Links Between Complexity Theory and Constrained Block Coding

Larry Stockmeyer and Dharmendra S. Modha, *Member, IEEE*

***Abstract*—The goal of this paper is to establish links between computational complexity theory and the theory and practice of constrained block coding. In particular, the complexities of several fundamental problems in constrained block coding are precisely classified in terms of the existing complexity-theoretic structure. One type of problem studied is that of designing encoder and decoder circuits using minimum or approximately minimum hardware; for our purposes, an "input" to this problem is i) a deterministic, irreducible finite-state transition diagram (abbreviated DIF) defining a set of constrained binary sequences, and ii) a desired rate $p : q$. Several of these minimum-encoder and minimum-decoder problems are shown to be NP-hard, and more interestingly some are shown to be complete in the second and third levels of the polynomial hierarchy. Another fundamental problem is that of computing the maximum rate of a block code; that is, given a DIF and a codeword length q, find the maximum p such that a rate $p : q$ block code exists for the constraint defined by the DIF. This problem is shown to be $\mathrm{NP}^{\#\mathrm{P}}$-complete. Although it is not known whether $\mathrm{NP}^{\#\mathrm{P}}$ contains problems of super-polynomial complexity, it lies "higher" in the complexity-class structure than NP in the sense that it is possible, given current knowledge, that $\mathrm{NP}^{\#\mathrm{P}}$ contains problems of super-polynomial complexity even if $\mathrm{P} = \mathrm{NP}$. Another question studied is whether maximum rate block codes can always be implemented by encoders and decoders of polynomial size. The answer to this question is shown to be closely related to whether the class $\#\mathrm{P}$ lies "lower" in the complexity-class structure than currently believed—a proof of either answer would have major implications in complexity theory.**

***Index Terms*—Block code, circuit size, complexity, constrained coding, decoder, encoder, $\#\mathrm{P}$, polynomial-time hierarchy, rate.**

I. Introduction

RELIABILITY of various storage devices and transmission channels, for example, magnetic/optical disk drives, tape drives, and electrical and fiber cables, can be substantially improved by matching the characteristics of the recorded or transmitted signals to those of the channel. Constrained codes, also known as modulation codes, are used to convert an arbitrary source signal to a coded signal that satisfies various desired channel constraints. For introductions to constraint coding, see, Immink [1], Marcus, Siegel, and Wolf [2], and Immink, Siegel, and Wolf [3].

Various constraints of practical interest are often specified using finite-state transition diagrams (FSTDs). An FSTD is a directed graph with edges labeled by 0 or 1; the vertices are thought of as states. An FSTD is *deterministic* if at every vertex of the graph, the edges directed out of the vertex include at most one edge labeled 0 and at most one edge labeled 1; otherwise, it is *nondeterministic*. Suppose we are given an FSTD, that is, a finite, directed, labeled graph. A finite binary word generated by following some walk on the graph from some (arbitrary) state to another (possibly the same, arbitrary) state is known as a constrained sequence. The set of all such constrained sequences is the constraint defined by the FSTD. Such constraints have been studied in systems theory, information and coding theory, computer science, and symbolic dynamics under the respective names of dynamical systems, discrete noiseless channels, regular languages, and sofic systems; see, Forney *et al.* [4] and Marcus [5].

These concepts are now illustrated by an example. A class of constraints that are useful in magnetic recording are the run-length-limited (RLL) constraints. For nonnegative integers d and k, the RLL(d, k) constraint contains all binary words such that, between every two consecutive 1's, there are at least d 0's but no more than k 0's. Fig. 1 shows a deterministic FSTD that defines the RLL$(1, 7)$ constraint.

Given a constraint, a fundamental problem is to design a high-rate, invertible mapping, known as the *encoder*, from arbitrary unconstrained source binary sequences into coded constrained binary sequences. Associated with an encoder is a *decoder* that takes constrained binary sequences produced by the encoder and recovers the original source sequence. Typically, the encoder and decoder take the form of synchronous finite-state machines. In particular, a *rate $p : q$ finite-state encoder* accepts an input block, a p-bit *dataword*, and generates a q-bit *codeword* depending upon the input block and the current internal state of the encoder. A *rate $p : q$ block encoder* is a static mapping from p-bit datawords to q-bit codewords; in other words, a block encoder is a finite-state encoder having only one state. Block encoders/decoders have the advantage of simplicity, as compared to general finite-state encoders/decoders. Another advantage is that when a block decoder tries to decode a q-bit block in which one or more bits are in error, the decoder recovers immediately at the next block; with finite-state decoders, errors can propagate indefinitely, and extra measures must be taken to avoid this. An advantage of finite-state encoders is that they can usually achieve larger rates than block encoders for particular con-

Manuscript received June 27, 2000; revised July 20, 2001. The material in this paper was presented in part at the IEEE International Symposium on Information Theory, Washington, DC, June, 2001.

The authors are with the IBM Research Division, Almaden Research Center, San Jose, CA 95120 USA (e-mail: stock@almaden.ibm.com; dmodha@us.ibm.com).

Communicated by R. Roth, Associate Editor for Coding Theory.

Publisher Item Identifier S 0018-9448(02)00059-7.

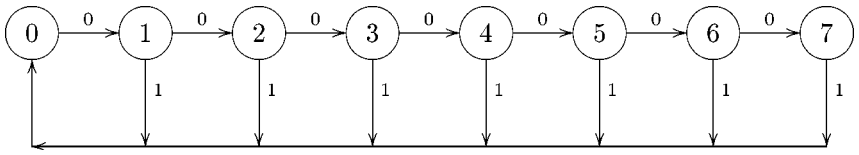

Fig. 1. A deterministic FSTD that defines the RLL$(1, 7)$ constraint.

straints and values of q.[1] Rate $p:q$ block encoders employ a set of 2^p q-bit codewords to encode the datawords; this set is called the *block codebook*. An important property of block encoders for a constraint $\mathcal{S}$ is that the codewords in any block codebook must be $\mathcal{S}$*-concatenable*, that is, they can be freely concatenated with each other without violating the constraint $\mathcal{S}$. For example, $w_1 = 0010000$ and $w_2 = 0000101$ both satisfy the RLL$(1, 7)$ constraint, but their concatenation $w_1w_2 = 00100000000101$ does not because it has a run of eight zeros. No block encoder for RLL$(1, 7)$ can employ both w_1 and w_2 as codewords, say as codewords for datawords d_1 and d_2, respectively, because the source sequence is completely unconstrained and so can contain d_1d_2.

Considering again the RLL$(1, 7)$ constraint, suppose that we have decided to use codewords of length $q = 6$. The question of the maximum rate that can be achieved by a block encoder depends on the maximum size of a RLL$(1, 7)$-concatenable set of q-bit codewords. In general, for an FSTD G defining the constraint $\mathcal{S}$ and a positive integer q, we let $MaxConcat(G, q)$ be the maximum size of a $\mathcal{S}$-concatenable set of q-bit codewords. It is not hard to see that if G is the FSTD in Fig. 1, then $MaxConcat(G, 6) = 10$. An example set of codewords is

$$\begin{array}{lllll} 010101 & 010010 & 010100 & 010001 & 001010 \\ 001001 & 001000 & 000101 & 000100 & 000010. \end{array} \quad (1)$$

Because each of the 2^p datawords must be mapped to a unique codeword, $p = 3$ is the largest possible, and the maximum block rate for the RLL$(1, 7)$ constraint with $q = 6$ is rate $3:6$.

One of the fundamental questions we study (Section IV) is the computational complexity of computing the maximum block rate $MaxBlkRate(G, q)$, where both the deterministic FSTD G and the codeword length q are inputs to the function; that is, $MaxBlkRate(G, q)$ equals p/q where p is the largest integer satisfying $2^p \leq MaxConcat(G, q)$. We give a precise classification of this function in terms of the existing complexity-theoretic structure by showing that the related decision problem (given G, q and an integer K, decide whether $MaxBlkRate(G, q) \geq K/q$) is NP$^{\#\mathrm{P}}$-complete. The complexity class $\#\mathrm{P}$ of functions was introduced by Valiant [7] to capture the complexity of certain "counting" problems, such as counting the number of satisfying assignments of a given Boolean formula or counting the number of Hamiltonian cycles in a given graph. The class NP$^{\#\mathrm{P}}$ is precisely defined later; for now, very roughly, it corresponds to the well-known class NP combined with an "oracle" for a function in $\#\mathrm{P}$ such as the two examples just mentioned. As will be explained later, the class NP$^{\#\mathrm{P}}$ lies fairly "high" in the complexity class structure; in particular, it is possible, given current knowledge, that NP$^{\#\mathrm{P}}$ contains problems that cannot be solved in polynomial time even if $\mathrm{P} = \mathrm{NP}$. We also obtain related completeness results for the functions $MaxBlkRate$ and $MaxConcat$.

The other types of problems we consider (Section III) are involved in the design of block encoders and decoders for a given deterministic FSTD G, dataword length p, and codeword length q, where we assume that a block encoder is possible, that is, $MaxConcat(G, q) \geq 2^p$. Specifically, we focus attention on the computational complexity of designing encoders and decoders that can be implemented using a small amount of hardware, that is, using circuits of minimum or approximately minimum size. Minimizing hardware complexity is important since such encoder/decoders are typically implemented in mass-market magnetic/optical recording systems. For a recent heuristic algorithm for finding encoder/decoders that can be implemented in a small amount of hardware, see Modha and Marcus [8]. For examples of various "practical" block encoder/decoders, see, Eggenberger and Patel [9], Marcus, Patel, and Siegel [10], and Galbraith [11]. A reason why one would expect these problems to be computationally difficult is that there are many choices to be made in designing an encoder/decoder for a given G, p, and q, where G defines $\mathcal{S}$. First, we can choose a block codebook $\mathcal{C}$, any $\mathcal{S}$-concatenable set of 2^p codewords of length q. We have extra freedom if $MaxConcat(G, q)$ is larger than 2^p (it could be almost twice as large even if the rate is maximum for this q). Second, we can choose the encoder mapping to be any one-to-one mapping from the 2^p datawords to the codewords in $\mathcal{C}$. Third, we can choose circuits that implement the encoder and associated decoder mappings. Although each of these three steps can be considered separately, in this paper we consider the total problem: given (G, p, q) find a small encoder (or decoder or encoder/decoder) circuit for (G, p, q). We give several results classifying these problems in terms of the polynomial-time hierarchy (see Stockmeyer [12]). The classes of this hierarchy extend the class NP. Intuitively, problems in NP can be defined using existential quantification over a "large" (exponentially sized) set of "short" (polynomially sized) objects. For example, a Boolean formula is satisfiable iff there exists a satisfying truth assignment to its variables; if the formula has n variables, the existential quantification is over the set of 2^n truth assignments, each of size n. If a problem is NP-complete, then intuitively an existential quantification over a large set is *inherent* in the problem. The classes of the polynomial-time hierarchy permit alternations of quantifiers over large sets. For example, the class Σ_2^p permits $\exists\forall$, the class Σ_3^p permits $\exists\forall\exists$, and so on. If a problem is Σ_3^p-complete, for example, then intuitively the quantifier structure $\exists\forall\exists$ is inherent in the problem. It is perhaps not surprising that minimizing the size of encoder/decoder circuits is NP-hard. What is more interesting is that we are able to precisely determine the inherent quantifier structure of

[1] Although, in the limit of large p and q, the maximum rate achievable by block encoders and general finite-state encoders is the same, namely, the Shannon capacity of the constrained system; see, for example, [6].

some of these problems. Moreover, our result that minimizing decoder size has the inherent structure $\exists\forall\exists$ means that this problem is "harder" than the circuit minimization problem, which is to take an arbitrary circuit C and output a circuit of minimum size that computes the same function as C. The circuit minimization problem can be defined using just $\exists\forall$. A more formal statement of this is that, if we assume that the polynomial-time hierarchy does not collapse to any fixed level, then the decoder minimization problem cannot be solved in polynomial time, even if we have an "oracle" for circuit minimization that instantly minimizes any circuit that we give it.

We also show that some of these problems are even hard to solve approximately. For example, assuming that $\mathrm{P} \neq \mathrm{NP}$, there is no polynomial-time algorithm that finds encoder circuits whose size is guaranteed to be within a factor $q^{1/5}$ of the optimum.

In Section V, we consider the question of whether maximum rate block codes can always be implemented by circuits of polynomial size. We do not answer this question, but we show that the answer is closely related to the position of the class $\#\mathrm{P}$ in the complexity class structure. The answer is "yes" if $\#\mathrm{P}$ lies lower in the structure than is currently believed. A definitive answer of either "yes" or "no" would imply a major breakthrough in computational complexity theory. Although the evidence from complexity theory is that, in general, the maximum rate cannot be achieved by polynomial-size encoders, we show that if the maximum rate is $p : q$ for a given G and q, the (finite) memory of G is at most q, and G contains at most $2^{p/2}$ states, then a polynomial-size encoder/decoder can always achieve rate $p : 2q$ (that is, half the maximum rate).

Although we concentrate on the case of block codes, some of our results apply also to the problem of designing encoders/decoders for individual states in finite-state encoders; this is described in Section VI. In this paper, we assume that all constraints are defined by *deterministic* FSTDs; constraints of practical interest are usually defined in this way. Because our main results show computational hardness of problems, our results become stronger by requiring FSTDs to be deterministic.

We close this section by mentioning some related work. To our knowledge, the only result on the computational complexity of a problem arising in constrained coding is the following. For finite-state encoders, as noted above, the encoder is a function of the input block and the current internal state of the encoder. In general, to decode a sequence encoded using a finite-state encoder requires a finite-state decoder. However, by appropriately choosing the mapping from datawords to codewords at each state of the finite-state encoder, it is sometimes possible to obtain a block decoder that is state-independent. Such a property is extremely desirable for limiting error propagation. Ashley, Karabed, and Siegel [13] have shown that the problem of determining the existence of a block decoder for a given rate $p : q$, $p > 1$, finite-state encoder is NP-complete. While our work focuses on different problems than [13], both are concerned with desirable properties of judicious choices of dataword-to-codeword assignments. We note, in passing, several papers on the general subject of the computational complexity of problems arising in error-correcting coding. These problems, such as the minimum-distance decoding of linear codes, are of a different nature than the problems that we study here. The first results of this type were by Berlekamp, McEliece, and van Tilborg [14], and many other examples are given by Vardy [15] and Sudan [16]. Horn and Kschischang [17] and Kschischang and Sorokine [18] have shown NP-completeness of certain problems of minimizing the trellis complexity of an error-correcting code.

Regarding related work in complexity theory, one of the goals of computational complexity theory is to bound or classify the complexities of problems that have some practical or mathematical motivation outside of complexity theory. Hundreds of such "natural" problems are now known to be NP-complete. There has been less success in identifying such problems complete in higher complexity classes, such as Σ_2^p, Σ_3^p, and $\mathrm{NP}^{\#\mathrm{P}}$, although some results are known. A recent result of Umans [19] is that the minimization problem for Boolean formulas in disjunctive normal form is Σ_2^p-complete, answering a question that had remained open for over 20 years. Additional examples of Σ_2^p-complete problems and a survey of previous results of this type are given by Schaefer [20]. Schaefer [21] shows that computing the Vapnik–Červonenkis dimension of a given concept class is Σ_3^p-complete provided that the concept class is represented implicitly in a "succinct" form. Many of the known Σ_2^p- and Σ_3^p-completeness results depend upon representing sets in certain succinct ways, where the size of the representation of a set can be exponentially smaller than the set itself. Our results also rely upon succinct representation, namely, using an FSTD to represent a set of constrained sequences. Wagner [22] has given a number of completeness results in higher complexity classes by representing multisets of integers in various succinct ways. Among these results are to decide for a given (succinctly represented) multiset of integers and a number K: 1) whether the set contains an interval of size at least K—this is Σ_3^p-complete; and 2) whether the set contains an element of multiplicity at least K—this is $\mathrm{NP}^{\#\mathrm{P}}$-complete. The succinct representations used in these two results are either integer expressions [12], [23] or the general hierarchic input language [24]. More recently, Mundhenk, Goldsmith, Lusena, and Allender [25] have shown that certain problems concerning Markov decision processes are $\mathrm{NP}^{\#\mathrm{P}}$-complete. They note that natural $\mathrm{NP}^{\#\mathrm{P}}$-complete problems are very rare, and they suggest further study of this class. The Markov decision process problems appear quite different than the problem of maximum block rate, although their common $\mathrm{NP}^{\#\mathrm{P}}$-completeness provides a link between them.

