yields a reduction from EMPTY to the explicit construction of hard truth tables. We also generalize this reduction to arbitrary oracle circuits, which allows us to prove the following more general statement: constructing a truth table which requires large $\Sigma_i^{\mathbf{P}}$-oracle circuits is complete for $\mathbf{APEPP}_{\Sigma_i^{\mathbf{P}}}$ under $\Delta_{i+2}^{\mathbf{P}}$ reductions (the complete problem for $\mathbf{APEPP}_{\Sigma_i^{\mathbf{P}}}$ is the variant of EMPTY where the input circuit can have $\Sigma_i^{\mathbf{P}}$-oracle gates).

By recasting and generalizing Jeřábek's theorem in the context of explicit construction problems, we are able to derive several novel results. First and foremost, we conclude that there is a $\mathbf{P^{NP}}$ construction of hard truth tables *if and only if* there is a $\mathbf{P^{NP}}$ algorithm for every problem in **APEPP**, and so in particular such a construction of hard truth tables would automatically imply $\mathbf{P^{NP}}$ constructions for each of the well-studied problems discussed in Section III. This tells us that constructing hard truth tables is, in a definite sense, *a universal explicit construction problem*. Since the existence of a $\mathbf{P^{NP}}$ construction of hard truth tables is equivalent to the existence of a language in $\mathbf{E^{NP}}$ with circuit complexity $2^{\Omega(n)}$, this completeness result actually gives an *exact algorithmic characterization* of proving $2^{\Omega(n)}$ circuit lower bounds for $\mathbf{E^{NP}}$:

Theorem (Theorem 11)**.** *There is a* $\mathbf{P^{NP}}$ *algorithm for*

EMPTY *if and only if* $\mathbf{E^{NP}}$ *contains a language of circuit complexity* $2^{\Omega(n)}$.

As a corollary we are able to derive the following:

Theorem (Corollaries 2 and 3)**.** $\mathbf{E^{NP}}$ *(resp.* $\mathbf{EXP^{NP}}$*) contains a language of circuit complexity* $2^{\Omega(n)}$ *(resp.* $2^{n^{\Omega(1)}}$*)* if and only if $\mathbf{E^{NP}}$ *(resp.* $\mathbf{EXP^{NP}}$*) contains a language of circuit complexity* $\frac{2^n}{2n}$.

Unpacking the proof of the above corollaries reveals an efficient algorithm to "extract hardness" from truth tables using an oracle for circuit minimization, a prospect previously considered in [3]:

Theorem (Theorem 12)**.** *There is a polynomial time algorithm using a circuit minimization oracle (or more generally an* **NP** *oracle) which, given a truth table* x *of length* M *and circuit complexity* s*, outputs a truth table* y *of length* $N = \Omega(\sqrt{\frac{s}{\log M}})$ *and circuit complexity* $\Omega(\frac{N}{\log N})$.

Finally, in Section V we consider **P** (as opposed to $\mathbf{P^{NP}}$) reductions from particular explicit construction problems to the problem of constructing hard truth tables. We show that in the case of rigidity, bit probe lower bounds, and certain communication complexity lower bounds, such reductions exist. These reductions take the following form: we show that the failure of an n-bit string x to satisfy certain pseudorandom properties implies a smaller than worst case circuit computing x. This then implies that any n-bit string of sufficiently high circuit complexity will necessarily possess a variety of pseudorandom properties, including high rigidity, high space-bounded communication complexity, and high bit-probe complexity.

Another concrete takeaway from this work is that we demonstrate, for several well-studied problems, the weakest known assumptions necessary to obtain explicit constructions of a certain type (polynomial time constructions in some cases and $\mathbf{P^{NP}}$ constructions in others). Perhaps the most interesting application of this is rigidity, as the complexity of rigid matrix construction has been studied extensively in both the **P** and $\mathbf{P^{NP}}$ regimes [1]. We obtain the following conditional constructions of rigid matrices:

Theorem (Theorems 4 and 11)**.** *If* $\mathbf{E^{NP}}$ *contains a language of circuit complexity* $2^{\Omega(n)}$*, then for any prime power* $q \leq 2^{poly(n)}$ *and any* $\epsilon \leq \frac{1}{16}$*, there is a* $\mathbf{P^{NP}}$ *construction of an* $n \times n$ *matrix over* $\mathbb{F}_q$ *which is* ϵn^2*-far (in hamming distance) from any rank-*ϵn *matrix.*

Theorem (Theorem 13)**.** *If* **E** *contains a language of circuit complexity* $\Omega(\frac{2^n}{n})$*, then for some* $\epsilon > 0$ *there is a polynomial time construction of an* $n \times n$ *matrix over* $\mathbb{F}_2$ *which is* ϵn^2*-far from any rank-*ϵn *matrix.*

In both cases, the rigidity parameters in the conclusion would be sufficient to carry out Valiant's lower bound program [23]. The weakest hardness assumptions previously known to yield constructions with even remotely similar parameters (in either the $\mathbf{P^{NP}}$ or **P** regimes) require a lower bound against *nondeterministic* circuits [14].

B. Related Work

A large body of work on the hardness/randomness connection, starting with that of Nisan and Wigderson [15], has exhibited the usefulness of explicit constructions of hard truth tables. The results of Impagliazzo and Wigderson [9] give, in particular, a reduction from explicit constructions of hard truth tables to explicit constructions of pseudorandom generators that fool polynomial size circuits. As noted by Santhanam [19], this immediately implies that for any "dense" property Π recognizable in **P** (dense meaning the fraction of n-bit strings holding this property is at least $1/poly(n)$), an efficient construction of a hard truth table immediately implies an efficient construction of an n-bit string with property Π. But many properties of interest such as Rigidity (or any of the other properties studied in this work) are only known to be recognizable in the larger class **NP**. Under the stronger assumption that we can construct truth tables hard for certain classes of *nondeterministic* circuits, constructions for all dense **NP** properties are known to follow as well [13] [14], so in particular $\mathbf{P^{NP}}$ constructions for every problem in **APEPP** would follow. However, constructing truth tables that are hard for nondeterministic circuits appears strictly harder than constructing truth tables hard for standard circuits, and in particular does not seem to be contained in **APEPP**, so although this yields an explicit construction problem which is *hard* for **APEPP**, it does not appear to be complete. In contrast, we show here that constructing a truth table which is hard for standard circuits is both contained in and hard for **APEPP**, thus showing that a $\mathbf{P^{NP}}$ construction of a hard truth table is possible *if and only if* such a construction is possible for every problem in **APEPP**.

For several of the explicit construction problems we study, a long line of work has gone into improving state-of-the-art constructions. A more detailed overview of this work in the important cases of rigidity and two-source extractors can be found in the full version of this paper. The crucial takeaway is that for each of the problems shown reducible to EMPTY in Section III, obtaining constructions with the same parameters by explicit means is an open problem.

C. Proof Sketch of Main Theorem

We give here an informal overview of the proof that EMPTY can be solved in polynomial time given access to a hard truth table and an **NP** oracle. At the core of this proof is a familiar construction in the theory of computing which dates back to the 1980's, namely the pseudorandom function generator of Goldreich, Goldwasser, and Micali [6]. Note that in the following, we will refer to the "circuit

complexity of an n-bit string x" to mean the size of the smallest circuit computing x_i given i in binary; this is well-defined even when n is not a power of 2, as we shall formalize Section III[1].