We assume that the reader is familiar with the concept of polynomial-time computation, and is somewhat familiar with other rudiments of the theory of NP-complete problems such as the complexity class NP and NP-complete problems. The definitions of NP and NP-complete problems are reviewed in Section II-D. The basic reference on the subject is Garey and Johnson [26]; see also Johnson [27] for a more recent reference on the subject.

Section II contains definitions of concepts from constrained coding theory and complexity theory that are used in the paper. Section III contains our results on the complexity of minimizing the hardware size of encoders and decoders. In Section IV, we discuss some additional complexity classes and give our re-

sults on the complexity of finding the maximum block rate. In Section V, we consider the question of whether maximum-rate block codes can be implemented by encoder/decoders of polynomial size. Section VI contains some conclusions and suggestions for further research.

II. Preliminaries

A. Coding Theory

An r-*bit word* is an element of $\{0, 1\}^r$. If y is an r-bit word, let $y_1, \ldots, y_r \in \{0, 1\}$ denote the bits of y, so $y = y_1 y_2 \cdots y_r$. Let $\{0, 1\}^+$ denote the set of r-bit words for all $r \geq 1$. Let $|y|$ denote the length of y. If S is a finite set, $|S|$ denotes the cardinality of S. $\boldsymbol{N}$ denotes the set of nonnegative integers, and $\boldsymbol{N}^+$ denotes the set of positive integers. All logarithms are to the base 2. For $y \in \{0, 1\}^r$, let $\mathrm{num}(y)$ denote the nonnegative integer whose r-bit binary representation is y; for example, $\mathrm{num}(000101) = 5$. Restricting attention to the binary alphabet strengthens our hardness results; however, all of our results hold if $\{0, 1\}$ is replaced by any finite alphabet A, $|A| \geq 2$, with trivial modifications needed because in general A^r contains $|A|^r$ words.

Definition 2.1:

1) A *finite-state transition diagram* (*FSTD*) has the form $G = (V, E)$ where V is a finite set (*the states*), and $E \subseteq V \times V \times \{0, 1\}$ (*the* $\{0, 1\}$*-labeled edges*). We say that the edge $e = (s, t, l) \in E$ has the start state s, the terminal state t, and the label l.
2) Let $|G|$ denote the number of states of G, that is, $|G| = |V|$.
3) The FSTD $G = (V, E)$ is *deterministic* if for every state $s \in V$ there is at most one $t \in V$ such that $(s, t, 0) \in E$ and at most one $t' \in V$ such that $(s, t', 1) \in E$.
4) For $s \in V$, let $\mathcal{S}(G, s)$ be the set of finite words *generated* from state s by following paths starting at s; that is, $y = y_1 \cdots y_r \in \mathcal{S}(G, s)$ iff there are states $s_0, \ldots, s_r \in V$ such that $s_0 = s$ and $(s_{i-1}, s_i, y_i) \in E$ for $1 \leq i \leq r$. If G is deterministic, then there is at most one such sequence $s_0, \ldots, s_r$ for each $s = s_0$; in this case, we define $\delta_G(s, y) = s_r$ (if G is clear from context, we write simply $\delta(s, y)$).
5) The *constraint defined by* G, which we denote $\mathcal{S}(G)$, is the set of words generated from all states of G; that is,

$$\mathcal{S}(G) = \bigcup_{s \in V} \mathcal{S}(G, s).$$

For $q \in \boldsymbol{N}^+$, let $\mathcal{S}_q(G)$ denote the set of words y such that $y \in \mathcal{S}(G)$ and $|y| = q$. Elements of $\mathcal{S}(G)$ are known as *codewords*, and elements of $\mathcal{S}_q(G)$ are known as q*-bit codewords*.

6) The FSTD G is *irreducible* if every state can be reached from every state; that is, for each $s, t \in V$, there exists a y such that y is generated from s and $\delta(s, y) = t$.

In this paper we limit attention to *deterministic* and *irreducible* finite-state transition diagrams. We use *DIF* to abbreviate one of these objects.

As described in the Introduction, for block codes we are interested only in codebooks $\mathcal{C}$ where an arbitrary concatenation of words in $\mathcal{C}$ satisfies the given constraint.

Definition 2.2:

1) Let $\mathcal{C}$ and $\mathcal{S}$ be subsets of $\{0, 1\}^+$. We say that $\mathcal{C}$ is $\mathcal{S}$*-concatenable* if, for all $k \geq 1$ and all $w_1, w_2, \ldots, w_k \in \mathcal{C}$, the concatenation $w_1 w_2 \cdots w_k$ belongs to $\mathcal{S}$.
2) For a DIF G and $q \in \boldsymbol{N}^+$, define $MaxConcat(G, q)$ to be the maximum of $|\mathcal{C}|$ over all $\mathcal{C} \subseteq \{0, 1\}^q$ such that $\mathcal{C}$ is $\mathcal{S}(G)$-concatenable.
3) The *maximum block rate for* (G, q) is defined as

$$MaxBlkRate(G, q) = \lfloor \log MaxConcat(G, q) \rfloor / q.$$

Definition 2.3:

1) Let G be a DIF and $p, q \in \boldsymbol{N}^+$. The set $\mathcal{C}$ is a *block codebook for* (G, p, q) if $\mathcal{C} \subseteq \{0, 1\}^q$, $\mathcal{C}$ is $\mathcal{S}(G)$-concatenable, and $|\mathcal{C}| = 2^p$.
2) (G, p, q) is *block feasible* iff $MaxConcat(G, q) \geq 2^p$.

Obviously, there exists a block codebook for (G, p, q) iff (G, p, q) is block feasible.

Many FSTDs G of practical interest have the "finite memory" property, which means that there is an integer m such that each word $y \in \mathcal{S}(G)$ with $|y| \geq m$ leads to a unique terminal state regardless of the start state. This can be defined as follows [2], [28].

Definition 2.4: The deterministic FSTD $G = (V, E)$ has *memory at most* m if, for each $y \in \mathcal{S}(G)$ with $|y| \geq m$, there is a unique $t \in V$ such that, if $s \in V$ and y is generated from s, then $\delta(s, y) = t$. G has *finite memory* if there exists an m such that G has memory at most m, and in this case the *memory of* G is the smallest such m. Let $\mathcal{S} \subseteq \{0, 1\}^+$. $\mathcal{S}$ has *finite memory* if there is a deterministic FSTD G and an m such that $\mathcal{S} = \mathcal{S}(G)$ and G has memory m; in this case, the *memory of* $\mathcal{S}$ is the smallest such m (over all deterministic FSTDs G with $\mathcal{S} = \mathcal{S}(G)$). G (resp., $\mathcal{S}$) has *infinite memory* if G (resp., $\mathcal{S}$) does not have finite memory.

An example of a practically relevant constraint having infinite memory is the *charge-k constraint* where $k \in \boldsymbol{N}^+$: the alphabet is $\{+1, -1\}$, and the word $y_1 \cdots y_r \in \{+1, -1\}^+$ satisfies this constraint if, for all $1 \leq j \leq r$, the jth prefix sum satisfies $-k \leq \sum_{i=1}^{j} y_i \leq k$ (see, [2, Sec. IV]). It is possible that $\mathcal{S} = \mathcal{S}(G)$ where $\mathcal{S}$ has finite memory and G has infinite memory. In this case, there is some other G' having finite memory and $\mathcal{S} = \mathcal{S}(G')$.

It is known that the memory of $\mathcal{S}(G)$ and G can be found in polynomial time, and if $\mathcal{S}(G)$ has finite memory m then $m \leq |G|(|G| - 1)/2$ (see [6, Sec. 2.7.3]). The following is stated without proof.

Proposition 2.1: There is a polynomial-time algorithm that decides for any DIF input G whether or not $\mathcal{S}(G)$ (resp., G) has

finite memory. If the answer is "yes," the algorithm outputs the memory of $\mathcal{S}(G)$ (resp., G).

To obtain several of our results, we need a representation of a $\mathcal{S}(G)$-concatenable set $\mathcal{C} \subseteq \{0, 1\}^q$, where the size of the representation is polynomially bounded in $|G|$ and q. (The representation that lists all the words in $\mathcal{C}$ typically has size exponential in q.) The following lemma is sufficient for our purposes, and it might have other applications. This result was proved by Freiman and Wyner [28] in the case that $\mathcal{S}(G)$ has finite memory m and $q \geq m$. We prove it in greater generality because we want our upper bounds to hold also for arbitrary q and for G with (possibly) infinite memory.

Definition 2.5: Let $G = (V, E)$ be a DIF and let $\tau \subseteq V$ with τ nonempty. Define $\mathcal{T}_q(\tau)$ to be the set of words y of length q such that, for all $t \in \tau$, y is generated from t and $\delta(t, y) \in \tau$. Also define $\mathcal{T}_q(\emptyset) = \emptyset$.

Note that $\mathcal{T}_q(\tau)$ is $\mathcal{S}(G)$-concatenable for each $q \geq 1$ and $\tau \subseteq V$. $\mathcal{T}_q(\tau)$ may be empty even if τ is nonempty.

Lemma 2.2: Let $\mathcal{C} \subseteq \{0, 1\}^q$ and let $G = (V, E)$ be a DIF. Then $\mathcal{C}$ is $\mathcal{S}(G)$-concatenable iff there exists a $\tau \subseteq V$ such that $\mathcal{C} \subseteq \mathcal{T}_q(\tau)$.

Proof: The "if" direction is immediate from the fact, noted above, that $\mathcal{T}_q(\tau)$ is always $\mathcal{S}(G)$-concatenable.

To prove the "only if" direction, assume that $\mathcal{C}$ is $\mathcal{S}(G)$-concatenable. If $\mathcal{C} = \emptyset$ then take $\tau = \emptyset$. For the rest of the proof, assume $\mathcal{C} \neq \emptyset$. Define a new FSTD G' where the set V' of "superstates" is the set of subsets of V (including the whole set V and the empty set) and the "symbols" are the words in $\mathcal{C}$. Fix a superstate $\sigma_1 \in V'$ and a $y \in \mathcal{C}$; then following the symbol y from σ_1 goes to superstate $\sigma_2 = \{\delta(v, y) \mid v \in \sigma_1 \text{ and } y \in \mathcal{S}(G, v)\}$ (note that "$y \in \mathcal{S}(G, v)$" defines a set of states v, because y is fixed). For example, if y is generated by no state in σ_1, then $\sigma_2 = \emptyset$. Note that G' is deterministic. (G' might be reducible; this is not a problem because we are interested only in the superstates in the set $\mathcal{R}$ defined next.)

Let $\mathcal{R}$ be the set of superstates that are reachable from superstate V in an arbitrary number of steps. We first claim that the empty superstate is not in $\mathcal{R}$. Because if there is a word $y_1 y_2 \cdots y_k$ that leads from superstate V to superstate $\emptyset$, where $y_1, \ldots, y_k \in \mathcal{C}$, then $y_1 y_2 \cdots y_k \notin \mathcal{S}(G)$, contradicting the assumption that $\mathcal{C}$ is $\mathcal{S}(G)$-concatenable.

Let c be the minimum cardinality of a superstate in $\mathcal{R}$. By the claim above, $c \geq 1$. Define τ as the union of all superstates $\sigma \in \mathcal{R}$ such that $|\sigma| = c$. It is clear that τ is nonempty because there is at least one $\sigma \in \mathcal{R}$ with $|\sigma| = c$, and $c \geq 1$. The proof is complete if we show, for all $y \in \mathcal{C}$ and all $t \in \tau$, that y is generated from t and $\delta(t, y) \in \tau$. Suppose for contradiction that there is a $y \in \mathcal{C}$ and $t \in \tau$ such that either y is not generated from t or $\delta(t, y) \notin \tau$. Let σ_1 be a superstate with $\sigma_1 \in \mathcal{R}$, $t \in \sigma_1$, and $|\sigma_1| = c$; at least one such σ_1 must exist by definition of τ. Say first that y is generated from t, but $\delta(t, y) \notin \tau$. Let σ_2 be the superstate reached by following y from σ_1, so $\sigma_2 \in \mathcal{R}$ and $|\sigma_2| \leq |\sigma_1|$, by definition of G' and the fact that G is deterministic. If $|\sigma_2| < |\sigma_1| = c$, this contradicts the minimality of c. If $|\sigma_2| = |\sigma_1| = c$, then $\sigma_2 \subseteq \tau$ by definition of τ; this is a contradiction because $\delta(t, y) \in \sigma_2$ but $\delta(t, y) \notin \tau$. Similarly, if y is not generated from t, then following y from σ_1 goes to a state $\sigma_2 \in \mathcal{R}$ with $|\sigma_2| < c$ because $t \in \sigma_1$ and G is deterministic. □

Say that $\mathcal{C} \subseteq \{0, 1\}^q$ is *maximal $\mathcal{S}$-concatenable* if $\mathcal{C}$ is $\mathcal{S}$-concatenable and $\mathcal{C} \cup \{y\}$ is not $\mathcal{S}$-concatenable for all $y \in \{0, 1\}^q - \mathcal{C}$. It follows easily from Lemma 2.2 that if $\mathcal{C}$ is maximal $\mathcal{S}(G)$-concatenable then there exists a τ such that $\mathcal{C} = \mathcal{T}_q(\tau)$. In particular, if $\mathcal{C}$ is $\mathcal{S}(G)$-concatenable and $|\mathcal{C}| = MaxConcat(G, q)$, then there exists a τ such that $\mathcal{C} = \mathcal{T}_q(\tau)$. For example, if G is the DIF in Fig. 1 and $\mathcal{C}$ is the set of codewords in (1), then $\mathcal{C} = \mathcal{T}_6(\{0, 1, 2, 3\})$.

A question raised by Lemma 2.2 is the difficulty of finding a "maximum-rate τ." Given G and q, find a τ such that

$$\lfloor \log |\mathcal{T}_q(\tau)| \rfloor / q = MaxBlkRate(G, q).$$

We show in Theorem 4.2 that if there is a polynomial-time algorithm for finding a maximum-rate τ, then two apparently different complexity classes (defined in Section IV) are actually the same.

B. Encoders, Decoders, and Circuits

One of the problems of interest in this paper is the following: the problem is first stated informally here, and more formally in Section II-C. Given a DIF G, the length p of datawords and the length q of codewords where $p \leq q$ and (G, p, q) is block feasible, find a block codebook $\mathcal{C}$ for (G, p, q) and find functions $\mathcal{E}$, the (block) encoder, and $\mathcal{D}$, the (block) decoder, such that $\mathcal{E}$ maps $\{0, 1\}^p$ one-to-one onto $\mathcal{C}$, and $\mathcal{D} = \mathcal{E}^{-1}$ on $\mathcal{C}$. In addition, we want the functions $\mathcal{E}$ and $\mathcal{D}$ to be efficiently computable, where efficiency is measured by the number of gates in a logic circuit that computes the function. As mentioned in the Introduction, in this paper we consider the total problem of finding small encoder and decoder circuits given G, p, q, without necessarily explicitly finding $\mathcal{C}$ and $\mathcal{E}$ as intermediate steps (of course, $\mathcal{C}$ and $\mathcal{E}$ are defined implicitly by the encoder circuit). We also consider problems of finding a small encoder circuit regardless of the size of the decoder circuit, and *vice versa*.

Definition 2.6:

1) Let $p, q \in \boldsymbol{N}^+$ and let $\mathcal{C} \subseteq \{0, 1\}^q$ with $|\mathcal{C}| = 2^p$. A function $\mathcal{E}$ is an *encoder* (resp., $\mathcal{D}$ is a *decoder*) *for* $\mathcal{C}$ if $\mathcal{E}$ is a one-to-one function from $\{0, 1\}^p$ onto $\mathcal{C}$ (resp., $\mathcal{D}$ is a one-to-one function from $\mathcal{C}$ onto $\{0, 1\}^p$).
2) Let G be a DIF and let $p, q \in \boldsymbol{N}^+$ be such that (G, p, q) is block feasible. A function $\mathcal{E}$ is an *encoder* (resp., $\mathcal{D}$ is a *decoder*) *for* (G, p, q) if there is a block codebook $\mathcal{C}$ for (G, p, q) such that $\mathcal{E}$ is an encoder (resp., $\mathcal{D}$ is a decoder) for $\mathcal{C}$. A pair $(\mathcal{E}, \mathcal{D})$ is an *encoder/decoder for* (G, p, q) (resp., $\mathcal{C}$) if $\mathcal{E}$ is an encoder for (G, p, q) (resp., $\mathcal{C}$) and $\mathcal{D} = \mathcal{E}^{-1}$, that is, $\mathcal{D}(\mathcal{E}(x)) = x$ for all $x \in \{0, 1\}^p$.

Our measure of the complexity of an encoder or decoder function is the minimum number of computation steps (or "gates") in a logic circuit that computes the function. We use a standard definition of a Boolean circuit containing 2-ary gates that belong to a specified basis, as in, for example, [29], [30]. If the

circuit computes a function of n Boolean inputs $x_1, \ldots, x_n$, then starting with the inputs and the Boolean constants 0 and 1, the circuit can compute new functions by applying a function in the basis to any two previously computed functions.

Definition 2.7:

1) Let Ω (the basis) be a set of 2-ary Boolean functions (that is, $\Omega \subseteq \{\beta \,|\, \beta\colon \{0,1\}^2 \to \{0,1\}\}$) and let n, m be positive integers. An *n-input m-output Ω-circuit* U consists of two parts: i) a sequence $\sigma_1, \sigma_2, \ldots, \sigma_{n+2+z}$ of *steps* for some integer $z \geq 0$, where, for $n+2 < k \leq n+2+z$, the step σ_k is a triple (β, i, j) for some $\beta \in \Omega$ and $1 \leq i, j < k$; and ii) an *output mapping* ϕ: $\{1, 2, \ldots, m\} \to \{1, 2, \ldots, n+2+z\}$.
2) For each k with $1 \leq k \leq n+2+z$ define the function $\gamma_k\colon \{0,1\}^n \to \{0,1\}$ computed at step σ_k inductively as follows: for $1 \leq k \leq n$, γ_k is the kth projection function, $\gamma_k(x_1, \ldots, x_n) = x_k$; γ_{n+1} (resp., γ_{n+2}) is the constant function 0 (resp., 1); and if $n+2 < k \leq n+2+z$ and $\sigma_k = (\beta, i, j)$, then $\gamma_k = \beta(\gamma_i, \gamma_j)$.
3) The *size* of the circuit U is z (that is, the number of gates). Let $|U|$ denote the size of U.
4) If $\mathcal{F}\colon \{0,1\}^n \to \{0,1\}^m$ and we write $\mathcal{F} = (f_1, \ldots, f_m)$ where each $f_i\colon \{0,1\}^n \to \{0,1\}$, then U *computes* $\mathcal{F}$ if $f_i = \gamma_{\phi(i)}$ for $1 \leq i \leq m$.
5) The *full basis* is $\{\beta \,|\, \beta\colon \{0,1\}^2 \to \{0,1\}\}$. A basis Ω is a *complete basis* if for all $n, m \geq 1$ and all $\mathcal{F}$: $\{0,1\}^n \to \{0,1\}^m$, there exists an Ω-circuit U such that U computes $\mathcal{F}$.
6) The Ω-circuit complexity of $\mathcal{F}$, denoted $\Xi_\Omega(\mathcal{F})$, is the minimum z such that there exists an Ω-circuit of size z that computes $\mathcal{F}$ (assuming that some Ω-circuit computes $\mathcal{F}$). Define $\Xi(\mathcal{F}) = \Xi_\Omega(\mathcal{F})$ where Ω is the full basis.

In practice, the choice of the basis Ω is dictated by the available technology. To simplify the notation, we state our results only for the full basis; however, the results and their proofs hold without modification for any complete basis Ω. This is true because in the proofs we need bounds on the sizes of circuits only to within a constant factor, and for any two complete bases Ω and Ω' there is a constant $c\ (= \max\{\Xi_\Omega(\beta) \,|\, \beta \in \Omega'\})$ such that $\Xi_{\Omega'}(\mathcal{F}) \leq c \cdot \Xi_\Omega(\mathcal{F})$ for all $\mathcal{F}$ [30, Sec. 1.3].

Encoder and decoder circuits for (G, p, q) are defined in the obvious way: $\hat{\mathcal{E}}$ is an *encoder circuit* for (G, p, q) if there is an encoder (mapping) $\mathcal{E}$ for (G, p, q) such that $\hat{\mathcal{E}}$ computes $\mathcal{E}$; *decoder circuits* $\hat{\mathcal{D}}$ and *encoder/decoder* pairs of circuits $(\hat{\mathcal{E}}, \hat{\mathcal{D}})$ are defined similarly.

C. The Problems

Minimization problems have the following general form: given an instance of the problem, find a solution of minimum cost. In the theory of NP-hard problems, an associated decision problem is usually considered, where a *decision problem* is a problem where each instance has a yes/no answer. Typically, this is done by defining the associated decision problem to have a number K (the "target") included in the instance, and the answer is "yes" or "no" depending on whether a solution of cost $\leq K$ exists or not. Assuming that the cost of a given solution can be computed in polynomial time (for example, the size of a given circuit can be found in polynomial time), a polynomial-time algorithm for the original minimization problem would immediately give a polynomial-time algorithm for the associated decision problem. In other words, if the decision problem is "hard" then so is the minimization problem.

A first attempt to define our problems as decision problems, say in the case that we want to minimize only the size of the encoder, would be to let the instances be of the form (G, p, q, K) and the answer to be "yes" iff there is an encoder $\mathcal{E}$ for (G, p, q) with $\Xi(\mathcal{E}) \leq K$. A difficulty with this definition is that the answer could be "no" simply because (G, p, q) is not block feasible, that is, $MaxConcat(G, q) < 2^p$. Therefore, if the problem turns out to be NP-hard, we do not know whether the hard part is to decide whether (G, p, q) is block feasible, or to decide if there is an encoder circuit of size $\leq K$ in the case that (G, p, q) is block feasible. A resolution to this would be immediate if the first part were decidable in polynomial time, for then the second part would have to be the hard part. However, one of our results in Section IV is that deciding whether (G, p, q) is block feasible has complexity even "higher" than NP in the complexity hierarchy. There is a standard way in complexity theory of dealing with this difficulty, namely, the "promise problem"; promise problems first appeared in [31] and their theory was further developed in [32]. In addition to the components "set of instances" and "question" of a decision problem, another component of a promise problem is a specified predicate M of the instances called the *promise*. The idea is that we are promised that the instance satisfies M, and the answer given by an algorithm for the promise problem does not have to be correct if the instance does not satisfy M. We can restate the encoder minimization problem as a promise problem as follows. The instances are again those of the form (G, p, q, K), the promise is that (G, p, q) is block feasible, and the question as before is whether there is an encoder $\mathcal{E}$ for (G, p, q) with $\Xi(\mathcal{E}) \leq K$. An algorithm solves this promise problem if it answers correctly on all instances that satisfy the promise, but we do not care what it answers on instances that do not satisfy the promise. Thus, the promise problem focuses on the problem of determining if there is an encoder circuit of size $\leq K$ when (G, p, q) is block feasible.

We now define the problems of interest as promise problems.

MINIMUM BLOCK ENCODER (resp., DECODER, ENCODER+DECODER)
Instance: (G, p, q, K) where G is a DIF and p, q, K are positive integers.
Promise: (G, p, q) is block feasible.
Question: Is there an encoder $\mathcal{E}$ (resp., a decoder $\mathcal{D}$, an encoder/decoder $(\mathcal{E}, \mathcal{D})$) for (G, p, q) with $\Xi(\mathcal{E}) \leq K$ (resp., $\Xi(\mathcal{D}) \leq K$, $\Xi(\mathcal{E}) + \Xi(\mathcal{D}) \leq K$)?

MINIMUM BLOCK ENCODER&DECODER
Instance: (G, p, q, K_1, K_2) where G is a DIF and p, q, K_1, K_2 are positive integers.
Promise: (G, p, q) is block feasible.
Question: Is there an encoder/decoder $(\mathcal{E}, \mathcal{D})$ for (G, p, q) such that $\Xi(\mathcal{E}) \leq K_1$ and $\Xi(\mathcal{D}) \leq K_2$?

In the sequel, we let MBE, MBD, MBE+D, MBE&D abbreviate MINIMUM BLOCK ENCODER, DECODER, ENCODER+DECODER, ENCODER&DECODER, respectively.

In these problems, the DIF defining the constraint is part of the instance. This allows the size of G to grow as p and q grow, and we make essential use of this in our proofs. One could also consider these problems with G fixed, and let only p, q, and K grow. As discussed in the concluding section, a proof that the problem with fixed G is NP-hard would have major implications in complexity theory.

We are also interested in the computational complexity of these problems when the instances are further restricted to the practically important cases that the rate p/q is at least a particular constant $R < 1$, and that G has finite memory (FM). This will be done definitionally by adding one or both of "rate $\geq R$" and "FM" in parentheses, and adding the corresponding restriction "$p/q \geq R$" and/or "G has finite memory" to the promise; for example, MBE (rate $\geq R$, FM) denotes the minimum block encoder problem with both restrictions. (Because both of these restrictions can be decided in polynomial time (see Proposition 2.1 for the latter), the restrictions could be added to the instance rather than the promise. But because we already have a promise problem, it is cleaner to add them to the promise.)

The complexity of these minimum encoder and decoder problems is the subject of Section III. Section IV considers the complexity of computing $MaxConcat(G, q)$ and the related problem of finding the maximum block rate for a given G and q.

Before formally describing the relevant complexity theory, we first discuss the issue of defining the "size" of an instance. This issue is important because the time complexity of an algorithm is given as a function of the size of the instance. We take the size of the instance (G, p, q, K), where $G = (V, E)$, to be $|V| + |E| + p + q + K$. Another choice would be to take the size of an integer to be the length of its binary representation. This would give a smaller size measure with $\log pqK$ in place of $p+q+K$. The problem with using binary representations is that an r-bit number can describe a integer of magnitude 2^r, so an instance of size z could ask a question about the existence of a circuit whose size and number of inputs and outputs is exponentially larger than z. If this problem turned out to be "hard," the hardness could be due to the fact that the question is about exponentially large objects. Defining the size of (G, p, q, K) as above avoids this possibility. In the terminology of Garey and Johnson [26, Sec. 4.2], our complexity results for these problems are in the "strong sense."

D. Computational Complexity Theory

In this section, we define some concepts from computational complexity theory that will be used in the paper.

Recall that for a decision problem, each instance has a yes/no answer. Formally, a decision problem is the set of instances that have the answer "yes," where each instance is represented as a word over a finite alphabet, which we may assume is $\{0, 1\}$. Representing each instance as a binary word is just for the sake of the formal definitions. The only detail of the representation that will concern us (recalling the discussion above about the size of an instance) is that, unless otherwise noted, the length of the representation of an integer z is proportional to z, rather than $\log z$. Thus, formally, a *decision problem* is some subset of $\{0, 1\}^+$. A complexity class is a class of decision problems, i.e., a class of subsets of $\{0, 1\}^+$.

1) The Polynomial-Time Hierarchy: We define a sequence of complexity classes that gives a "hierarchy" above NP. Recall that P denotes the class of decision problems that can be solved in polynomial time. The *polynomial-time hierarchy* was first defined in [33], and additional information can be found in [12], [34], and [27, Sec. 2.5]. Let $k \geq 1$ and let R: $(\{0, 1\}^+)^{k+1} \rightarrow \{0, 1\}$ denote a relation on $k+1$ binary-word arguments, which we denote $u, w_1, \ldots, w_k$. Call R a *ptime-relation* if R can be computed in time $P(|u| + |w_1| + \cdots + |w_k|)$ for some polynomial P.

The *polynomial-time hierarchy* (for short, the *polynomial hierarchy*) contains the complexity classes Σ_k^p for $k \geq 0$, where in particular $\Sigma_0^p = \mathrm{P}$. Let $A \subseteq \{0, 1\}^+$ be a decision problem and $k \geq 1$. Then A belongs to the complexity class Σ_k^p iff there is a ptime-relation R and a polynomial $Q(\cdot)$ such that, for all $u \in \{0, 1\}^+$

$$u \in A \text{ iff } (\exists\, w_1)(\forall\, w_2)(\exists\, w_3)\cdots \cdots(\{\exists, \forall\}w_k)[R(u, w_1, w_2, \ldots, w_k) = 1]$$

where the quantifiers alternate and where $w_1, w_2, \ldots, w_k$ range over all binary words of length $Q(|u|)$.

In particular, $\Sigma_1^p = \mathrm{NP}$. The complement $\overline{A}$ of the decision problem A has the same set of instances as A, and for all instances u, $u \in \overline{A}$ iff $u \notin A$. Let co-NP be the class of complements of decision problems in NP. Of the other classes, we will use Σ_2^p and Σ_3^p. Just as the instance u can represent a more complicated object, such as (G, p, q, K), the words $w_1, \ldots, w_k$ may also represent more complicated objects. To illustrate this, we use the definition to show that the decision problem, with instance an undirected graph H and question of whether H has a Hamiltonian cycle, belongs to $\Sigma_1^p = \mathrm{NP}$. In this case, u represents the graph H, and w_1 represents a permutation of the vertices of H. The ptime-relation R checks that w_1 represents a cyclic permutation that defines a Hamiltonian cycle in H. The Σ_k^p classes for $k > 1$ provide complexity-class homes for certain problems that are naturally defined using one or more alternations of quantifiers. Just as it is not known whether or not $\mathrm{P} = \mathrm{NP}$, it is not known whether or not $\Sigma_k^p = \Sigma_{k+1}^p$. However, it is known [12] that if $\mathrm{P} = \mathrm{NP}$ then $\mathrm{P} = \Sigma_j^p$ for all $j \geq 1$, and more generally that if $\Sigma_k^p = \Sigma_{k+1}^p$ then $\Sigma_k^p = \Sigma_j^p$ for all $j \geq k$. Therefore, the only possibilities are: i) the hierarchy is *strict*, meaning that $\Sigma_k^p \neq \Sigma_{k+1}^p$ for all $k \geq 0$; or ii) there exists an $\ell \geq 0$ such that the hierarchy *collapses to the ℓ-th level*, meaning that $\Sigma_\ell^p = \Sigma_j^p$ for all $j \geq \ell$. Strictness of the polynomial hierarchy has become a working hypothesis in complexity theory; evidence for the truth of a conjecture is often given by proving that the negation of the conjecture implies that the polynomial hierarchy collapses to some level. Results of this type are given later in Corollary 3.5. After extending the definitions to promise problems, we will show that the minimum encoder and decoder problems defined above belong to Σ_2^p or Σ_3^p.

Formally, a *promise problem* is a pair (A, M) where $A \subseteq \{0, 1\}^+$ and M: $\{0, 1\}^+ \rightarrow \{0, 1\}$ (M is the promise). A

solution of (A, M) is a decision problem $A' \subseteq \{0,1\}^+$ such that A and A' agree on the instances that satisfy the promise M, that is, for all $u \in \{0,1\}^+$ such that $M(u) = 1$ we have $u \in A$ iff $u \in A'$. Let $\mathcal{L}$ be a class of decision problems, that is, a class of subsets of $\{0,1\}^+$. The promise problem (A, M) belongs to $\mathcal{L}$ if there is a solution A' of (A, M) such that A' belongs to $\mathcal{L}$.