Consider the special case of EMPTY where our input is a circuit $C : \{0,1\}^n \to \{0,1\}^{2n}$ which exactly doubles its input size. For a moment let us forget our primary goal of finding a $2n$-bit string outside C's range, and instead consider C as a cryptographic pseudorandom generator which we are attempting to break. Since C is a function which extends its input size by a positive number of bits, it is indeed of the same syntactic form as a cryptographic PRG, so this viewpoint is well-defined.

In [6], Goldreich, Goldwasser and Micali give a procedure[2] which, for any fixed $0 < \epsilon < 1$, takes C and produces in polynomial time a new circuit $C^* : \{0,1\}^n \to \{0,1\}^m$ for some $m = poly(n)$, which satisfies the following two properties:

1) Every string in the range of C^* has circuit complexity at most m^ϵ.
2) Given a statistical test breaking C^*, we can construct a statistical test of similar complexity breaking C.

The construction of C^* is in fact quite simple: for an appropriate choice of k, we recursively apply C to an n-bit input for k iterations as follows: first apply C to an n-bit string to get 2 n-bit strings, then apply it again to each of those to get 4, and continue k times until we obtain 2^k n-bit strings.

A key observation made by Razborov and Rudich [18] is that condition (1) automatically implies a *particular statistical test which breaks* C^*, namely the test which accepts precisely those m-bit strings with circuit complexity exceeding m^ϵ. But by property (2), C^* inherets the security of C, which is an arbitrary candidate PRG. This means that determining if an m-bit string has circuits of size m^ϵ is in fact a *universal test for randomness*, capable of simultaneously breaking all pseudorandom generators.

Recall now our original goal for C, which was solve the associated instance of EMPTY by finding a $2n$-bit string outside its range. Property (1) of C^* implies that an explicit construction of a length-m truth table of circuit complexity m^ϵ would immediately yield an explicit m-bit string outside the range of C^*. In Section IV, we show that C^* obeys the following third property:

3) Given a string outside the range of C^*, we can find a string outside the range of C using a polynomial number of calls to an **NP** oracle.

The analogue of statement (3) in the context of Bounded Arithmetic was first shown by Jeřábek [10], and a quite similar argument appears even earlier in the work of Paris, Wilkie, and Woods [17]. Combining properties (1) and (3), we get the desired result: any m-bit string of complexity m^ϵ must lie outside the range of C^*, so using such a string together with an **NP** oracle we can solve our original instance of EMPTY.

To summarize, the construction C^* of Goldreich, Goldwasser and Micali shows that the property of requiring large circuits is a *universal pseudorandom property of strings* in two concrete senses:

- (Original analysis of [6] and [18]) A test determining whether a string requires large circuits can be efficiently boostrapped into a test distinguishing any pseudorandom distribution from the uniform distribution.
- (This work together with Jeřábek [10]) An explicit example of a string requiring large circuits can be used to generate an explicit example of a string outside the range of any efficiently computable map $C : \{0,1\}^n \to \{0,1\}^{2n}$ (in fact any $C : \{0,1\}^n \to \{0,1\}^{n+1}$ as shown in Section IV), and in particular can be used to construct explicit examples of strings posessing each of the fundamental pseudorandom properties examined in Section III.

II. DEFINITIONS

Following [12], we define the set of total functions in $\mathbf{\Sigma_2^P}$, denoted $\mathbf{TF\Sigma_2^P}$, as follows:

Definition 1. *A relation $R(x,y)$ is in $\mathbf{TF\Sigma_2^P}$ if there exists a polynomial $p(n)$ such that the following conditions hold:*

1) *For every x, there exists a y such that $|y| \leq p(|x|)$ and $R(x,y)$ holds*
2) *There is a polynomial time Turing machine M such that*
   $R(x,y) \iff \forall z \in \{0,1\}^{p(|x|)} M(x,y,z)$ accepts

The search problem associated with such a relation is: "given x, find some y such that $R(x,y)$ holds." For the majority of this paper, we will be concerned primarily with *sparse* $\mathbf{TF\Sigma_2^P}$ search problems, where the only relevant part of the input is its length. We can thus define the following "sparse" subclass of $\mathbf{TF\Sigma_2^P}$:

Definition 2. *A relation $R(x,y)$ is in $\mathbf{STF\Sigma_2^P}$ if $R \in \mathbf{TF\Sigma_2^P}$ and for any x_1, x_2 such that $|x_1| = |x_2|$, we have that for all y, $R(x_1,y) \Leftrightarrow R(x_2,y)$.*

Since the length of x fully determines the set of solutions, the relevant search problem here is: "given 1^n, find some y such that $R(1^n,y)$ holds." All explicit construction problems considered in Section III will be in $\mathbf{STF\Sigma_2^P}$.

We now define the search problem EMPTY, which will be the primary subject of this work:

[1]See Definition 4

[2][6] and [18] apply the construction described here in a different parameter regime, so our statement of the result differs slightly from its original presentation. The version described here has been noted subsequently in the literature on MCSP, see for example [20].

Definition 3. EMPTY *is the following search problem: given a boolean circuit C with n input wires and m output wires where $m > n$, find an m-bit string outside the range of C.*

This problem is total due to the basic lemma, referred to in [12] as the "Empty Pigeonhole Principle" and in the field of Bounded Arithmetic as the "Dual Pigeonhole Principle [10]," which tells us that a map from a smaller set onto a larger one cannot be surjective. Since verifying a solution y consists of determining that for all x, $C(x) \neq y$, we have:

Observation 1. EMPTY $\in \mathbf{TF\Sigma_2^P}$

Note that for any instance of EMPTY the number of output bits m is at least $n+1$, so a random m-bit string will be a solution with probability at least $\frac{1}{2}$. Since verifying a solution can be accomplished with one call to an **NP** oracle, this implies the following inclusion:

Observation 2. EMPTY $\in \mathbf{FZPP^{NP}}$

As mentioned in the introduction, this fact tells us that sufficiently strong pseudorandom generators capable of fooling *nondeterministic* circuits such as those in [13] would suffice to derandomize the above inclusion and yield a $\mathbf{P^{NP}}$ algorithm for EMPTY. In Section IV, we will show that this derandomization can be accomplished under a significantly weaker assumption, using a reduction of a very different form then the hardness-based pseudorandom generators of [15], [9], and [13].

We can now define the class **APEPP**, which is simply the class of search problems polynomial-time reducible to EMPTY. This class was originally defined in [12], and is an abbreviation for "Abundant Polynomial Empty Pigeonhole Principle." The term "Abundant" was used to distinguish this from the larger class **PEPP** also studied in [12]. The complete problem for **PEPP** is to find a string outside the range of a map $C : \{0,1\}^n \setminus \{0^n\} \rightarrow \{0,1\}^n$, which appears significantly more difficult (it is at least as hard as **NP** [12]). The distinction between **APEPP** and **PEPP** also appears in the Bounded Arithmetic literature, where the principle corresponding to **APEPP** is referred to as the "Dual *weak* Pigeonhole Principle," while the principle corresponding to **PEPP** is referred to simply as the "Dual Pigeonhole Principle." We will be concerned only with the abundant/weak principle in this work. It should be noted that we employ a slight change of notation from [12] for the sake of simplicity: we use EMPTY to refer to the search problem associated with the *weak* pigeonhole principle, while in [12] EMPTY refers to the search problem associated with the full pigeonhole principle.