The following result can be thought of as giving "upper bounds" on the complexities of the circuit minimization problems of interest.

Theorem 2.3:

1) MBE, MBE+D, and MBE&D belong to Σ_2^p.
2) MBD belongs to Σ_3^p.

Proof:

1) We first show that MBE belongs to Σ_2^p. Recall that an instance for this problem is (G, p, q, K) where G is a DIF. Let $G = (V, E)$. We use the fact that U is an encoder for (G, p, q) iff U is a p-input q-output circuit such that

$$\mathrm{Range}(U) \stackrel{\mathrm{def}}{=} \{U(x) \,|\, x \in \{0,1\}^p\}$$

is $\mathcal{S}(G)$-concatenable, and $U(x) \neq U(x')$ for all $x, x' \in \{0,1\}^p$ with $x \neq x'$. By Lemma 2.2, $\mathrm{Range}(U)$ is $\mathcal{S}(G)$-concatenable iff there exists a $\tau \subseteq V$ such that $\mathrm{Range}(U) \subseteq \mathcal{T}_q(\tau)$. Using the notation above in the definition of Σ_k^p, u represents the instance (G, p, q, K), w_1 represents a circuit U and a set $\tau \subseteq V$, and w_2 represents a pair (x, x') of p-bit words. The relation $R(u, w_1, w_2)$ (that is, $R((G, p, q, K), (U, \tau), (x, x'))$) equals 1 iff: i) U is a p-input q-output circuit of size $\leq K$; ii) for all $t \in \tau$, $U(x)$ is generated from t and $\delta(t, U(x)) \in \tau$; and iii) if $x \neq x'$ then $U(x) \neq U(x')$. It is easy to see that all three conditions can be checked in polynomial time.

The proofs that MBE+D and MBE&D belong to Σ_2^p are similar. The only differences are: w_1 represents a set τ and a pair (U, U') of circuits (the encoder circuit and decoder circuit), and the relation R holds iff U and U' have the correct number of inputs and outputs, the sizes of U and U' satisfy the total bound K or the pair of bounds K_1 and K_2, $U'(U(x)) = x$, and ii) and iii) above hold.

2) We use the fact that (G, p, q, K) is a "yes" instance of MBD iff

A) $\exists$ a q-input p-output circuit U of size $\leq K$ (the decoder), such that $\forall\, x \in \{0,1\}^p$, $\exists$ a $y = y_x \in \{0,1\}^q$, such that $U(y_x) = x$ and $\{y_x \,|\, x \in \{0,1\}^p\}$ is $\mathcal{S}(G)$-concatenable.

Note that $x \neq x'$ implies $U(y_x) \neq U(y_{x'})$ implies $y_x \neq y_{x'}$. Using Lemma 2.2, we will argue that statement A) is equivalent to

B) $\exists$ a q-input p-output circuit U of size $\leq K$ and a set $\tau \subseteq V$, such that $\forall\, x \in \{0,1\}^p$, $\exists$ a $y = y_x \in \{0,1\}^q$, such that $U(y_x) = x$ and $y_x \in \mathcal{T}_q(\tau)$ (that is, for all $t \in \tau$, the word y_x is generated from t and $\delta(t, y_x) \in \tau$).

Because the condition $y \in \mathcal{T}_q(\tau)$ can be decided in polynomial time, B) fits the definition of Σ_3^p. We argue that A) is equivalent to B). Define

$$\mathrm{Range}(y) \stackrel{\mathrm{def}}{=} \{y_x \,|\, x \in \{0,1\}^p\}.$$

Say first that A) is true. Because $\mathrm{Range}(y)$ is $\mathcal{S}(G)$-concatenable, by Lemma 2.2 there is a $\tau \subseteq V$ such that $\mathrm{Range}(y) \subseteq \mathcal{T}_q(\tau)$. In B), choose such a τ and choose U as it is chosen in A). In B), choose $y = y_x$ as it is chosen in A). For each $x \in \{0,1\}^p$, $y_x \in \mathrm{Range}(y)$ by definition, so $y_x \in \mathcal{T}_q(\tau)$. Say now that B) is true. In A), choose U and for each x choose y_x as these are chosen in B). By B), $\mathrm{Range}(y) \subseteq \mathcal{T}_q(\tau)$, from which it is obvious that $\mathrm{Range}(y)$ is $\mathcal{S}(G)$-concatenable. □

Remark: Notice that in the proof of Theorem 2.3 we did not use the promise that (G, p, q) is block feasible. In particular, we have shown that the "general" minimum block encoder (GMBE) decision problem belongs to Σ_2^p, where (G, p, q, K) is a "yes" instance of GMBE iff there is an encoder circuit of size $\leq K$ for (G, p, q).

2) Reducibility, Hardness, and Completeness: We next give the definitions of a decision problem or a promise problem being "hard" or "complete" for a complexity class under a certain class of reductions. The only characteristic of a "complexity class" used in the definitions is that it is a class of decision problems, that is, a class of subsets of $\{0,1\}^+$. The most familiar class of reductions is the one containing all functions that are computable in polynomial time. A *polynomial-time transformation* is a function $f\colon \{0,1\}^+ \to \{0,1\}^+$ that is computable in polynomial time. For $A, B \subseteq \{0,1\}^+$, the notation $B \leq_m^p A$ means that there is a polynomial-time transformation f such that $u \in B$ iff $f(u) \in A$ for all $u \in \{0,1\}^+$. Let A be a decision problem and $\mathcal{L}$ be a complexity class. Then, A is *$\mathcal{L}$-hard under $\leq_m^p$*, written $\mathcal{L} \leq_m^p A$, if $B \leq_m^p A$ for all B in $\mathcal{L}$. In general, if $\preceq$ is a reflexive, transitive relation on subsets of $\{0,1\}^+$, then A is *$\mathcal{L}$-hard under $\preceq$*, written $\mathcal{L} \preceq A$, if $B \preceq A$ for all B in $\mathcal{L}$.

The decision problem A is *$\mathcal{L}$-complete under $\preceq$* if A is in $\mathcal{L}$ and A is $\mathcal{L}$-hard under $\preceq$. Let (A, M) be a promise problem and $\mathcal{L}$ be a complexity class. Then (A, M) is *$\mathcal{L}$-hard under $\preceq$* if each solution A' of (A, M) is $\mathcal{L}$-hard under $\preceq$. Similarly, (A, M) is *$\mathcal{L}$-complete under $\preceq$* if (A, M) is in $\mathcal{L}$ and (A, M) is $\mathcal{L}$-hard under $\preceq$. In the common case that reducibility is by polynomial-time transformations, "$\mathcal{L}$-complete" abbreviates "$\mathcal{L}$-complete under $\leq_m^p$."

One can think of a result of the form "A is $\mathcal{L}_1$-hard under $\preceq$" as providing a "lower bound" on the complexity of the problem A; that is, A is a hardest problem in $\mathcal{L}_1$ to within $\preceq$ reducibility. Similarly, "$A \in \mathcal{L}_2$" gives an "upper bound" on the complexity of A. If $\mathcal{L}_1 = \mathcal{L}_2 \stackrel{\mathrm{def}}{=} \mathcal{L}$, then A is $\mathcal{L}$-complete under $\preceq$; this gives in a certain sense (that is, to within $\preceq$ reducibility) a precise classification of the complexity of A.

Theorem 3.3 uses a type of reduction that is "weaker" than $\leq_m^p$, namely, probabilistic polynomial (PR) reduction. The definition is usually given in terms of probabilistic Turing machines. To avoid defining this model, we give an equivalent definition in terms of polynomial-time computable transformations f that take two arguments, u and w. As before, u represents an instance of a decision problem, and w is an "random" $Q(|u|)$-bit input, where Q is a polynomial. The next definition is due to Vazirani and Vazirani [35]. Adleman and Manders [36] had earlier defined a version of this, called *unfaithful random reduction*, where $\varepsilon = \frac{1}{2}$.

Definition 2.8: Let A, B be decision problems. Then $B \leq_{\mathrm{PR}} A$ if there is a polynomial-time computable function $f(u, w)$, a polynomial $Q(\cdot)$, and a real number $0 < \varepsilon \leq 1$, such that for all $u \in \{0, 1\}^+$

1) if $u \in B$, then $f(u, w) \in A$ for all $Q(|u|)$-bit words w;
2) if $u \notin B$, then $f(u, w) \notin A$ for at least a fraction ε of the $Q(|u|)$-bit words w.

Item 2) says that, for each $u \notin B$, if we think of w as being chosen uniformly at random from $\{0, 1\}^{Q(|u|)}$, then the probability that $f(u, w) \notin A$ is at least ε. It is obvious that $B \leq_m^p A$ implies $B \leq_{\mathrm{PR}} A$. Because given the transformation f showing that $B \leq_m^p A$, if f' is defined by $f'(u, w) = f(u)$, then f' shows that $B \leq_{\mathrm{PR}} A$. This implication is why we say that $\leq_{\mathrm{PR}}$ is weaker than $\leq_m^p$. It is not known whether the reverse implication, $B \leq_{\mathrm{PR}} A$ implies $B \leq_m^p A$, holds. It is known [35] that $\leq_{\mathrm{PR}}$ is a transitive relation on decision problems.

3) Oracles and Functions: Another concept from complexity theory that we use is that of solving a problem in polynomial time with an "oracle." Let h: $\{0, 1\}^+ \to \{0, 1\}^+$ where there is a polynomial Q such that $|h(u)| \leq Q(|u|)$ for all $u \in \{0, 1\}^+$. Intuitively, A is in the class of decision problems that can be solved in polynomial time with oracle h, denoted P^h, if A can be decided by a polynomial-time algorithm that can call h as a subroutine, and the answer $h(u)$ to any subroutine call u is returned instantly. The algorithm can call the subroutine (ask the oracle) many times during its computation, but the number of calls is limited by the polynomial-time bound of the algorithm. For a formal definition of this in terms of "oracle Turing machines," see [26, Sec. 5.1]. If $A \subseteq \{0, 1\}^+$ is a decision problem, then $\mathrm{P}^A = \mathrm{P}^{h_A}$ where $h_A(u) = 1$ if $u \in A$ or 0 if $u \notin A$. If $\mathcal{H}$ is a class of functions, then $\mathrm{P}^{\mathcal{H}}$ is the union of P^h over all $h \in \mathcal{H}$. If $\mathcal{D}$ is a class of decision problems, $\mathrm{P}^{\mathcal{D}}$ is defined similarly.

Above we have defined complexity classes of decision problems. It is standard to attach the prefix "F" to a complexity class of decision problems to denote the corresponding complexity class of functions. For example, $\mathrm{FP}^{\mathcal{H}}$ is the class of functions that can be computed in polynomial-time using any oracle function in $\mathcal{H}$. We consistently write the names of decision problems in capitals (e.g., MBE, MBE+D) and the names of functions in italic (e.g., $MaxBlkRate$, $MaxConcat$).

In Sections IV and V we will need additional complexity classes and reductions; we defer definitions of these until the point where they are needed.

4) Three Propositions: We close this section with three propositions having easy proofs, which are not given. The first gives a sufficient condition for a promise problem to be $\mathcal{L}$-hard; this condition is shown in our proofs of $\mathcal{L}$-hardness.

Proposition 2.4: Let (A, M) be a promise problem and $\mathcal{L}$ be a complexity class. Assume that for each B in $\mathcal{L}$ there is a polynomial-time transformation f such that, for all $u \in \{0, 1\}^+$: i) $M(f(u)) = 1$ (that is, $f(u)$ satisfies the promise); and ii) $u \in B$ iff $f(u) \in A$. Then (A, M) is $\mathcal{L}$-hard under $\leq_m^p$.

The second shows that we obtain stronger results by using fewer restrictions when showing membership in a class, and by using more restrictions when showing hardness.

Proposition 2.5: If (A, M_1) belongs to $\mathcal{L}$, then $(A, M_1 \wedge M_2)$ belongs to $\mathcal{L}$. If $(A, M_1 \wedge M_2)$ is $\mathcal{L}$-hard under $\preceq$, then (A, M_1) is $\mathcal{L}$-hard under $\preceq$.

The third gives cases where the complexity of a problem can be related to the $\mathrm{P} = \mathrm{NP}$ question even if the problem is not known to be NP-complete. The proof follows easily from the fact noted above, that if $\mathrm{P} = \mathrm{NP}$ then $\Sigma_k^p = \mathrm{P}$ for all $k \geq 1$.

Proposition 2.6: Let (A, M) be a promise problem such that (A, M) belongs to Σ_k^p for some $k \geq 1$, and (A, M) is either NP-hard or co-NP-hard under $\leq_m^p$. Then (A, M) belongs to P iff $\mathrm{P} = \mathrm{NP}$.

E. Boolean Formulas

To show that a particular decision problem A is $\mathcal{L}$-hard under $\preceq$, where $\preceq$ is a transitive relation on decision problems, it is sufficient to show that $B \preceq A$ where B is already known to be $\mathcal{L}$-hard under $\preceq$. Many of the decision problems B that we use in this way involve Boolean formulas in their instances. So we review some terminology regarding Boolean formulas.

Here we consider Boolean formulas in either conjunctive normal form or disjunctive normal form. Let $\{v_1, \ldots, v_n\}$ be a set of Boolean variables. The corresponding set of *literals* is $L_n = \{v_1, \overline{v}_1, \ldots, v_n, \overline{v}_n\}$, where $\overline{v}_i$ denotes the negation of v_i. A *DNF formula* (*of n variables*) is a set $F = \{D_1, D_2, \ldots, D_m\}$ of *disjuncts* where $D_j \subseteq L_n$ for $1 \leq j \leq m$. The formula F is a *kDNF formula* if $|D_j| \leq k$ for all j. Logically

$$F = \bigvee_{j=1}^{m} \bigwedge_{\ell \in D_j} \ell.$$

An *assignment* is an n-bit word $\alpha = \alpha_1 \cdots \alpha_n$. The disjunct D_j is *satisfied by* α if D_j is true when v_i is set to α_i for all i; more precisely, for each $\ell \in D_j$, if $\ell = v_i$ then $\alpha_i = 1$ and if $\ell = \overline{v}_i$ then $\alpha_i = 0$. The formula F is *satisfied by* α if there exists a D_j that is satisfied by α. The formula F is a *tautology* if F is satisfied by all assignments $\alpha \in \{0, 1\}^n$.

A conjunctive normal form is the logical dual of the above. A *CNF formula* is a set $F = \{C_1, C_2, \ldots, C_m\}$ of *conjuncts* where $C_j \subseteq L_n$ for $1 \leq j \leq m$. The formula F is a *kCNF formula* if $|C_j| \leq k$ for all j. Logically

$$F = \bigwedge_{j=1}^{m} \bigvee_{\ell \in C_j} \ell.$$

The conjunct C_j is *satisfied by* α if there exists an $\ell \in C_j$ such that: either $\ell = v_i$ and $\alpha_i = 1$, or $\ell = \overline{v}_i$ and $\alpha_i = 0$. The formula F is *satisfied by* α if C_j is satisfied by α for all j.

It is convenient to assume that no disjunct or conjunct contains both a variable and its negation. Such disjuncts (conjuncts) can be removed from the formula because they are identically false (true).

In the reductions involving formulas we use that for each disjunct $D \subseteq L_n$ (resp., each conjunct $C \subseteq L_n$) there is a deterministic FSTD with distinguished states d and t such that for all $\alpha \in \{0, 1\}^n$: if α satisfies D (resp., C) then α is generated from d and $\delta(d, \alpha) = t$; and if α does not satisfy D (resp., C) then α is not generated from d. Moreover, the size of the FSTD

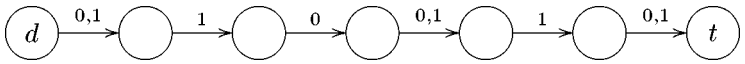

Fig. 2. FSTD that generates the set of 6-bit satisfying assignments of $v_2 \wedge \overline{v}_3 \wedge v_5$.