Infinitely-often vs. almost-everywhere circuit lower bounds: As a final point of clarification, whenever we make the statement "L requires circuits of size $s(n)$" for some language L and size bound s, we mean that circuits of size $s(n)$ are required to compute L on length n inputs for *all but finitely many* n. This is in contrast to the statement "$L \notin \mathbf{SIZE}(s(n))$," which means the circuit size lower bound holds for infinitely many input lengths. All circuit lower bounds referred to in this work will be of the first kind.

III. EXPLICIT CONSTRUCTIONS IN APEPP

In this section, we show that a variety of well-studied explicit construction problems can be reduced in polynomial time to EMPTY. Each proof follows roughly the following format: there is some property of interest Π, and our goal is to construct an n-bit string which holds this property. For each such Π we consider, whenever an n-bit string x fails to have this property, it indicates that x is somehow more "structured" than a random n-bit string, and this structure allows us to specify x using fewer then n bits. We then actualize this argument in the form of an efficiently computable map $C : \{0,1\}^k \rightarrow \{0,1\}^n$ with $k < n$, such that any string not having property Π is in the range of C. This immediately implies that any n-bit string outside the range of C must hold property Π, and thus any solution to the instance of EMPTY defined by C will be a solution to our explicit construction problem. For many of the proofs, we will only show that the reduction is valid for n sufficiently large; clearly this is sufficient, since explicit constructions can be done by brute force for fixed input lengths.

A useful coding lemma: In the proofs to come, it will be helpful to utilize succinct and efficiently computable encodings of low-weight strings (the "weight" of binary string is the number of 1 bits it contains). In particular we rely on the following result in [7]:

Lemma 1. *[7] For any $k \leq n$, there exists a map $\Phi : \{0,1\}^{\log \binom{n}{k}} \rightarrow \{0,1\}^n$ computable in $poly(n)$ time such that any n-bit string of weight k is in the range of Φ.*

and a useful corollary:

Lemma 2. *For any $0 < \epsilon < \frac{1}{2}$, there exists a map $\Phi : \{0,1\}^{n-\epsilon^2 n + \log n} \rightarrow \{0,1\}^n$ computable in $poly(n)$ time such that any n-bit string of weight at most $\frac{n}{2} - \epsilon n$ is in the range of Φ.*

A. Hard Truth Tables

Definition 4. *Given a string x of length N, we say that x is computed by a circuit of size s if there is a boolean circuit C of fan-in 2 over the basis $\langle \wedge, \vee, \neg \rangle$ with $\lceil \log N \rceil$ inputs and s gates, such that $C(i) = x_i$ for all $1 \leq i \leq |x|$. If N is not a power of 2, we put no restriction on the value of $C(i)$ for $i > |x|$.*

Definition 5. HARD TRUTH TABLE *is the following search problem: given 1^N, output a string x of length N such that x is not computed by any circuit of size at most $\frac{N}{2 \log N}$.*

In the typical case where $N = 2^n$ for some n, this is equivalent to finding a truth table for an n-input boolean

function requiring circuits of size $\frac{2^n}{2n}$, which is within a $2+o(1)$ factor of the worst case circuit complexity for any n-input boolean function.

Theorem 1. HARD TRUTH TABLE *reduces in polynomial time to* EMPTY.

Proof: This proof follows Shannon's classical argument for the existence of functions of high circuit complexity [21]. We construct an instance of EMPTY in the form of a circuit Φ which maps an encoding of a circuit to its corresponding truth table. Φ interprets its input as a circuit on $\lceil \log N \rceil$ bits (using an encoding of circuits to be described below), tests its value on every possible input to generate a $2^{\lceil \log N \rceil}$ bit truth table, and then truncates this truth table to be of length exactly N.

Given a circuit of size $s \geq n$, we can encode it in a straightforward way using $2s \log s + O(s)$ bits by specifying explicitly the two inputs of each gate, a list of the logical types of each gate, and an indication of which gate is the output. It is also clear that from such an encoding we can efficiently decode the represented circuit and test it on all possible input values. In this way we can construct our circuit Φ to interpret its $2s \log s + O(s)$-bit input as a circuit encoding of this form, and then print the truth table corresponding to that encoded circuit. If we chose $s = \frac{N}{2 \log N}$, we have $2s \log s + O(s) < N$ (for N sufficiently large). So Φ is a valid instance of EMPTY, and any string outside its range is a solution to HARD TRUTH TABLE. It is also clear from the above description that Φ can be constructed in $poly(N)$ time. ∎

B. Pseudorandom Generators

Definition 6. *We will say that a sequence $R = (x_1, \ldots, x_m)$ of n-bit strings is a pseudorandom generator if, for all n-input circuits of size n:*

$$|Pr_{x \sim R}[C(x) = 1] - Pr_{y \sim \{0,1\}^n}[C(y) = 1]| \leq 1/n$$

Standard applications of the probabilistic method show that such pseudorandom generators exist of size polynomial in n. Thus we can define the following total search problem:

Definition 7. PRG *is the following search problem: given 1^n, output a pseudorandom generator $R = (x_1, \ldots, x_m)$, $x_i \in \{0,1\}^n$.*

A polynomial time algorithm for PRG would suffice to derandomize **BPP** [15]. We now show how to formalize the argument for the totality of PRG using the empty pigeonhole principle. In particular, we show that a PRG of size n^6 can be constructed in **APEPP**.

As noted in the introduction, the results of Impagliazzo and Wigderson [9] imply that PRG reduces directly to HARD TRUTH TABLE, so a reduction of PRG to EMPTY follows from Theorem 1. However, we provide here a much simpler direct proof that PRG reduces to EMPTY, relying only on Yao's next bit predictor lemma, and neither the nearly disjoint subsets construction of Nisan and Wigderson [15] nor the rather involved worst-case to average-case reductions of Impagliazzo and Wigderson [9]. Together with our completeness result in Section IV, this gives an alternative, self-contained proof that worst-case-hard truth tables can be used to construct pseudorandom generators (although it yields a weaker result, as our derandomization will require an **NP** oracle).

Theorem 2. PRG *reduces in polynomial time to* EMPTY

Proof Sketch: Let R be a sequence $(x_1, \ldots, x_{n^6})$ of n^6 n-bit strings which fails to be a pseudorandom generator. By Yao's next bit predictor lemma [24][22], this implies the existence of a circuit D of size $O(n)$, such that:

$$Pr_{x \sim R}[D(x^{-i}) = x^i] > \frac{1}{2} + \frac{1}{n^2}$$

where x^i denotes the i^{th} bit of x, and x^{-i} is the $n-1$-bit string obtained by deleting x^i. In other words, from the description of D, the index i, and the sequence $R^{-i} = (x_1^{-1}, \ldots, x_{n^6}^{-i})$, we can efficiently reconstruct the i^{th} bit of x_j for at least $\frac{n^6}{2} + n^4$ indices $j \in [n^6]$. Thus, if we let S be the n^6-bit string denoting the indices where this guess is wrong, we have that from i, R^{-i}, D, S we can efficiently recover R exactly. Since S has at most $\frac{n^6}{2} - n^4$ non-zero entries, we can apply Lemma 2 to encode S using $n^6 - n^2 + O(\log n)$ bits. Clearly R^{-i} can be encoded with $n^6(n-1) = n^7 - n^6$ bits, D can be encoded with $O(n \log n)$ bits, and i with $\log n$ bits, so overall this encoding has length $n^7 - \Omega(n^2) + O(n \log n)$, which is strictly less than the n^7 bits needed to encode an arbitrary set R, for n sufficiently large.

Thus, we can construct an instance of EMPTY in the form of a circuit Φ with $\leq n^7 - 1$ input wires and n^7 output wires, which takes as input an encoding of the above form, and uses the above reconstruction procedure to produce R. By the arguments above, any R which fails to be a pseudorandom generator lies in the range of Φ, so any string outside its range is a solution to PRG. ∎

C. Strongly Explicit Randomness Extractors and Ramsey Graphs

A (k, ϵ) two-source extractor with one bit of output is a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ such that for any pair of distributions X, Y on $\{0,1\}^n$ of min-entropy at least k, the value of $f(xy)$ for a random $(x, y) \sim X \times Y$ is ϵ-close to an unbiased coin flip. By a well-known simplification [4], the following definition is in fact equivalent:

Definition 8. *We say that a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is a (k, ϵ) extractor if the following holds: for any two sets $X, Y \subseteq \{0,1\}^n$ of size 2^k, $|Pr_{x \sim X, y \sim Y}[f(xy) = 1] - \frac{1}{2}| \leq \epsilon$.*

Definition 9. *For any pair of functions $k, \epsilon : \mathbb{N} \to \mathbb{N}$, (k, ϵ)-*EXTRACTOR *is the following search problem: given 1^n, output a circuit C with $2n$ inputs such that the function $f_C : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ defined by C is a $(k(n), \epsilon(n))$ extractor.*

The above problem definition does not expressly constrain the size of C, though for a construction to be "explicit" in any useful sense (efficiently computable as a function of n), C would have to have size polynomial in n. The following reduction placing extractor construction in **APEPP** will immediately imply that we can construct a $(\log n + O(1), \epsilon)$ extractor of circuit size approximately n^3 in **APEPP** for any fixed ϵ.

Theorem 3. *For any efficiently computable $\epsilon(n)$ satisfying $\frac{1}{n^c} < \epsilon(n) < \frac{1}{2}$ for a constant c and sufficiently large n, $(\log n + 2\log(1/\epsilon(n)) + 3, \epsilon(n))$-*EXTRACTOR *reduces in polynomial time to* EMPTY.

Proof Sketch: Let $\epsilon = \epsilon(n)$, let $d = \lceil \frac{4}{\epsilon^2} \rceil$, and let A be any $2d^2n^3$-bit string, viewed as an ordered list of d^2n^2 elements of $\mathbb{F}_{2^{2n}}$ denoted $\alpha_1, \ldots \alpha_{d^2n^2}$. Now, consider the function $f : \{0,1\}^{2n} \to \{0,1\}^{2n}$ defined by

$$f(x) = \sum_{i=1}^{d^2n^2} \alpha_i x^{i-1}$$

and the function $g : \{0,1\}^{2n} \to \{0,1\}$ defined by

$$g(x) = f(x) \mod 2$$

We will show that if g fails to be a $(\log dn, \epsilon)$ extractor, this implies an encoding of A using strictly fewer then $2d^2n^3$ bits, from which we can efficiently reconstruct A. As in the proof of Theorem 2, this immediately implies a reduction to EMPTY (since for our choice of d we have $\log dn \leq \log n + 2\log(1/\epsilon) + 3$).

Say g is not a $(\log dn, \epsilon)$ extractor. So there exist two sets of n-bit strings X, Y, each of size $2^{\log dn} = dn$, and some $b \in \{0,1\}$ such that $Pr_{x \sim X, y \sim Y}[f(xy) = b] > \frac{1}{2} + \epsilon$. Let $R = \{xy \mid x \in X, y \in Y\} \subseteq \{0,1\}^{2n}$. We have $|R| = |X||Y| = d^2n^2$. Let $r_1, \ldots r_{d^2n^2}$ denote the lexicographical enumeration of R. By assumption, we have that $g(r_i) = b \mod 2$ for at least a $\frac{1}{2} + \epsilon$ fraction of indices i. So then, if we let β_i be the $2n-1$-bit prefix of $f(r_i)$, we can deduce the value of $f(r_i)$ from β_i and b for at least $d^2n^2(\frac{1}{2} + \epsilon)$ values of i. Thus, there is some $d^2n^2(\frac{1}{2} - \epsilon)$-weight d^2n^2-bit string S, such that from b, S, and the β_i's, we can deduce $f(r_i)$ for all i. Now, once we are able to deduce $f(x)$ for each of the d^2n^2 distinct values of x in R, since f is a degree d^2n^2-1 polynomial, we can uniquely and efficiently determine the coefficients α_i of f using Gaussian elimination on the corresponding $d^2n^2 \times d^2n^2$ Vandermonde matrix, and thus recover A.

It is clear that we can encode X, Y using $2dn^2$ bits, b using 1 bit, and the β_i's using $d^2n^2(2n-1) = 2d^2n^3 - d^2n^2$ bits. Since S is a $d^2n^2(\frac{1}{2} - \epsilon)$-weight d^2n^2-bit string, by Lemma 2, we can encode S using at most $d^2n^2(1 - \epsilon^2) + \log(d^2n^2) = d^2n^2 - \epsilon^2 d^2n^2 + \log(d^2n^2)$ bits. So the total bit length of this encoding is at most:

$$2dn^2 + 1 + d^2n^2 - \epsilon^2 d^2n^2 + \log(d^2n^2) + 2d^2n^3 - d^2n^2 = 2d^2n^3 + (2d - \epsilon^2 d^2)n^2 + 2\log(dn) + 1$$

Since we chose $\frac{4}{\epsilon^2} + 1 \geq d \geq \frac{4}{\epsilon^2}$, this is at most $2d^2n^3 - n^2 + O(\log dn)$, and by our assumption that $\frac{1}{\epsilon(n)} = poly(n)$ this is $2d^2n^3 - n^2 + O(\log n)$ overall, which is strictly less then the number of bits required to encode an arbitrary string A of length $2d^2n^3$ for n sufficiently large. ■

For the typical parameter regime where ϵ is an arbitrarily small constant, this gives a two source extractor for min-entropy $\log n + O(1)$, which is the best possible up to the $O(1)$ term [4].

Corollary 1. *Explicit construction of strongly explicit Ramsey graphs (n-vertex graphs containing no clique or independent set of size $c \log n$ for some constant c), in both the bipartite and non-bipartite case, reduces to* EMPTY.

Proof: As noted in [2], any two-source extractor in the above sense (with ϵ fixed to any constant less then one half) is automatically a bipartite Ramsey graph, and from a strongly explicit bipartite Ramsey graph we can construct a strongly explicit non-bipartite Ramsey graph efficiently. ■

D. Rigid Matrices

Definition 10. *[23] We say that $n \times n$ matrix M over $\mathbb{F}_q$ is (r, s) rigid if for any matrix $S \in \mathbb{F}_q^{n \times n}$ with at most s non-zero entries, $M + S$ has rank greater than r.*

Definition 11. *For any $q : \mathbb{N} \to \mathbb{N}$ such that $q(n)$ is a prime power $\forall n$, (ϵ, q)-*RIGID *is the following search problem: given 1^n, output an $n \times n$ matrix M over $\mathbb{F}_{q(n)}$ which is $(\epsilon n, \epsilon n^2)$ rigid.*

Theorem 4. *For any $\epsilon \leq \frac{1}{16}$, and any efficiently computable $q(n)$ satisfying the above, (ϵ, q)-*RIGID *reduces in polynomial time to* EMPTY.