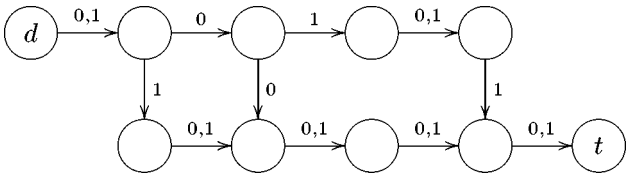

Fig. 3. FSTD that generates the set of 6-bit satisfying assignments of $v_2 \vee \overline{v}_3 \vee v_5$.

is $O(n)$. This is illustrated in Fig. 2 for $n = 6$ and the disjunct $v_2 \wedge \overline{v}_3 \wedge v_5$, and in Fig. 3 for $n = 6$ and the conjunct $v_2 \vee \overline{v}_3 \vee v_5$. Note that all paths from d to t have the same length.

III. Complexity of Minimizing Encoder and Decoder Size

In this section, we prove several theorems that classify the computational complexities of the encoder and decoder circuit minimization problems defined above. We first state the results and some corollaries. The proofs are then given.

We begin by proving (Theorem 3.1) that the minimum block decoder problem is co-NP-hard under $\leq_m^p$. This result is subsumed by Theorem 3.4 which states that the minimum block decoder problem is Σ_3^p-hard under $\leq_m^p$. However, because the proof of Theorem 3.1 is considerably simpler than that of Theorem 3.4 we give the proof as an introduction to the more complicated proofs to follow. In addition, the proof of Theorem 3.1 has special properties that can be used to show that the result holds in the context of state-dependent decoding (see Section VI). Finally, if one wants only to show that the minimum block decoder problem can be solved in polynomial time iff $\mathrm{P} = \mathrm{NP}$, then Theorem 3.1 suffices (by Theorem 2.3 and Proposition 2.6).

Theorem 3.1: Let $R < 1$. MBD (rate $\geq R$, FM) is co-NP-hard under $\leq_m^p$.

We then prove two incomparable results (neither one implies the other) on the complexity of finding encoders of minimum or approximately minimum size. The first result is that the MBE problem (where there is a target bound only on encoder size), the MBE+D problem (where there is a target bound on the sum of encoder and decoder size), and the MBE&D problem (where there are separate target bounds for encoder size and decoder size) are NP-hard under $\leq_m^p$. Moreover, assuming that $\mathrm{P} \neq \mathrm{NP}$ and $\varepsilon > 0$, there is no polynomial-time algorithm that when given (G, p, q) is guaranteed to find an encoder (encoder/decoder) whose size (total size) is at most $q^{\frac{1}{4}-\varepsilon}M$, where M is the minimum possible size (total size). The notion of an *approximation algorithm* for these problems is defined next. Let $M_E(G, p, q)$ (resp., $M_{E+D}(G, p, q)$) be the minimum of $|\hat{\mathcal{E}}|$ (resp., $|\hat{\mathcal{E}}| + |\hat{\mathcal{D}}|$) over all encoder/decoder circuits $(\hat{\mathcal{E}}, \hat{\mathcal{D}})$ for (G, p, q). Let g be a function from $\boldsymbol{N}^+$ to the set of real numbers ≥ 1. Say that MBE (resp., MBE+D) *can be approximated within the factor $g(q)$ in polynomial time*, if there is an $e_0 \in \boldsymbol{N}^+$ and a polynomial-time algorithm $\mathcal{A}$ which, when given (G, p, q), outputs an encoder circuit $\hat{\mathcal{E}}$ for (G, p, q) (resp., an encoder/decoder circuit pair $(\hat{\mathcal{E}}, \hat{\mathcal{D}})$ for (G, p, q)) such that

$$|\hat{\mathcal{E}}| \leq \max(g(q) \cdot M_E(G, p, q),\ M_E(G, p, q) + e_0) \quad (2)$$

(resp., $|\hat{\mathcal{E}}| + |\hat{\mathcal{D}}|$
$\leq \max(g(q) \cdot M_{E+D}(G, p, q), M_{E+D}(G, p, q) + e_0)$).

Because we prove the nonexistence of an approximation algorithm (assuming $\mathrm{P} \neq \mathrm{NP}$), the result is made stronger by allowing the approximation algorithm to make an additive error of e_0 when $(g(q) - 1) \cdot M_E(G, p, q)$ (resp., $(g(q) - 1) \cdot M_{E+D}(G, p, q)$) is smaller than e_0. (If $g(q)$ is increasing and the optimum circuit size is nonzero, then the first term in the max dominates for all sufficiently large q.)

Theorem 3.2: Let $R < 1$.

1) MBE, MBE+D, and MBE&D, all with the restrictions (rate $\geq R$, FM), are NP-hard under $\leq_m^p$.
2) If $\mathrm{P} \neq \mathrm{NP}$ and $0 < \varepsilon < \frac{1}{4}$, then neither MBE (rate $\geq R$, FM) nor MBE+D (rate $\geq R$, FM) can be approximated within the factor $q^{\frac{1}{4}-\varepsilon}$ in polynomial time.

The second result about encoders shows that the MBE&D problem is Σ_2^p-hard under $\leq_{\mathrm{PR}}$ (Definition 2.8) Thus, Theorem 3.2 part 1) proves hardness for a "smaller" class (NP) under a "stronger" reduction $(\leq_m^p)$, while Theorem 3.3 proves hardness for a "larger" class (Σ_2^p) under a "weaker" reduction $(\leq_{\mathrm{PR}})$. Both of these results would be subsumed by the result that the MBE&D problem is Σ_2^p-hard under $\leq_m^p$, but this question is open, as is the question of whether either MBE or MBE+D is Σ_2^p-hard under either $\leq_m^p$ or $\leq_{\mathrm{PR}}$.

Theorem 3.3: Let $R < \frac{1}{2}$. MBE&D (rate $\geq R$, FM) is Σ_2^p-hard under $\leq_{\mathrm{PR}}$.

The final result of this section permits us to precisely classify the decoder circuit minimization problem in terms of the polynomial hierarchy.

Theorem 3.4: Let $R < 1$. MBD (rate $\geq R$, FM) is Σ_3^p-hard under $\leq_m^p$.

The following immediate corollaries of these theorems, together with Theorem 3.2 part 2), are the main results of this section. The statement of part 6) of Corollary 3.5 needs a new definition. We say that μ is a *circuit minimization function* if, for each circuit U, $\mu(U) = U'$ such that U and U' compute the same function $\mathcal{F}$ and the size of U' is minimal, that is, $|U'| = \Xi(\mathcal{F})$.

Corollary 3.5:

1) Let A be one of the decision problems MBE, MBD, MBE+D, or MBE&D. Then $A \in \mathrm{P}$ iff $\mathrm{P} = \mathrm{NP}$.

2) If either MBD belongs to NP, or one of MBE, MBE+D, MBE&D belongs to co-NP, then NP = co-NP = Σ_k^p for all $k \geq 2$.
3) MBE&D is Σ_2^p-complete under $\leq_{\mathrm{PR}}$.
4) If MBE&D belongs to P^{NP} then the polynomial hierarchy collapses to the third level.
5) MBD is Σ_3^p-complete under $\leq_m^p$.
6) There is a circuit minimization function μ such that, if MBD belongs to P^{μ}, then the polynomial hierarchy collapses to the third level.

Moreover, parts 1)–6) hold with the restrictions (rate $\geq R$, FM) for any constant $R < 1$ in parts 1), 2), 5), and 6) and $R < \frac{1}{2}$ in parts 3) and 4).

Remark: The proofs of Theorems 3.1–3.4 actually show that these hold with the restriction that the memory of G is at most q. So the finite memory restriction can be replaced by this stronger restriction also in Corollary 3.5.

Parts 2), 4), and 6) each contain a statement of the form: "If $\mathcal{X}$, then the polynomial hierarchy collapses to some finite level." Thus, if the polynomial hierarchy is strict, then each of these statements $\mathcal{X}$ is false. Parts 1), 2), 3), and 5) of this corollary are immediate from Theorems 2.3, 3.1, 3.2, 3.3, and 3.4, and Propositions 2.5 and 2.6. Part 4) is an implication of part 3), using methods of Adleman [37] and Karp and Lipton [38].[2] Part 6) is proved from part 4) as follows. The "decision version" of circuit minimization has instances (U, K) where U is a circuit and $K \in \boldsymbol{N}^+$, and the question is whether there is a circuit U' equivalent to U with $|U'| \leq K$. This problem has a $\exists\forall$ form: there exists a U' such that $|U'| \leq K$ and, for all inputs x, $U(x) = U'(x)$. A standard argument now shows that there is a circuit minimization function in $\mathrm{FP}^{\Sigma_2^p}$. The proof is completed by another standard fact that if any Σ_3^p-hard problem belongs to $\mathrm{P}^{\Sigma_2^p}$, then the polynomial hierarchy collapses to the third level.

The remainder of this section contains proofs of the theorems above. The technical preliminaries in Section III-A are needed to understand all the proofs, and the proof of Theorem 3.1 (Section III-B) is a good introduction to the more complicated proofs to follow. However, the proofs of Theorems 3.2–3.4 are independent of one another. Additional technical material in Section III-E is needed to understand the proof of Theorem 3.4, as well as the proof of Lemma 4.4.

A. Technical Preliminaries

We first cover some technicalities that arise in all of our hardness proofs. By dealing with them here, we avoid dealing with them in each proof. When giving a hardness proof, it will be useful to allow words over the alphabet $\{0, 1, 2\}$. In particular, we are interested in q-bit codewords (where $q \geq 2$) that contain exactly one occurrence of the symbol 2; denote this set of words containing "One Two" by

$$\mathrm{OT}_q \stackrel{\mathrm{def}}{=} (\{0, 1\}^* \cdot 2 \cdot \{0, 1\}^*) \cap \{0, 1, 2\}^q.$$

[2]For those readers familiar with complexity theory, the argument follows. Using that $\Sigma_2^p \leq_{\mathrm{PR}}$ MBE&D, it follows as in [37] that if MBE&D $\in \mathrm{P}^{\mathrm{NP}}$ then $\Sigma_2^p \subseteq \mathrm{P}^{\mathrm{NP}}$/poly. Hemaspaandra *et al.* [39, Theorem 11] have observed that a result of Karp and Lipton [38] relativizes; in particular, if $\Sigma_2^p \subseteq \mathrm{P}^{\mathrm{NP}}$/poly then $\Sigma_k^p = \Sigma_3^p$ for all $k \geq 3$.

As will become clear shortly, the symbol 2 serves as a "synchronization symbol"; for example, the DIFs we construct have the property that, in any block codebook $\mathcal{C}$ of q-symbol words, the (unique) occurrence of "2" appears at the same position in all codewords in $\mathcal{C}$. We define the function θ_q mapping OT_q to $\{0, 1\}^+$.

First, assume that $y = 2y'$ where $y' \in \{0, 1\}^{q-1}$. Let $a = \lceil \sqrt{q} \rceil$. Write $y' = w_1 w_2 \cdots w_b$ where $|w_i| = a - 1$ for $1 \leq i < b$, $|w_b| \leq a - 1$, and $b = \lceil (q-1)/(a-1) \rceil$. Define

$$\theta_q(y) = \theta_q(2y') \stackrel{\mathrm{def}}{=} 0^a\ 1\ w_1\ 1\ w_2\ 1\ \cdots\ 1\ w_b\ 1. \quad (3)$$

An important property of this mapping is that each cyclic shift of $\theta_q(y)$ contains at most one occurrence of 0^a. θ_q is defined in general for $y \in \mathrm{OT}_q$ as follows. We illustrate the definition by using the example $y = 000101210100011$, where $q = 15$, $a = 4$, and $b = \lceil 14/3 \rceil = 5$. Let $y = w2w'$ where $ww' \in \{0, 1\}^{q-1}$. Apply θ_q to $2w'3w$, where "3" is mapped to "3"; in the example (where dots have been inserted as an aid to parsing)

$$\theta_{15}(2 \cdot 101 \cdot 000 \cdot 1130 \cdot 001 \cdot 01)$$
$$= 0000 \cdot 1 \cdot 101 \cdot 1 \cdot 000 \cdot 1 \cdot 1130 \cdot 1 \cdot 001 \cdot 1 \cdot 01 \cdot 1.$$

Then $\theta_q(2w'3w)$ is cyclicly shifted so that "3" is the last symbol; in the example, this gives

$$0 \cdot 1 \cdot 001 \cdot 1 \cdot 01 \cdot 1 \cdot 0000 \cdot 1 \cdot 101 \cdot 1 \cdot 000 \cdot 1 \cdot 113.$$

Finally, the "3" is removed to obtain $\theta_q(y)$; in the example

$$\theta_{15}(0 \cdot 001 \cdot 01 \cdot 2 \cdot 101 \cdot 000 \cdot 11)$$
$$= 0 \cdot 1 \cdot 001 \cdot 1 \cdot 01 \cdot 1 \cdot 0000 \cdot 1 \cdot 101 \cdot 1 \cdot 000 \cdot 1 \cdot 11. \quad (4)$$

Let $\theta(q) = q + a + b$, so if $|y| = q$ then $|\theta_q(y)| = \theta(q)$. Recall that $a + b = O(\sqrt{q})$. We have chosen the map θ_q so that $\theta(q) = q + o(q)$, as opposed to $\theta(q) \approx cq$ for some constant $c > 1$. This will be useful in showing hardness when restricted to rate $p/q \geq R$ for arbitrary $R < 1$.

We also define θ_q^{-1} on the set of cyclic shifts of words $\theta_q(y)$ for $y \in \mathrm{OT}_q$, that is, cyclic shifts of words of the form (3). Let $\hat{y}$ be a cyclic shift of $0^a\ 1\ w_1\ 1\ w_2\ 1\ \cdots\ 1\ w_b\ 1$. Let $\hat{y}'$ be the minimum right cyclic shift of $\hat{y}$ (the choice of right rather than left is arbitrary) such that $\hat{y}' = \theta_q(y)$ for some $y \in \mathrm{OT}_q$. Then $\theta_q^{-1}(\hat{y}) = y$. For example, if

$$\hat{y} = 00 \cdot 1 \cdot 101 \cdot 1 \cdot 000 \cdot 1 \cdot 110 \cdot 1 \cdot 001 \cdot 1 \cdot 01 \cdot 1 \cdot 00$$

(this is a cyclic shift of (4)), the minimum right cyclic shift is

$$\hat{y}' = 0000 \cdot 1 \cdot 101 \cdot 1 \cdot 000 \cdot 1 \cdot 110 \cdot 1 \cdot 001 \cdot 1 \cdot 01 \cdot 1$$

and

$$\theta_{15}^{-1}(\hat{y}) = 2 \cdot 101 \cdot 000 \cdot 110 \cdot 001 \cdot 01.$$

All DIFs $G = (V, E)$ (using labels $\{0, 1, 2\}$) and codeword lengths q described in our reductions will have the following form: i) the set of states is partitioned into disjoint *levels* $V = V_0 \cup V_1 \cup \cdots \cup V_{q-1}$, where the states in level V_0 are called *synchronizing states*; ii) there is an edge directed from a state in level V_i to a state in level V_j iff $j = (i+1) \bmod q$, and the label of that edge is "2" iff $i = 0$. It follows that G is periodic with period q, and if y is generated from $v \in V$ then the first symbol of y is "2" iff v is a synchronizing state. We call any DIF of this form *special*.

When we describe a special DIF $G = (V, E)$ with label alphabet $\{0, 1, 2\}$, we actually mean the DIF, denoted $\theta_q(G)$, with label alphabet $\{0, 1\}$, having the property that $y \in \mathrm{OT}_q$ is generated by G iff $\theta_q(y)$ is generated by $\theta_q(G)$. The idea is that,

whenever G would produce a 2 (this must be starting in a state in V_0), $\theta_q(G)$ will produce 0^a1. Then $\theta_q(G)$ produces $a-1$ 0's and 1's as G would, produces a 1, produces $a-1$ more 0's and 1's as G would, produces a 1, and so on, until $\theta_q(G)$ produces w_b as in (3); then $\theta_q(G)$ produces a 1 and returns to a state in V_0 as G would. We outline how $\theta_q(G)$ can be formally defined to satisfy $|\theta_q(G)| = O(\sqrt{q}|G|)$. Let the states of G be partitioned into levels $V_0, V_1, \ldots, V_{q-1}$ as above. Then in $\theta_q(G)$: level V_0 is "repeated" an additional a times to generate 0^a1 where G would generate 2; and each level $V_{j(a-1)}$ for $1 \le j \le b-1$, as well as level V_{q-1}, is "repeated" an additional one time to generate 01 or 11 where G would generate 0 or 1, respectively. Because G is periodic with period q, $\theta_q(G)$ is periodic with period $\theta(q)$. Because of the similarity between G and $\theta_q(G)$, it is easy to see (and we let the reader verify) that if G is a special DIF then $\theta_q(G)$ is a DIF, and if G has finite memory (resp., memory $\le q$) then $\theta_q(G)$ has finite memory (resp., memory $\le \theta(q)$). When q is clear from context (as it usually is) we sometimes write θ in place of θ_q.

Let G be a special DIF with period q. Let $\hat{\mathcal{C}} \subseteq \{0, 1\}^{\theta(q)}$ be $\mathcal{S}(\theta(G))$-concatenable. Let $\hat{y} \in \hat{\mathcal{C}}$. By definition of special G and θ, the word $\hat{z} = \hat{y}_1\hat{y}_2\cdots\hat{y}_{\theta(q)}\hat{y}_1\hat{y}_2\cdots\hat{y}_{a-1}$ contains exactly one occurrence of 0^a. Writing $\hat{z} = w0^aw'$, we say that $\hat{y}$ has *rotation* $|w|$; so the rotation ρ always satisfies $0 \le \rho \le |\hat{y}| - 1 = \theta(q) - 1$. Similarly, if $y \in \mathrm{OT}_q$ for some q and $y = w2w'$, then y has rotation $|w|$.

Claim 1: Let G be a special DIF.

Let $\hat{\mathcal{C}} \subseteq \{0, 1\}^{\theta(q)}$ be $\mathcal{S}(\theta(G))$-concatenable. All words in $\hat{\mathcal{C}}$ have the same rotation.

Let $\mathcal{C} \subseteq \mathrm{OT}_q$ be $\mathcal{S}(G)$-concatenable. All words in $\mathcal{C}$ have the same rotation.

Proof: We prove the first statement. The proof of the second is identical. For $i = 1, 2$, assume that $\hat{y}_i \in \hat{\mathcal{C}}$, $\hat{y}_i$ has rotation ρ_i, and $\rho_1 \ne \rho_2$. Note that $\hat{y}_i\hat{y}_i$ can be written $w_i0^aw_i'$ where $|w_i| = \rho_i$ and $|w_i'| = 2\theta(q) - a - \rho_i$. Then

$$\hat{y}_1\hat{y}_1\hat{y}_2\hat{y}_2 = w_10^aw_1'w_20^aw_2' \in \mathcal{S}(\theta(G))$$

and

$$|0^aw_1'w_2| = 2\theta(q) + \rho_2 - \rho_1.$$

By assumption, $\rho_2 - \rho_1 \not\equiv 0 \bmod \theta(q)$, and this violates the $\theta(q)$-periodicity of $\theta(G)$. □

It follows from Claim 1 and the definition of $\theta_q(G)$ that if $\mathcal{C} \subseteq \mathrm{OT}_q$ is $\mathcal{S}(G)$-concatenable, then $\hat{\mathcal{C}} = \{\theta_q(y) \,|\, y \in \mathcal{C}\}$ is $\mathcal{S}(\theta_q(G))$-concatenable and $|\hat{\mathcal{C}}| = |\mathcal{C}|$.

Several easy observations made above are collected in the following claim; it is stated without further proof. This claim states that key properties of G are inherited by $\theta_q(G)$.

Claim 2: If G is a special DIF having finite memory (resp., memory $\le q$) and (G, p, q) is block feasible, then $\theta_q(G)$ is a DIF having finite memory (resp., memory $\le \theta(q)$) and $(\theta_q(G), p, \theta(q))$ is block feasible.

In the hardness proofs in Sections III-B–III-D, we will use special DIFs having only one synchronizing state. In this case, Claim 1 and the following claim permit us essentially to ignore the issue of rotation.

Claim 3: Let G be a special DIF having one synchronizing state. Let $\mathcal{C}$ be a block codebook for (G, p, q). Let $\mathcal{C}'$ be the set of words formed by cyclicly shifting (by the same amount) each word in $\mathcal{C}$ so that it begins with "2." Then $\mathcal{C}'$ is a block codebook for (G, p, q).

Proof: By Claim 1, there is a number ρ (the rotation) with $0 \le \rho \le q-1$ such that each y in $\mathcal{C}$ can be written $y = y'2y''$ where $y', y'' \in \{0, 1\}^*$ and $|y'| = \rho$. Let $\mathcal{C}' = \{2y''y' \,|\, y \in \mathcal{C}\}$. We must show that $\mathcal{C}'$ is $\mathcal{S}(G)$-concatenable. For arbitrary $y \in \mathcal{C}$, $yy = y'2y''y'2y'' \in \mathcal{S}(G)$. Letting s be the unique synchronizing state of G, it follows that $2y''y'$ is generated from s and $\delta(s, 2y''y') = s$. It is then immediate that $\mathcal{C}'$ is $\mathcal{S}(G)$-concatenable. □

Claim 4: Let G be a special DIF having one synchronizing state. Then $\mathcal{S}(G)$ has finite memory m where $m \le q$.

Proof: If $y \in \mathcal{S}(G)$ and $|y| \ge q$, then y contains at least one occurrence of the symbol "2," and any path in G that generates y must be in the unique synchronizing state just before a "2" is produced. □

Finally, to use a special DIF G as a description of $\theta_q(G)$, we would like to translate encoders and decoders for G to encoders and decoders for $\theta_q(G)$, and *vice versa*, without changing the size of the encoder or decoder. One minor technical point is that we have not defined encoders and decoders (and their circuits) when the symbols are $\{0, 1, 2\}$. But if $\mathcal{C}$ is a block codebook for (G, p, q) where G is special, then we know from Claim 1 that all words in $\mathcal{C}$ contain exactly one symbol 2 at the same position. Therefore, we require encoders (decoders) only to encode (decode) the 0's and 1's of words in $\mathcal{C}$. That we can translate between G and $\theta_q(G)$ is immediate from the following observations. For a word $y \in \mathrm{OT}_q$ having fixed rotation ρ, we can think of θ_q as replacing the symbol 2 by 0^a and inserting additional 1's at various positions in y. The locations of these positions and the number a depend only on $q = |y|$ and ρ, but not on the actual 0's and 1's of y. Because the Boolean constants come "for free" in our definition of the size of a circuit, a circuit can set any output to a constant bit while using no computation steps. The next claim should be now be obvious.

Claim 5: Let $p, q, K_1, K_2 \in \boldsymbol{N}^+$ and let G be a special DIF. There is an encoder/decoder pair $(\hat{\mathcal{E}}, \hat{\mathcal{D}})$ of circuits for (G, p, q) with $|\hat{\mathcal{E}}| \le K_1$ and $|\hat{\mathcal{D}}| \le K_2$ iff there is an encoder/decoder pair $(\hat{\mathcal{E}}', \hat{\mathcal{D}}')$ of circuits for $(\theta(G), p, \theta(q))$ with $|\hat{\mathcal{E}}'| \le K_1$ and $|\hat{\mathcal{D}}'| \le K_2$.

A Note on Notation: The rest of the proofs in Section III and the proof of Lemma 4.4 in Section IV have some features in common, and consistent notation is used for common features. In each proof we describe a set of codewords in OT_q and a special DIF G that generates them. The binary part of a codeword is broken into fields of fixed length. In all the proofs, each codeword has a β-field, used to increase the rate and to divide the codewords into various types. We use k to denote the length of β. (For example, in the proof given in Section III-B, codewords of two types are distinguished by $\beta = 0^k$ or $\beta \ne 0^k$.) For reductions where the input to the reduction includes a Boolean formula F, we let m denote the number of clauses. For a formula of the variables u_i (resp., v_i, w_i) for $1 \le i \le n$, the length-n

field μ (resp., ν, ω) represents an assignment to the u_i's (resp., v_i's, w_i's). In the reduction involving graphs (Theorem 3.2), n (resp., m) denotes the number of vertices (resp., edges) of the graph. The fields η and ζ have length ℓ logarithmic in m and n; they are viewed as binary numbers that serve as indexes for objects such as clauses or bit positions in other fields. The fields ξ and ψ are used only in the proofs of Theorems 3.2 and 3.3, respectively. For a codeword y containing the field ϕ, the binary word in the ϕ-field of y is denoted $\phi(y)$.

B. Proof of Theorem 3.1

We use the fact that the tautology problem for 3DNF formulas is co-NP-complete [40], [41], [26].

TAUTOLOGY
Input: A 3DNF formula F.
Question: Is F a tautology?

Theorem 3.1: Let $R < 1$. MBD (rate $\geq R$, FM) is co-NP-hard under $\leq_m^p$.

Proof: We describe a polynomial-time transformation from an arbitrary 3DNF formula F to (G, p, q) such that G is a special DIF having exactly one synchronizing state (so, by Claim 4, $\mathcal{S}(G)$ has finite memory at most q), (G, p, q) is block feasible, $p/\theta(q) \geq R$, and (G, p, q) has a decoder of size zero iff F is a tautology. The output of the transformation is $(\theta(G), p, \theta(q), 0)$.

The basic idea of the proof is that codewords have a field ν of length n, where F has n variables. For each $\alpha \in \{0, 1\}^n$, there is a codeword whose ν-field is α iff F is satisfied by α. This is done by using a graph like the one in Fig. 2 for each disjunct of F. If F is a tautology, then there is a codebook of 2^n codewords such that the ν-fields of these codewords comprise the set $\{0, 1\}^n$; in this case, the circuit of size zero that maps each codeword in the codebook to the binary word in its ν-field is a decoder. But if F is not a tautology, then no such codebook and circuit exists because there is some $\alpha \in \{0, 1\}^n$ that does not appear in the ν-field of any codeword. Our construction uses two additional fields: a β-field that is used to increase the rate close to 1 and divide the codewords into two types; and an η-field that is used to index the clauses of F. Exactly how these fields are used is described in the detailed proof given next.

Let the variables of F be $\{v_1, \ldots, v_n\}$ and the disjuncts be $\{D_1, \ldots, D_m\}$. We may assume that $m \geq 8$. Let $\ell = \lceil \log(m+1) \rceil$. The special DIF G has one synchronizing state: $V_0 = \{s\}$. Let y be a word of length q that is generated from s; therefore, $\delta(s, y) = s$. Then y has the form

$$y = 2\beta\eta\nu \tag{5}$$

where $|\beta| = k$, $|\eta| = \ell$, and $|\nu| = n$, where k is to be determined. Thus, $q = k + \ell + n + 1$. Let $p = k + n$. Choose k large enough that $k > \ell$ and $p/\theta(q) \geq R$. Recalling that $R < 1$ and there is a constant c such that $\theta(q) \leq q + c\sqrt{q}$, the condition $p/\theta(q) \geq R$ can be met for some $k = c_R(q - p) = c_R(\ell + 1)$ where the constant c_R depends on c and R.

For y as in (5), define the subwords $\beta(y)$, $\eta(y)$, and $\nu(y)$ in the obvious way. The special DIF G is constructed so that any q-bit word y generated from s satisfies either

1) $\beta(y) = 0^k$, and letting $j = \mathrm{num}(\eta(y))$, then $1 \leq j \leq m$ and $\nu(y)$ satisfies the disjunct D_j (call these codewords of the *first type*); or
2) $\beta(y) \in \{0, 1\}^k - \{0^k\}$, $\eta(y) = 0^\ell$, and $\nu(y) \in \{0, 1\}^n$ (call these codewords of the *second type*).

It is straightforward (but tedious to define formally) that there is a special DIF G having this property, and, in addition, G has exactly one synchronizing state s, and $|G| = O(k + 2^\ell n)$. In the case that $\beta = 0^k$, the part of the DIF that generates η is a binary tree of depth ℓ with m leaves, $d_1, \ldots, d_m$. The (unique) path from the root to d_j generates the word η such that $\mathrm{num}(\eta) = j$. For each $1 \leq j \leq m$, the set of n-bit words generated from d_j is the set of satisfying assignments of D_j. As described at the end of Section II-E and illustrated in Fig. 2, this can be done for each j by a graph having starting state d_j and some terminal state t_j. In the DIF G, the states $s, t_1, \ldots, t_m$ are identical. Because $\ell = \lceil \log(m+1) \rceil$ and $k = O(\ell)$, the size of G is polynomial in n and m.

We next show that (G, p, q) is block feasible. The set of codewords of the first type (which all have $\beta(y) = 0^k$) is disjoint from the set of codewords of the second type (which all have $\beta(y) \neq 0^k$). The number of codewords of the first type is at least $m2^{n-3} \geq 2^n$, because $m \geq 8$ and each of the m disjuncts is satisfied by at least 2^{n-3} assignments. The number of codewords of the second type is exactly $(2^k - 1)2^n$. Therefore, the set of codewords of both types is a $\mathcal{S}(G)$-concatenable set of size at least $2^k 2^n = 2^p$.

It remains only to show that (G, p, q) has a decoder of size zero iff F is a tautology. Say first that F is a tautology. A block codebook $\mathcal{C}$ of size 2^p can be formed as follows: $\mathcal{C}$ contains all codewords of the second type; for each $\alpha \in \{0, 1\}^n$, $\mathcal{C}$ contains a codeword y of the first type, where $\beta(y) = 0^k$, $\nu(y) = \alpha$, and $\eta(y)$ is chosen by $\mathrm{num}(\eta(y)) = j$ where the assignment α satisfies the disjunct D_j (at least one D_j exists for each α, because F is a tautology). The decoder of size zero maps the $(q-1)$-bit word $\beta\eta\nu$ to the p-bit word $\beta\nu$.

Say now that (G, p, q) has a decoder of size zero. Let $\mathcal{C}$ be a block codebook for (G, p, q) having a decoder of size zero. By Claim 3, we can assume that all words in $\mathcal{C}$ are of the form $2y'$ where $y' \in \{0, 1\}^{q-1}$. A decoder of size zero is defined by its output mapping, which maps each of the p outputs (the dataword) to one of the $q - 1$ inputs (the 0's and 1's of the codeword). Let ϕ: $\{1, 2, \ldots, p\} \to \{1, 2, \ldots, q-1\}$ be the output mapping of the decoder of size zero, and let Φ be the range of ϕ. Because each p-bit word must be an output dataword for some input codeword, ϕ must be one-to-one, so $|\Phi| = p$.

We claim that Φ cannot contain any coordinate of η. To see this, let η_i be a coordinate of η. Note that $\eta_i(y) = 0$ in all codewords y of the second type. Therefore, there are at most $2^{\ell-1}2^n$ codewords $y \in \mathcal{C}$ such that $\eta_i(y) = 1$. But every coordinate in Φ must have a "1" in that coordinate for $2^{p-1} = 2^{k+n-1}$ codewords $y \in \mathcal{C}$. Because we have chosen $k > \ell$ this is a contradiction.