Proof: Let M be any $n \times n$ matrix over $\mathbb{F}_q$ which is not (r, s) rigid. So there exists an $n \times r$ matrix L, an $r \times n$ matrix R, and an $n \times n$ matrix S with at most s non-zero entries, such that $M = LR + S$. It is clear that from the descriptions of L, R, S we can efficiently compute M.

L and R can each be described explicitly using $nr \log q$ bits. For S, we encode it by specifying an n^2-bit string T of weight s denoting the entries of S which are nonzero, together with an $s \log q$-bit string giving the values of the nonzero entries. Applying the encoding scheme in Lemma 1 for T, overall the number of bits in this encoding is at most $\log \binom{n^2}{s} + (2nr + s)\log q$. Setting $r = \epsilon n$ and $s = \epsilon n^2$ this

is at most:

$$\log \binom{n^2}{\epsilon n} + (3\epsilon n^2)\log q \leq$$
$$(\frac{3}{4} + \epsilon - \epsilon^2)n^2 + (3\epsilon n^2)\log q \leq$$
$$(\frac{3}{4} + \epsilon - \epsilon^2 + 3\epsilon)n^2 \log q$$

Since we chose $\epsilon \leq \frac{1}{16}$, this is at most $\frac{997}{1000}n^2 \log q$, which is strictly less then the $n^2 \log q$ bits needed to specify an arbitrary matrix in $\mathbb{F}_q^{n \times n}$. ∎

E. Other Problems

In Section V, we introduce two more explicit construction problems and show that each of these, in addition to a variant of the rigidity problem, can be reduced directly to HARD TRUTH TABLE in polynomial time. This also implies that both problems are contained in **APEPP**. We will postpone a formal definition of each of these new problems until Section V, but give an informal statement here for completeness:

Theorem 5 (Informal)**.** *Construction of a $2^n \times 2^n$ communication matrix which cannot by solved by any $o(n)$-space protocol reduces to* EMPTY *(such a matrix lies outside of the communication class $\mathbf{PSPACE^{CC}}$).*

Theorem 6 (Informal)**.** *The construction of a data structure problem of nearly maximum complexity in the bit-probe model reduces to* EMPTY.

We also demonstrate in the full version the following:

Theorem 7 (Informal)**.** *The construction of n-bit strings of time-bounded Kolmogorov complexity $n - 1$ reduces to* EMPTY*, for any fixed polynomial time bound.*

IV. CONSTRUCTING HARD TRUTH TABLES IS COMPLETE FOR APEPP

In this section we show that constructing a hard truth table is complete for **APEPP** under $\mathbf{P^{NP}}$ reductions. As mentioned before, the core of this theorem was originally proven by Jeřábek [10], and the main construction underlying the reduction dates back further to the work of Goldreich, Goldwasser, and Micali [6]. Jeřábek's result is phrased in the language of proof complexity, stating that the theorem asserting the existence of hard boolean functions is *equivalent* to the empty pigeonhole principle in a particular theory of Bounded Arithmetic. We demonstrate below that when translated to the language of search problems and explicit constructions, his proof yields a $\mathbf{P^{NP}}$ reduction from EMPTY to the problem of constructing a hard truth table. We in fact prove a more general statement here which holds for arbitrary circuit classes equipped with oracle gates.

Definition 12. *For any oracle A, the class of search problems $\mathbf{APEPP}^A$ is defined by the following complete problem* EMPTY^A*: given an A-oracle circuit with more output wires than input wires, find a boolean string whose length is equal to the number of output wires but which is not in the range of this circuit. For any strictly increasing function $f : \mathbb{N} \to \mathbb{N}$, we define the problem* $\text{EMPTY}^A_{f(n)}$*, which is the special case of* EMPTY^A *where the circuit is required to have $f(n)$ output wires, where n is the number of input wires.*

We now define the type of reduction used in this section:

Definition 13. *For any oracle A, an "A-circuit inverter oracle" (or simply A-inverter) is an oracle which, given an A-oracle circuit C and a potential output y, determines if there exists some x such that $C(x) = y$, and produces one if so. An A-inverter reduction is a polynomial time reduction that uses an A-inverter orace.*

Note that in the absence of an oracle, an inverter reduction is equivalent to a $\mathbf{P^{NP}}$ reduction, since inverting a standard boolean circuit is **NP**-complete.

We start with the following technical lemma, which allows us to restrict our attention to circuits with exactly twice as many outputs as inputs.

Lemma 3. *For any oracle A,* EMPTY^A_{2n} *is complete for* $\mathbf{APEPP}^A$ *under A-inverter reductions.*

We omit the proof of Lemma 3 here, as it utilizes similar techniques that appear in the following proof of Theorem 8. We now define the hard truth table construction problem that will be used in our reduction:

Definition 14. *Let ϵ-*HARD^A *denote the following search problem: given 1^N, output a string x of length N such that x cannot be computed by A-oracle circuits of size N^ϵ.*

In the absence of an oracle, we drop the superscript and refer to this problem simply as ϵ-HARD. For $N = 2^n$, a solution to ϵ-HARD on input 1^N is a truth table of a function on n variables requiring $2^{\epsilon n}$-sized circuits, the same object used to build the Impagliazzo-Wigderson generator.

Theorem 8. *Let A be an oracle and $\epsilon > 0$ be a constant such that ϵ-*HARD^A *is total for sufficiently large input lengths. Then* EMPTY^A *reduces in polynomial time to ϵ-*HARD^A *under A-inverter reductions.*

Proof Sketch: By Lemma 3 we know that EMPTY^A reduces to EMPTY^A_{2n} under A-inverter reductions. Now, let C be an instance of EMPTY^A_{2n} and let $k = 2\lceil \log |C| \rceil \lceil \frac{1}{\epsilon} \rceil$. Consider the following map $C^* : \{0,1\}^n \to \{0,1\}^{2^k n}$, defined informally as follows: given a string $x \in \{0,1\}^n$, apply C once to get 2 n-bit strings, then apply C to both of those n-bit strings to get four, and continue k times until we have 2^k n-bit strings, or equivalently a $2^k n$-bit string. This process is illustrated in Figure 1. As mentioned previously, this construction of C^* from C is essentially identical to the pseudorandom function generator of [6].

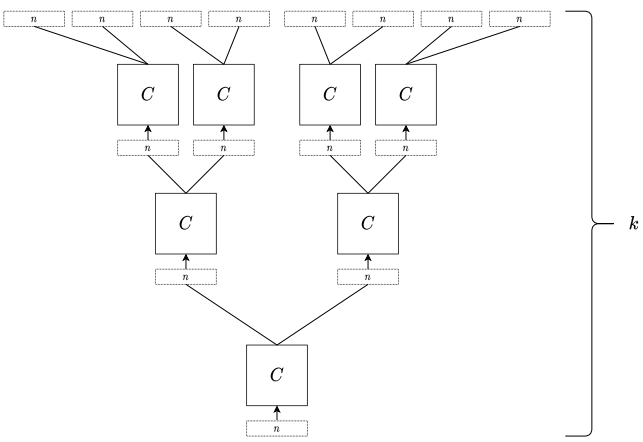

Figure 1. Extending a map $C : \{0,1\}^n \to \{0,1\}^{2n}$ to a map $C^* : \{0,1\}^n \to \{0,1\}^{2^k n}$. Dotted boxes indicate the number of bits along a wire.

The proof now proceeds by demonstrating two key properties of this construction C^*. First, we show that by setting $m = n2^k = poly(|C|)$, any solution to ϵ-HARD^A on input 1^m will be a string that is not in the range of C^*. Second, we will show that given a string outside the range of C^*, we can find a string outside the range of C using only a polynomial number of calls to an A-inverter.

To carry out the first of these steps, we will show that any string in the range of C^*, when interpreted as a truth table of length $m = n2^k$ on $\lceil \log n \rceil + k$ variables, can be computed by an A-oracle circuit of size $O(|C|k)$. Since, by construction of k, we have that $m \geq |C|^{\frac{2}{\epsilon}}$, a solution to ϵ-HARD^A on input 1^m will be a truth table of length m not computable by an A-oracle circuit of size $m^\epsilon \geq |C|^2$, and thus a circuit of size $O(|C|k) = O(|C| \log |C|)$ would be a contradiction for all input lengths greater than some absolute constant. We construct such a circuit for any string in the range of C^* as follows: let y be a $2^k n$-bit string such that for some $x \in \{0,1\}^n$, $C^*(x) = y$. The circuit computing y will have x written as advice/constants, and will feed x through k copies of the circuit C in series. We will split the $\lceil \log n \rceil + k$ input variables into a block of k variables we call i, and a block of $\lceil \log n \rceil$ variables we call j. We then use i to determine whether to take the first or last n bits of output from one of the copies of C before feeding it into the next, to get some resulting string x^i, and then we use j to index into the j^{th} position of x^i, to get $y_{i,j}$. A diagram of this circuit is shown in Figure 2.

Thus, we now know that any solution to ϵ-HARD^A on input 1^m will not be in the range of C^*, and by assumption ϵ-HARD$_{\mathcal{C}}$ is total for sufficiently large input lengths so such a solution exists. It remains only to show that we can use a string outside the range of C^*, together with a $\mathcal{C}$-inverter oracle, to find a string outside the range of C.

Let y be any string outside the range of C^*. Refer to

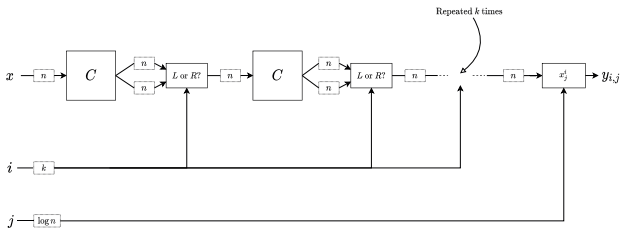

Figure 2. A succinct circuit whose truth table is y, for any y in the range of C^*. Dotted boxes indicate the number of bits along a wire. The circuit "L or R?" takes $2n$ bits plus an additional control bit, and based on the control bit either outputs the "leftmost" or "rightmost" block of n bits. Note that although x is shown as an input in this diagram, for any given y we fix a preimage x as constants/advice, and so the only true inputs to this circuit are i, j.

Figure 1 which gives a diagram of a circuit computing C^*; at a layer $i \in [k]$ of this circuit, we have 2^i blocks of n bits feeding into 2^i copies of C, and these copies of C then output 2^{i+1} blocks of n bits at the next layer. So working back from the output layer k, we can test if any consecutive $2n$-bit block of y is outside of the range of C. If none of them are, then we find a preimage for all blocks, interpret this as the output of the previous layer, and continue our search from there. We follow this process all the way back to the input layer or until we find an empty pigeonhole of C. If we never find an empty pigeonhole of C, then this process will terminate at the input layer with a string x such that $C^*(x) = y$, which is impossible by assumption, so at some point we must indeed find a string outside the range of C. Checking whether a particular string is an empty pigeonhole, or finding a preimage if it's not, can be accomplished with one call to an A-inverter by definition. We perform this test at most $2^k = poly(|C|)$ times (once for every copy of C in the diagram in Figure 1), so overall this process can be accomplished in polynomial time using an A-inverter. ■

We now examine the implications of this theorem for particular circuit classes of interest.

Theorem 9. *For any $0 < \epsilon < \frac{1}{2}$, ϵ-HARD$^{\Sigma_i^{\mathbf{P}}}$ is complete for* **APEPP**$^{\Sigma_i^{\mathbf{P}}}$ *under $\Delta_{i+2}^{\mathbf{P}}$ reductions.*

In the absence of any oracle gates, we have the following:

Theorem 10. *For any $0 < \epsilon < 1$, ϵ-HARD is complete for* **APEPP** *under* $\mathbf{P}^{\mathbf{NP}}$ *reductions.*

A. Implications of Completeness

This result gives an *exact algorithmic characterization* of the possibility of proving $2^{\Omega(n)}$ $\Sigma_i^{\mathbf{P}}$-circuit lower bounds for $\mathbf{E}^{\Sigma_{i+1}^{\mathbf{P}}}$:

Theorem 11. *There exists a language in $\mathbf{E}^{\Sigma_{i+1}^{\mathbf{P}}}$ with $\Sigma_i^{\mathbf{P}}$-circuit complexity $2^{\Omega(n)}$ if and only if there is a $\Delta_{i+2}^{\mathbf{P}}$ algorithm for* EMPTY$_{\Sigma_i^{\mathbf{P}}}$.

In the most interesting case, we conclude that a $2^{\Omega(n)}$

circuit lower bound for $\mathbf{E^{NP}}$ holds *if and only if* there is a $\mathbf{P^{NP}}$ algorithm for EMPTY. Together with the results in Section III, this gives newfound insight into the difficulty of proving exponential circuit lower bounds for the class $\mathbf{E^{NP}}$: proving such a lower bound requires solving a *universal explicit construction problem*, and would immediately imply $\mathbf{P^{NP}}$ constructions for a vast range of combinatorial objects which we currently have no means of constructing without a $\mathbf{\Sigma_2^P}$ oracle. Theorem 11 also allows us to derive the following interesting fact about the circuit complexity of $\mathbf{E^{NP}}$:

Corollary 2 (Worst-Case to Worst-Case Hardness Amplification in $\mathbf{E^{NP}}$)**.** *If there is a language in* $\mathbf{E^{NP}}$ *of circuit complexity* $2^{\Omega(n)}$*, then there is a language in* $\mathbf{E^{NP}}$ *requiring circuits of size* $\frac{2^n}{2n}$.

Proof: By Theorem 11, if there is a language in $\mathbf{E^{NP}}$ of circuit complexity $2^{\Omega(n)}$, then there is a $\mathbf{P^{NP}}$ algorithm for EMPTY. By Theorem 1, this implies a $\mathbf{P^{NP}}$ algorithm for HARD TRUTH TABLE, and thus a $\mathbf{P^{NP}}$ construction of a truth table of length N with hardness $\frac{N}{2\log N}$. This in turn implies the existence of a language in $\mathbf{E^{NP}}$ of circuit complexity $\frac{2^n}{2n}$. ∎

Tweaking the proof of Theorem 8 slightly we also obtain the following:

Corollary 3 (Worst-Case to Worst-Case Hardness Amplification in $\mathbf{EXP^{NP}}$)**.** *If there is a language in* $\mathbf{EXP^{NP}}$ *of circuit complexity* $2^{n^{\Omega(1)}}$*, then there is a language in* $\mathbf{EXP^{NP}}$ *requiring circuits of size* $\frac{2^n}{2n}$.