It follows from $p = k + n$ and the fact that Φ contains no coordinate of η, that Φ must contain all coordinates of β and ν. This implies that F is a tautology, because for all $\alpha \in \{0, 1\}^n$ there is a $y \in \mathcal{C}$ with $\beta(y) = 0^k$ and $\nu(y) = \alpha$. □

C. Proof of Theorem 3.2

Before proving the theorem, we need to state some definitions and a known result. Let $H = (V, E)$ be a connected, undirected graph containing no multiple edges or self-loops. A set $I \subseteq V$ is an *independent set* in H if, for all $v, v' \in I$, the vertices v and v' are not connected by an edge in E. The *independence number of* H, denoted $\alpha(H)$, is the maximum of $|I|$ over all independent sets I in H. The relevant decision problem has instances of the form (H, K) and question "$\alpha(H) \geq K$?" It has been known since the early 1970s that this problem is NP-complete (Karp [41]; the reduction is also implicit in Cook [40]). More recently, it has been shown in a remarkable series of papers that it is hard even to *approximate* $\alpha(H)$. We use the currently best known result of this type, which is due to Håstad [42]. See also Bellare, Goldreich, and Sudan [43] (a close precursor of [42]) for more discussion of the history of nonapproximation results for $\alpha(H)$ and the methods used.[3]

Theorem 3.6 [42]: Let $0 < \varepsilon < \frac{1}{2}$. There is an NP-complete decision problem B_1 and a polynomial time transformation f mapping instances of B_1 to pairs (H, K) where H is an undirected graph and $K \in \boldsymbol{N}^+$ such that

1) if $u \in B_1$ then $f(u) = (H, K)$ where $\alpha(H) \geq K$; and
2) if $u \notin B_1$ then $f(u) = (H, K)$ where $\alpha(H) < K/n^{\frac{1}{2}-\varepsilon}$, where n is the number of vertices of H.

It follows from this that if there is a polynomial-time algorithm $\mathcal{A}$ that, when given H, is guaranteed to find an independent set I in H with $|I| \geq \alpha(H)/n^{\frac{1}{2}-\varepsilon}$, then $\mathcal{A}$ can be used to solve the NP-complete problem B_1 in polynomial time, and, therefore, $\mathrm{P} = \mathrm{NP}$.[4] We use Theorem 3.6 to prove a similar result about finding approximately minimum size encoders.

In the proof we use the following lemma; it is stated (in an equivalent form) in [44, Corollary 2, p. 279] as a corollary of Turan's theorem.

Lemma 3.7: If a graph has n' vertices, m' edges, and independence number α', then

$$m' \geq \frac{(n')^2}{2\alpha'} - \frac{n'}{2}.$$

Theorem 3.2: Let $R < 1$.

1) MBE, MBE+D, and MBE&D, all with the restrictions (rate $\geq R$, FM), are NP-hard under $\leq_m^p$.
2) If $\mathrm{P} \neq \mathrm{NP}$ and $0 < \varepsilon' < \frac{1}{4}$, then neither MBE (rate $\geq R$, FM) nor MBE+D (rate $\geq R$, FM) can be approximated within the factor $q^{\frac{1}{4}-\varepsilon'}$ in polynomial time.

Proof: Fix an ε' with $0 < \varepsilon' < \frac{1}{4}$ and an $R < 1$. Choose an ε such that $0 < \varepsilon < 2\varepsilon'$. We describe a polynomial-time transformation from inputs (H, K), where $H = (V, E)$ is a undirected graph and $K \in \boldsymbol{N}^+$, to outputs (G, p, q, K') such that for some constant $n_0 > 0$

I) if $\alpha(H) \geq K$, then $M_{E+D}(G, p, q) \leq K'$ and $M_E(G, p, q) > 0$; and
II) if $\alpha(H) < K/n^{\frac{1}{2}-\varepsilon}$ and $n \geq n_0$ where $n = |V|$, then $M_E(G, p, q) > q^{\frac{1}{4}-\varepsilon'}K'$.

Moreover, G is a special DIF having one synchronizing state s, (G, p, q) is block feasible, and $p/\theta(q) \geq R$. By Theorem 3.6, this is sufficient to prove part 1), taking the output of the transformation to be $(\theta(G), p, \theta(q), K')$ for MBE and MBE+D, or $(\theta(G), p, \theta(q), K', K')$ for MBE&D. A minor technical detail is that II) holds only for $n \geq n_0$. This is not a problem, because if the transformation of Theorem 3.6 outputs (H, K) where H has fewer than n_0 vertices, then $\alpha(H)$ can be found by exhaustive search in constant time. By Theorem 3.6, this also suffices to prove part 2). If part 2) were false then, as in footnote 4, the polynomial-time approximation algorithm for MBE or MBE+D could be combined with the transformation of Theorem 3.6 and the transformation satisfying I) and II), to show that $B_1 \in \mathrm{P}$. Recalling the definition of an approximation algorithm for MBE and MBE+D (beginning of Section III), a detail that must be checked (to meet condition (2) in the definition) is that, for each constant c_0, there is a constant n_0' such that

$$(q^{\frac{1}{4}-\varepsilon'} - 1)M_E(G, p, q) \geq c_0, \qquad \text{for all } n \geq n_0'.$$

In the reduction, we choose $q > n$. Because $q^{\frac{1}{4}-\varepsilon'}$ increases as n increases, and $M_E(G, p, q) > 0$, it is clear that such an $n_0' \geq n_0$ exists. Because B_1 is NP-complete, this contradicts the assumption $\mathrm{P} \neq \mathrm{NP}$.

The idea behind the construction of G follows. Assume, for simplicity, that $K = 2^p$. Let $n = |V|$ and $m = |E|$. The DIF G generates codewords having an η-field where $|\eta| = \lceil \log n \rceil$ and a ξ-field where $|\xi| = m$. For each vertex v of H there is a codeword y_v where $\eta(y_v)$ is the name of v and the jth bit of $\xi(y_v)$ is 1 iff the jth edge is incident on v. On the one hand, if $\alpha(H) \geq K$, that is, if H has an independent set I with $|I| = K = 2^p$, let the block codebook contain those y_v with $v \in I$. If we look at any fixed "column" of ξ, where the rows are the codewords in the codebook, we see at most one 1 because I is an independent set. On the other hand, if $\alpha(H)$ is "small" (precisely, $\alpha(H) < K/n^{\frac{1}{2}-\varepsilon}$), then, for any block codebook $\mathcal{C}$, consider the subgraph H' induced by the v such that $y_v \in \mathcal{C}$. So $\alpha(H')$ is also "small." It then follows from Lemma 3.7 that H' must contain a "large" number of edges. So now, there are a "large" number of columns of ξ that have two 1's when the rows are restricted to those $y_v \in \mathcal{C}$. The final step is to relate the minimum size of an encoder for the codebook to the number of columns containing two 1's, when the rows are restricted to the codewords in the codebook. The idea here is that if $y_v \in \mathcal{C}$ then in the encoder circuit, all the bit positions (columns) of ξ having a 1 in the y_v-row can be outputs of the same gate, provided that this is the only 1 in each such column. However, if the jth column (bit position) of ξ contains two 1's, and if j' is any other column, then there must be a row that contains a 1 in one of the columns j or j' but not the other (because H has no multiple edges between the same pair of vertices). Thus, in this case, the

[3]It should be mentioned that these papers actually consider the problem of approximating $\omega(H)$, the maximum size of a clique in H; a set $C \subseteq V$ is a *clique* if each pair of distinct vertices in C are connected by an edge. Note that $\omega(H) = \alpha(\overline{H})$ where $\overline{H}$ is the complement of H (if $H = (V, E)$ then $\overline{H} = (V, \overline{E})$ where, for each pair of distinct vertices v and w, $\{v, w\}$ is an edge in $\overline{E}$ iff $\{v, w\}$ is not an edge in E). Therefore, any nonapproximation result for ω holds also for α.

[4]The algorithm $\mathcal{A}'$ that solves B_1 works as follows. Given input u, $\mathcal{A}'$ computes $f(u) = (H, K)$ and gives input H to $\mathcal{A}$; letting I be the independent set returned by $\mathcal{A}$, the original input u is a "yes" instance of B_1 iff $|I| \geq K/n^{\frac{1}{2}-\varepsilon}$.

jth and j'th bits of ξ must take their values from two different gates.

Let $n = |V|$ and $m = |E|$. Let $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. We can assume $n \le m < n^2$ because H is connected. We can assume $K \le n$ because $\alpha((V, E)) \le |V|$. Choose the integer d to satisfy $K \le 2^d < 2K$, and let $c = 2^d - K$. Note that $0 \le c < K \le n$. Let $\ell = \lceil \log(n + c + 1) \rceil$. Let $p = d + k$ and $q = k + \ell + m + 1$, where k is chosen large enough that $p/\theta(q) \ge R$. As in the proof of Theorem 3.1, we can take $k = O(q - p)$, so $k = O(\ell + m) = O(m)$.

For $y = 2y'$ where $y' \in \{0, 1\}^{q-1}$, write

$$y = 2\beta\eta\xi \tag{6}$$

where $|\beta| = k$, $|\eta| = \ell$, and $|\xi| = m$, and define the subwords $\beta(y)$, $\eta(y)$, $\xi(y)$ in the obvious way. The special DIF G is constructed so that, if y, as in (6), is generated from s and $|y| = q$, then y satisfies either

1) $\beta(y) \in \{0, 1\}^k$, if $i = \mathrm{num}(\eta(y))$ then $1 \le i \le n$ and, for all $1 \le j \le m$, $\xi_j(y) = 1$ iff the edge e_j has the vertex v_i as one of its endpoints (*codewords of the first type*); or
2) $\beta(y) \in \{0, 1\}^k$, $n+1 \le \mathrm{num}(\eta(y)) \le n+c$, and $\xi(y) = 0^m$ (*codewords of the second type*).

The constraints on $\eta(y)$ ensure that no y is of both types. Note that $\xi(y)$ is a function of $\eta(y)$. Therefore, each codeword $y = 2y'$ is uniquely identified by $\beta(y)$ and $\eta(y)$. Note also that the only role of β in this proof is "padding" to increase the rate to R; that is, for all β, β', η, and ξ, $2\beta\eta\xi$ is a codeword iff $2\beta'\eta\xi$ is a codeword. It is easy to see that there is a special DIF G having this property such that $|G| = O(nm)$.

To see that (G, p, q) is block feasible, note that the number of codewords (of both types) is

$$2^k(n + c) = 2^k(n + 2^d - K) \ge 2^k 2^d = 2^p.$$

Let $K' = CK \log n$, where the constant C will be chosen shortly. It remains to show that conditions I) and II) in the first paragraph of the proof hold (for some constant C).

We first show I). Assume that $\alpha(H) \ge K$. Let I be an independent set in H with $|I| = K$. For each i such that $v_i \in I$ and each $\beta \in \{0, 1\}^k$, the block codebook $\mathcal{C}$ contains the codeword y of the first type where $\beta(y) = \beta$, $\eta(y)$ is such that $\mathrm{num}(\eta(y)) = i$, and $\xi(y)$ is determined by $\eta(y)$. In addition, $\mathcal{C}$ contains all $c2^k$ codewords of the second type. Therefore,

$$|\mathcal{C}| = (K + c)2^k = 2^d 2^k = 2^p.$$

Write the p-bit dataword x as $x = \beta(x)\chi(x)$ where $|\beta(x)| = k$ and $|\chi(x)| = d$. The x's with $0 \le \mathrm{num}(\chi(x)) < K$ will be mapped one-to-one onto the set of codewords of the first type that correspond to vertices in I. Formally, let

$$\iota : \{0, 1, \ldots, K-1\} \to \{1, 2, \ldots, n\}$$

be such that $I = \{v_{\iota(0)}, v_{\iota(1)}, \ldots, v_{\iota(K-1)}\}$ and

$$\iota(0) < \iota(1) < \cdots < \iota(K-1).$$

Extend ι to be from $\{0, 1, \ldots, 2^d - 1\}$ to $\{1, 2, \ldots, n + c\}$ by defining $\iota(i) = i + n + 1 - K$ for $K \le i < 2^d$ (recall that $c = 2^d - K$, so $\iota(2^d - 1) = n + c$). The map ι is still one-to-one after the extension because $\iota(i) \ge n + 1$ if $i \ge K$. For $0 \le \mathrm{num}(\chi(x)) < 2^d$, the encoder $\mathcal{E}$ maps the dataword $x = \beta(x)\chi(x)$ to the codeword $y \in \mathcal{C}$ where $\beta(y) = \beta(x)$, $\eta(y)$ is such that $\mathrm{num}(\eta(y)) = \iota(\mathrm{num}(\chi(x)))$, and $\xi(y)$ is determined by $\eta(y)$.

To place a small enough upper bound on the size of an encoder circuit for this $\mathcal{E}$, the following claim is important. It is the formalization of the statement in the proof outline that if we look at any column j of ξ, where the rows are the codewords in the codebook, then we see at most one 1. More precisely (because there can be many values of β), there is at most one $\eta^{(1)}$ such that a row y with $\eta(y) = \eta^{(1)}$ has a 1 in the jth column.

Claim 6: Fix j with $1 \le j \le m$. Either a) $\xi_j(y) = 0$ for all $y \in \mathcal{C}$, or b) there is an $\eta^{(1)}$ with $1 \le \mathrm{num}(\eta^{(1)}) \le n$ such that, for each $y' \in \mathcal{C}$, $\xi_j(y') = 1$ iff $\eta(y') = \eta^{(1)}$ (this $\eta^{(1)}$ is necessarily unique).

Proof: Recall that $\xi_j(y) = 0$ for all y of the second type. If there does not exist a $y \in \mathcal{C}$ of the first type with $\xi_j(y) = 1$, then obviously case a) holds. Say then that there exists $y^{(1)} \in \mathcal{C}$ and $\eta^{(1)} \in \{0, 1\}^\ell$ where $1 \le \mathrm{num}(\eta^{(1)}) \le n$, $\eta(y^{(1)}) = \eta^{(1)}$, and $\xi_j(y^{(1)}) = 1$. Clearly, for each $y' \in \mathcal{C}$, if $\eta(y') = \eta^{(1)}$ then $\xi_j(y') = 1$, because $\xi(y)$ is a function of $\eta(y)$.

We show that if $\xi_j(y) = 1$ then $\eta(y) = \eta^{(1)}$. Assume otherwise that there exist $y^{(2)} \in \mathcal{C}$ and $\eta^{(2)}$ such that $y^{(1)} \ne y^{(2)}$, $\eta^{(1)} \ne \eta^{(2)}$, $\xi_j(y^{(2)}) = 1$, and $\eta(y^{(2)}) = \eta^{(2)}$. For $i = 1, 2$, the definition of codewords of the first type associates $y^{(i)}$ with the vertex $v^{(i)} \stackrel{\mathrm{def}}{=} v_z$ where $z = \mathrm{num}(\eta(y^{(i)}))$. Because $\eta(y^{(1)}) \ne \eta(y^{(2)})$ and $y^{(1)}$ and $y^{(2)}$ both belong to $\mathcal{C}$, it follows that $v^{(1)}$ and $v^{(2)}$ are unequal and both belong to I. By definition of the set of codewords, $\xi_j(y^{(1)}) = \xi_j(y^{(2)}) = 1$ implies that the vertices $v^{(1)}$ and $v^{(2)}$ are connected by the edge e_j. This contradicts that I is an independent set. □

We can now describe the encoder circuit. Let x denote the input dataword and y the output codeword in $\mathcal{C}$. Part of the encoder computes $\beta(y) = \beta(x)$ using no gates. Another part computes $\eta(y)$ from $\chi(x)$ by following the map ι, that is, $\mathrm{num}(\eta(y)) = \iota(\mathrm{num}(\chi(x)))$; this can be done using $O(\ell 2^d)$ gates. (In general, any function mapping $\{0, 1\}^a$ to $\{0, 1\}^b$ can be computed by a circuit of size $O(b2^a)$; this follows from [29, Lemma 5.1].) Another part of the encoder computes bits b_i for $0 \le i < 2^d$, such that $b_i = 1$ iff $\mathrm{num}(\chi(x)) = i$. Each b_i can be computed using $O(d)$ gates because $|\chi(x)| = d$, for a total of $O(d2^d)$ gates. We argue, using Claim 6, that $\xi(y)$ can be computed from the b_i using no additional gates. Fix a coordinate j of ξ with $1 \le j \le m$. If case a) of Claim 6 holds then $\xi_j(y)$ is the constant 0. If case b) holds, let $\eta^{(1)} \in \{0, 1\}^\ell$ be such that, for each $y' \in \mathcal{C}$, $\xi_j(y') = 1$ iff $\eta(y') = \eta^{(1)}$. Recalling that ι is one-to-one, define i by $\iota(i) = \mathrm{num}(\eta^{(1)})$. It follows that $\xi_j(y) = b_i$.