Proof: The proof follows that of the previous corollary, with the following modification to the reduction in Theorem 8: we start with the assumption that for some $\epsilon > 0$ we are able to construct N-bit truth tables with hardness $2^{\log^{\epsilon} N}$ in time quasipolynomial in N using an **NP** oracle, and then apply the same reduction setting $k = \log^{\lceil \frac{1}{\epsilon} \rceil} |C|$. ∎

We thus obtain a rather unexpected "collapse" theorem for the circuit complexity of $\mathbf{EXP^{NP}}$: if $\mathbf{EXP^{NP}}$ has circuits of size $\frac{2^n}{2n}$ infinitely often, then this class in fact has circuits of size $2^{n^{\epsilon}}$ infinitely often for every $\epsilon > 0$.

We can refine this slightly as follows.

Definition 15. MCSP*, defined originally in [11], is the following decision problem: given a truth table* x *and a size parameter* s*, determine whether* x *has a circuit of size at most* s*. Let* sMCSP *denote the search variant of this problem, where we are given a truth table* x *and must output a circuit computing* x *of minimum size.*

For the hardness amplification procedures in Corollaries 2 and 3, we can in fact replace the **NP** oracle with an oracle for sMCSP, which is non-trivial since sMCSP is not known to be **NP**-hard.

Corollary 4. *If there is a language in* $\mathbf{E^{sMCSP}}$ *(resp.* $\mathbf{EXP^{sMCSP}}$*) of circuit complexity* $2^{\Omega(n)}$ *(resp.* $2^{n^{\Omega(1)}}$*), then there is a language in* $\mathbf{E^{sMCSP}}$ *(resp.* $\mathbf{EXP^{sMCSP}}$*) requiring circuits of size* $\frac{2^n}{2n}$.

Proof: Recall the two reductions in Lemma 3 and Theorem 8. In order to find an empty pigeonhole of the input circuit C given a solution to ϵ-HARD, we only need to use the $\mathcal{C}$-inverter on C itself. In the case of a reduction from HARD TRUTH TABLE to ϵ-HARD, the circuit of interest C maps circuits of size at most $\frac{N}{2\log N}$ to their N-bit truth tables, and so an oracle for sMSCP would suffice to invert C. ∎

It should be noted that a related result was proven in [11], showing that this type of hardness amplification is possible in $\mathbf{E}$ assuming MCSP$\in \mathbf{P}$. However, their proof does not translate directly to an unconditional result in the oracle setting. Due to their use of the Impagliazzo-Wigderson generator, directly applying their proof in the oracle setting using the relativized generator of [13] would instead show that if $\mathbf{E^{MCSP}}$ requires $2^{\Omega(n)}$-sized *nondeterministic* circuits, then $\mathbf{E^{MCSP}}$ requires $\frac{2^n}{2n}$-sized *standard* circuits, which is a weaker statement then what is shown above (modulo the search/decision distinction between sMCSP and MCSP). Another result of a similar flavor was also proven in [8], which establishes that, assuming the (unproven) **NP**-completeness of MCSP, $2^{n^{\Omega(1)}}$ lower bounds for $\mathbf{NP} \cap \mathbf{coNP}$ imply $2^{\Omega(n)}$ lower bounds for $\mathbf{E^{NP}}$. This type of amplification is incomparable to the amplification demonstrated in Corollaries 2 and 3, although using Corollary 2 we can strengthen the lower bound in their conclusion to $\frac{2^n}{2n}$.

In [3], Buresh-Oppenheim and Santhanam define a notion of "hardness extraction" that is highly relevant to the results in this section. Informally, a hardness extractor is a procedure which takes a truth table of length N and circuit complexity s, and produces a truth table with nearly maximum circuit complexity relative to its size, whose length is as close to s as possible. The proof of Corollary 4 can in fact be viewed as a construction of a near-optimal hardness extractor using an sMCSP oracle. In particular our procedure is able to extract approximately the square root of the input's hardness:

Theorem 12. *There is a polynomial time algorithm using an* sMCSP *oracle which, given a truth table* x *of length* M *and circuit complexity* s*, outputs a truth table* y *of length* $N = \Omega(\sqrt{\frac{s}{\log M}})$ *and circuit complexity* $\Omega(\frac{N}{\log N})$.

V. DIRECT **P** REDUCTIONS TO HARD TRUTH TABLE

Ideally we could extend the completeness result in Theorem 10 to work with polynomial time reductions, as opposed to $\mathbf{P^{NP}}$ reductions. However, the **NP** oracle seems highly necessary for the proof techniques used above. Despite this obstacle, we show that there are several interesting problems

in **APEPP** which can be reduced to the problem of finding truth tables of hard functions via **P** reductions.

In the full version of this paper, three explicit construction problems are shown reducible to hard truth table construction:

1) Construction of rigid matrices over $\mathbb{F}_2$.
2) Construction of communication matrices outside of $\mathbf{PSPACE}^{\mathbf{CC}}$.
3) Construction of data structure problems of near-maximal bit-probe complexity.

Each of these reductions follows essentially the same format. Recall our common strategy for the reductions in Section III – to demonstrate that a certain construction problem reduces to EMPTY, it suffices to show that when a string fails to have the desired property, this implies a succinct representation from which we can efficiently recover that string. To reduce to HARD TRUTH TABLE, we will use roughly the same strategy, except we must prove in addition that we can use this succinct representation to recover *each bit* of the original string in *sublinear time*. Such a reconstruction procedure is, in essence, a circuit. We give here a proof only for the case of rigidity. To obtain the tightest reduction possible, we will introduce one new parameterized version of the hard truth table construction problem:

Definition 16. δ-QUITE HARD *is the following problem: given* 1^N*, output an* N*-bit truth table with hardness* $\frac{\delta N}{\log N}$

This problem is total for sufficiently small δ. Recall also the search problem (ϵ, q)-RIGID, where we are asked to construct a $(\epsilon n, \epsilon n^2)$ rigid matrix over $\mathbb{F}_q$.

Theorem 13. *For any sufficiently small* $\delta > 0$*, there exists some* $\epsilon > 0$ *such that* $(\epsilon, 2)$-RIGID *reduces in polynomial time to* δ-QUITE HARD.

Proof: To prove this, it suffices to show that for any matrix $M \in \mathbb{F}_2^{N \times N}$ which is not $(\epsilon N, \epsilon N^2)$-rigid, we can construct a boolean circuit with $f(\epsilon)O(\frac{N^2}{\log N})$ gates which decides the value of $M[i,j]$ given the $2\lceil \log N \rceil$-bit input (i,j), for some function f which approaches zero as ϵ approaches zero. This then implies that for any fixed δ, an N^2-bit truth table requiring circuits of size $\frac{\delta N^2}{\log N}$ must be $(\epsilon N, \epsilon N^2)$-rigid for some $\epsilon > 0$ which is a function only of δ (and otherwise determined by f and the constants hidden in the $O(\cdot)$ term).

Say M is not $(\epsilon N, \epsilon N^2)$-rigid. So there exists an $N \times \epsilon N$ matrix L, an $\epsilon N \times N$ matrix R, and an $N \times N$ matrix S with at most ϵN^2 nonzero entries, such that $LR \oplus S = M$. We will construct a circuit allowing us to efficiently index M which uses these matrices L, R, S as advice. To encode L and R, we can utilize the well-known theorem of Shannon that any truth table of length N can be computed by a circuit of size $O(\frac{N}{\log N})$ [21]. Thus, L can be specified as a list of ϵN circuits, each of size $O(\frac{N}{\log N})$, where the j^{th} circuit C_j represents the j^{th} column, and $C_j(i)$ computes $L[i,j]$. Then if we let $\mathsf{row}_L : \{0,1\}^{\log N} \to \{0,1\}^{\epsilon N} = C_1(i)C_2(i)\dots C_{\epsilon N}(i)$, we see that row_L has circuit size $\epsilon O(\frac{N^2}{\log N})$, and outputs the i^{th} row of L given i. We then analogously construct a circuit col_R for R, interchanging rows and columns in the above description. Finally, for S, we employ a refinement of Shannon's result due to Lupanov [7], which tells us that for sufficiently large N, any truth table of length N with at most ϵN nonzero entries can be computed by circuit of size

$$\frac{\log \binom{N}{\epsilon N}}{\log \log \binom{N}{\epsilon N}} + o\left(\frac{N}{\log N}\right) \leq H(\epsilon)O(\frac{N}{\log N})$$

Where H denotes the binary entropy function. Thus there is a circuit $\mathsf{entry}_S : \{0,1\}^{2\log N} \to \{0,1\}$ of size $H(\epsilon)O(\frac{N^2}{\log N})$ which computes $S[i,j]$ given i,j in binary.

Given these circuits $\mathsf{row}_L, \mathsf{col}_R, \mathsf{entry}_S$, computing $M[i,j]$ is straightforward. By definition, we have that:

$$M[i,j] = \langle \mathsf{row}_L(i), \mathsf{col}_R(j) \rangle \oplus \mathsf{entry}_S(i,j)$$

where the dot product is taken over $\mathbb{F}_2$. It is clear that the dot product of two ϵN bit strings can be computed by a circuit of size $O(\epsilon N)$, and that the final $\oplus$ operation can be implemented with a constant number of gates, so overall there must exist a circuit entry_M which computes $M[i,j]$ given i,j of size:

$$\epsilon O(\frac{N^2}{\log N}) + H(\epsilon)O(\frac{N^2}{\log N}) + O(\epsilon N) + O(1) = (\epsilon + H(\epsilon))O(\frac{N^2}{\log N}) + o(\frac{N^2}{\log N})$$

Since $\epsilon + H(\epsilon)$ aproaches zero as ϵ aproaches zero, this gives the required result. ∎

We thus conclude that if **E** contains a language of circuit complexity $\Omega(\frac{2^n}{n})$, then there is a polynomial time construction of $(\Omega(n), \Omega(n^2))$-rigid matrices over $\mathbb{F}_2$.

ACKNOWLEDGMENT

The author would like to thank Christos Papadimitriou for his guidance and for many inspiring discussions throughout the completion of this work, and Mihalis Yannakakis for his comments on an early draft of this manuscript. The author would also like to thank the anonymous referees for suggesting various improvements to this paper, in particular the addition of Corollary 3, the connection to hardness extractors and the GGM generator, and the simplification of Lemma 2.

REFERENCES

[1] J. ALMAN AND L. CHEN, *Efficient construction of rigid matrices using an NP oracle*, in 2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS), 2019, pp. 1034–1055.

[2] B. Barak, A. Rao, R. Shaltiel, and A. Wigderson, *2-source dispersers for sub-polynomial entropy and ramsey graphs beating the frankl-wilson construction*, in Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing, STOC '06, New York, NY, USA, 2006, Association for Computing Machinery, p. 671–680.

[3] J. Buresh-Oppenheim and R. Santhanam, *Making hard problems harder*, 21st Annual IEEE Conference on Computational Complexity (CCC'06), (2006), pp. 15 pp.–87.

[4] B. Chor and O. Goldreich, *Unbiased bits from sources of weak randomness and probabilistic communication complexity*, in 26th Annual Symposium on Foundations of Computer Science (sfcs 1985), 1985, pp. 429–442.

[5] P. Erdös, *Some remarks on the theory of graphs*, Bulletin of the American Mathematical Society, 53 (1947), pp. 292–294.

[6] O. Goldreich, S. Goldwasser, and S. Micali, *How to construct random functions*, J. ACM, 33 (1986), p. 792–807.

[7] A. Golovnev, R. Ilango, R. Impagliazzo, V. Kabanets, A. Kolokolova, and A. Tal, *Ac0[p] lower bounds against mcsp via the coin problem*, Electron. Colloquium Comput. Complex., 26 (2019), p. 18.

[8] J. M. Hitchcock and A. Pavan, *On the np-completeness of the minimum circuit size problem*, in FSTTCS, 2015.

[9] R. Impagliazzo and A. Wigderson, *P = BPP if E requires exponential circuits: Derandomizing the XOR lemma*, in Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, STOC '97, New York, NY, USA, 1997, Association for Computing Machinery, p. 220–229.

[10] E. Jeřábek, *Dual weak pigeonhole principle, boolean complexity, and derandomization*, Annals of Pure and Applied Logic, 129 (2004), pp. 1–37.

[11] V. Kabanets and J.-Y. Cai, *Circuit minimization problem*, in Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, STOC '00, New York, NY, USA, 2000, Association for Computing Machinery, p. 73–79.

[12] R. Kleinberg, O. Korten, D. Mitropolsky, and C. Papadimitriou, *Total Functions in the Polynomial Hierarchy*, in 12th Innovations in Theoretical Computer Science Conference (ITCS 2021), J. R. Lee, ed., vol. 185 of Leibniz International Proceedings in Informatics (LIPIcs), Dagstuhl, Germany, 2021, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, pp. 44:1–44:18.

[13] A. R. Klivans and D. van Melkebeek, *Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses*, SIAM Journal on Computing, 31 (2002), pp. 1501–1526.

[14] P. Miltersen and N. Vinodchandran, *Derandomizing arthur-merlin games using hitting sets*, in 40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039), 1999, pp. 71–80.

[15] N. Nisan and A. Wigderson, *Hardness vs randomness*, Journal of Computer and System Sciences, 49 (1994), pp. 149–167.

[16] C. H. Papadimitriou, *On the complexity of the parity argument and other inefficient proofs of existence*, Journal of Computer and System Sciences, 48 (1994), pp. 498 – 532.

[17] J. Paris, A. Wilkie, and A. R. Woods, *Provability of the pigeonhole principle and the existence of infinitely many primes*, J. Symb. Log., 53 (1988), pp. 1235–1244.

[18] A. A. Razborov and S. Rudich, *Natural proofs*, Journal of Computer and System Sciences, 55 (1997), pp. 24–35.

[19] R. Santhanam, *The complexity of explicit constructions*, Theory of Computing Systems, 51 (2012), pp. 297–312. Copyright - Springer Science+Business Media, LLC 2012; Document feature - ; Equations; Last updated - 2020-11-18; CODEN - TCSYFI.

[20] ———, *Pseudorandomness and the minimum circuit size problem*, in 11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA, T. Vidick, ed., vol. 151 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, pp. 68:1–68:26.

[21] C. E. Shannon, *The synthesis of two-terminal switching circuits*, The Bell System Technical Journal, 28 (1949), pp. 59–98.

[22] S. P. Vadhan, *Pseudorandomness*, vol. 7, Now Delft, 2012.

[23] L. G. Valiant, *Graph-theoretic arguments in low-level complexity*, in Mathematical Foundations of Computer Science 1977, J. Gruska, ed., Berlin, Heidelberg, 1977, Springer Berlin Heidelberg, pp. 162–176.

[24] A. C. Yao, *Theory and application of trapdoor functions*, in 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982), 1982, pp. 80–91.