The decoder circuit computes $\beta(x) = \beta(y)$ using no gates, and it computes $\chi(x)$ from $\eta(y)$ using $O((d+\ell)2^d)$ gates. To see how this is done, first note that there are 2^d ℓ-bit words that can appear as $\eta(y)$ for $y \in \mathcal{C}$. Let these words be $\eta^{(i)}$ for $0 \le i < 2^d$, in order of increasing $\mathrm{num}(\eta^{(i)})$. The circuit first computes bits b'_i $(0 \le i < 2^d)$ such that $b'_i = 1$ iff $\eta(y) = \eta^{(i)}$; this can be done with $O(\ell 2^d)$ gates because for each fixed i it takes at most $O(|\eta(y)|) = O(\ell)$ gates to determine whether $\eta(y)$ equals the fixed value $\eta^{(i)}$. The circuit then computes $\chi(x)$ such that $\mathrm{num}(\chi(x)) =$ the unique i such that $b'_i = 1$; this can be done

using $O(d2^d)$ gates as follows. The jth bit of $\chi(x)$ $(1 \le j \le d)$ is the OR of b'_i over all i $(0 \le i < 2^d)$ such that the jth bit of i is 1.

The total size of this encoder circuit and decoder circuit is

$$O((d+\ell)2^d) = O(K \log n).$$

Choose C large enough that $CK \log n$ is an upper bound on the total size, and let $K' = CK \log n$.

It is shown next that $M_E(G, p, q) > 0$. Let $\mathcal{C}$ be an arbitrary block codebook for (G, p, q). We may assume that $d \ge 3$, so $p \ge k+3$. By definition of the codewords, for each j with $1 \le j \le m$, $\xi_j(y) = 1$ for exactly 2^{k+1} codewords y and, therefore, for at most 2^{k+1} codewords $y \in \mathcal{C}$. It follows that ξ_j, viewed as a function of x, cannot be the constant function 1 and cannot be a projection function x_i for some $1 \le i \le p$, because each of these functions equals 1 for at least $2^{p-1} > 2^{k+1}$ datawords. Because the number of codewords of the second type is $c2^k < 2^d 2^k$, we can assume that $\mathcal{C}$ contains a codeword of the first type. Let $y \in \mathcal{C}$ and j be such that, if $i = \mathrm{num}(\eta(y))$, then $1 \le i \le n$ and edge e_j is incident on vertex v_i. Then $\xi_j(y) = 1$, so ξ_j cannot be the constant function 0. Therefore, any encoder circuit for $\mathcal{C}$ must have at least one gate.

This completes the proof of I).

To prove II), let $g = n^{\frac{1}{2}-\varepsilon}$, and assume that $\alpha(H) < K/g$. Fix an arbitrary block codebook $\mathcal{C}$ for (G, p, q). Because $|\mathcal{C}| = 2^p = 2^k 2^d$, it follows that $\{\eta(y) \mid y \in \mathcal{C}\}$ contains at least 2^d distinct words. Because $1 \le \mathrm{num}(\eta(y)) \le n+c$ for all codewords y, if $S \stackrel{\mathrm{def}}{=} \{\mathrm{num}(\eta(y)) \mid y \in \mathcal{C} \text{ and } 1 \le \mathrm{num}(\eta(y)) \le n\}$, then $|S| \ge 2^d - c = K$. Consider the subgraph H' of H induced by the set V' of vertices defined by $V' = \{v_i \mid i \in S\}$. Thus, $H' = (V', E')$ where (u, v) is an edge of H' iff $u, v \in V'$ and (u, v) is an edge of H. Because any independent set in H' is also an independent set in H, it follows that

$$\alpha' \stackrel{\mathrm{def}}{=} \alpha(H') \le \alpha(H) < K/g.$$

Applying Lemma 3.7 to H' with $n' = K$ and $\alpha' < K/g$, we see that the number m' of edges of H' satisfies i) $m' > K(g-1)/2$.

Below we argue that if $\hat{\mathcal{E}}$ is an encoder circuit for $\mathcal{C}$, then $|\hat{\mathcal{E}}| \ge m'$. Because $\mathcal{C}$ is an arbitrary block codebook for (G, p, q), this implies that ii) $M_E(G, p, q) \ge m'$. We first show how this is sufficient to prove that $M_E(G, p, q) > q^{\frac{1}{4}-\varepsilon'} K'$ as required by II). Because $q = k+\ell+m+1$ and $k+\ell+1 = O(m)$, there is a constant C' such that iii) $q \le (C')^2 m < (C'n)^2$. Recall that iv) $K' = CK \log n$ for some constant C. By i) and ii)

$$M_E(G, p, q) \ge m' > K(g-1)/2 = K(n^{\frac{1}{2}-\varepsilon} - 1)/2.$$

Therefore,

$$\begin{aligned} M_E(G, p, q) &> K\left(n^{\frac{1}{2}-\varepsilon} - 1\right)\Big/2 \\ &> (K/(2C'))\left(q^{\frac{1}{4}-\varepsilon/2} - C'\right) && \text{by iii)} \\ &= (K'/(2CC' \log n))\left(q^{\frac{1}{4}-\varepsilon/2} - C'\right) && \text{by iv)} \\ &> q^{\frac{1}{4}-\varepsilon'} K' \end{aligned}$$

where the last inequality holds for all sufficiently large q, because $q > n$ and ε was chosen so that $\varepsilon/2 < \varepsilon'$. Because $q > n$, this inequality holds for all sufficiently large n.

It remains only to show that if $\hat{\mathcal{E}}$ is an encoder circuit for $\mathcal{C}$, then $|\hat{\mathcal{E}}| \ge m'$. Note that if $e_j \in E'$ and $\xi_j(x)$ is viewed as a function of x computed by the encoder, then $\xi_j(x) = 1$ for between 2 and 2^{k+1} datawords x: letting the endpoints of e_j be v_{i_1} and v_{i_2}, these x are those that the encoder maps to y where either $i_1 = \mathrm{num}(\eta(y))$ or $i_2 = \mathrm{num}(\eta(y))$. Because H contains no multiple edges between the same two vertices, and we may assume $d \ge 3$, it follows that if $1 \le j_1 < j_2 \le m$ and $e_{j_1}, e_{j_2} \in E'$, then $\xi_{j_1}(x)$ is not the same function as $\xi_{j_2}(x)$, and neither is a constant function nor a projection function. Thus, we need at least $|E'| = m'$ gates to compute the m' different functions $\xi_j(x)$ for $e_j \in E'$. □

Remark: It is clear from the proof that if $\frac{1}{2}$ can be replaced by a larger constant C in Theorem 3.6, then $\frac{1}{4}$ can be replaced by $C/2$ in Theorem 3.2 part 2). For example, Håstad [42] has shown for each $\varepsilon > 0$ that the independence number cannot be approximated within the factor $n^{1-\varepsilon}$ in polynomial time, assuming that NP $\neq$ ZPP, where ZPP is the class of decision problems that can be solved in expected polynomial time by a probabilistic algorithm. This assumption is stronger than P $\neq$ NP because it is possible that NP $=$ ZPP and P $\neq$ NP. (However, NP $=$ ZPP implies that the polynomial hierarchy collapses.) Thus, assuming that NP $\neq$ ZPP and $0 < \varepsilon < \frac{1}{2}$, MBE and MBE+D cannot be approximated within the factor $q^{\frac{1}{2}-\varepsilon}$ in polynomial time.

D. Proof of Theorem 3.3

Here we prove that the MBE+D problem is Σ_2^p-hard under $\le_{\mathrm{PR}}$, in the case that the instance contains target upper bounds K_1 for encoder circuit size and K_2 for decoder circuit size. We will take $K_2 = 0$ in the reduction; thus, the question is whether there is an encoder of size $\le K_1$ such that its decoder can be computed by a circuit of size zero.

We use a recent result of Umans [45] that gives a nonapproximability result for a Σ_2^p-complete problem. The problem, called SUCCINCT SET COVER (SSC), is defined as follows. For a formula F of the variables $\{v_1, \ldots, v_n\}$, let $\mathrm{SAT}(F) \subseteq \{0, 1\}^n$ denote the set of satisfying assignments of F. Think of a formula F as a succinct description of the set $\mathrm{SAT}(F)$. An instance of SSC has the form $\mathcal{F} = (F_1, \ldots, F_m)$ where $m \ge 1$, F_r for $1 \le r \le m$ is a 3DNF formula of the variables $\{v_1, \ldots, v_n\}$, and

$$\bigcup_{1 \le r \le m} \mathrm{SAT}(F_r) = \{0, 1\}^n.$$

A set $S \subseteq \{1, 2, \ldots, m\}$ is a *set cover for* $\mathcal{F}$ if

$$\bigcup_{r \in S} \mathrm{SAT}(F_r) = \{0, 1\}^n.$$

Define $\mathrm{MinSC}(\mathcal{F})$ (minimum set cover of $\mathcal{F}$) to be the minimum of $|S|$ over all set covers S for $\mathcal{F}$. By duplicating formulas and disjuncts if necessary, it is convenient to assume that $n \le m$ and F_r contains exactly m disjuncts for all $1 \le r \le m$.

Theorem 3.8 [45]: There is a constant $\varepsilon > 0$, a Σ_2^p-complete problem B_2, and a polynomial-time transformation f mapping instances of B_2 to pairs $(\mathcal{F}, K)$ where $\mathcal{F}$ is an instance of SSC and $K \in \boldsymbol{N}^+$ such that

1) if $u \in B_2$ then $f(u) = (\mathcal{F}, K)$ where $\mathrm{MinSC}(\mathcal{F}) \le K$; and

2) if $u \notin B_2$ then $f(u) = (\mathcal{F}, K)$ where $\mathrm{MinSC}(\mathcal{F}) > m^{\varepsilon}K$ (where m is the number of formulas in $\mathcal{F}$).

We can assume that $K \geq \log m$ in this theorem. If $K < \log m$, then increase K to $\lceil \log m \rceil$ and replace ε by some ε' with $0 < \varepsilon' < \varepsilon$.

Theorem 3.3: Let $R < \frac{1}{2}$. MBE&D (rate $\geq R$, FM) is Σ_2^p-hard under $\leq_{\mathrm{PR}}$.

Proof: Recalling the definition of $\leq_{\mathrm{PR}}$ (Section II-D2) we describe a polynomial-time transformation that takes two inputs, one of the form $(\mathcal{F}, K)$ and the second a $w \in \{0, 1\}^+$, and produces an output of the form $(G, p, q, K', 0)$. Because the size of the decoder must be zero, the question is whether (G, p, q) has an encoder circuit of size $\leq K'$ whose decoder is a projection function. The idea of our proof is to use the "random" input w in the reduction to give a long random "name" to each F_i, and embed these names in the set of codewords in such a way that the size of a set cover $\mathcal{F}'$ is related to the number of these long random names that must be outputs of any encoder whose decoder is a projection function. A counting argument shows that a "small" encoder with a "small" number of inputs cannot output a "large" number of long random names.[5] Thus, a "small" set cover ($\mathrm{MinSC}(\mathcal{F}) \leq K$) implies a "small" encoder, and a "large" set cover ($\mathrm{MinSC}(\mathcal{F}) > m^{\varepsilon}K$) requires a "large" encoder.

Let $\mathcal{F} = (F_1, \ldots, F_m)$ where each F_r is a 3DNF formula of the variables $\{v_1, \ldots, v_n\}$, where $n \leq m$. Let

$$\ell = \lceil \log(m+1) \rceil$$
$$z = \lceil c(m+n) \rceil$$
$$p = m + n$$

and

$$q = m + n + z + \ell + 1$$

where $c > 1$ is chosen so that $p/\theta(q) \geq R$ for all sufficiently large n. To see that such a choice of c is possible, note that $\ell + 1$ is negligible compared to m and recall that $\theta(q)/q$ approaches 1 as $q \to \infty$; so $p/\theta(q)$ approaches

$$p/(q - \ell - 1) = (m+n)/(m + n + \lceil c(m+n) \rceil)$$

which approaches $1/(1+c)$. So it suffices to choose $c > 1$ such that $1/(1+c) > R$. The "random" input w to the $\leq_{\mathrm{PR}}$ transformation has length mz. Let $M_{E,0}(G, p, q)$ be the minimum of $\Xi(\mathcal{E})$ over all $(\mathcal{E}, \mathcal{D})$ such that $(\mathcal{E}, \mathcal{D})$ is an encoder/decoder for (G, p, q) and $\Xi(\mathcal{D}) = 0$. The transformation from $((\mathcal{F}, K), w)$ to (G, p, q, K') has the following properties, for some $n_0 > 0$:

I) if $\mathrm{MinSC}(\mathcal{F}) \leq K$, then $M_{E,0}(G, p, q) \leq K'$ for all $w \in \{0, 1\}^{mz}$;

II) if $\mathrm{MinSC}(\mathcal{F}) > m^{\varepsilon}K$ and $n \geq n_0$, then $M_{E,0}(G, p, q) > K'$ for at least one half of the $w \in \{0, 1\}^{mz}$.

Moreover, G is a special DIF having one synchronizing state s, and (G, p, q) is block feasible. Therefore, combining the polynomial-time transformation f from u to $(\mathcal{F}, K)$ in Theorem 3.8 with this transformation from $((\mathcal{F}, K), w)$ to (G, p, q, K') gives a transformation showing that $B_2 \leq_{\mathrm{PR}}$ MBE&D.

For $y = 2y'$ where $y' \in \{0, 1\}^{q-1}$, write

$$y = 2\beta\psi\eta\nu \tag{7}$$

where $|\beta| = m$, $|\psi| = z$, $|\eta| = \ell$, and $|\nu| = n$. The field ψ contains a "long random name" of one of the formulas in $\mathcal{F}$.

Let $((\mathcal{F}, K), w)$ be an input to the transformation where $|w| = mz$. For $1 \leq r \leq m$, let $(D_{r,1}, D_{r,2}, \ldots, D_{r,m})$ be the disjuncts of F_r. Write $w = w_1 w_2 \cdots w_m$ where $|w_r| = z$ for $1 \leq r \leq m$. The special DIF G is constructed so that, if y as in (7) is generated from s, then y satisfies either

1) $\beta(y) = 0^m$, and there exist r, j with $1 \leq r, j \leq m$ such that $\psi(y) = w_r$, $\mathrm{num}(\eta(y)) = j$, and $\nu(y)$ satisfies the disjunct $D_{r,j}$; or

2) $\beta(y) \in \{0, 1\}^m - \{0^m\}$, $\psi(y) = 0^z$, $\eta(y) = 0^{\ell}$, and $\nu(y) \in \{0, 1\}^n$.

(G, p, q) is block feasible because the number of codewords of the first (resp., second) type is $m^2 2^{n-3}$ (resp., $(2^m - 1)2^n$), so the total number is at least $2^{m+n} = 2^p$ if $m \geq 3$.

The basic idea is that if $\mathrm{MinSC}(\mathcal{F}) \leq K$, and $S \subseteq \{1, 2, \ldots, m\}$ is a set cover for $\mathcal{F}$ with $|S| = K$, then we can build a block codebook using only codewords y such that $\psi(y) = w_r$ for $r \in S$, and this codebook has a "small" (size $\leq K'$) encoder circuit and a corresponding decoder circuit of size zero. But if $\mathrm{MinSC}(\mathcal{F}) > m^{\varepsilon}K$, then any block codebook (having a decoder of size zero) must contain a y with $\psi(y) = w_r$ for more than $m^{\varepsilon}K$ different values of r. The proof is completed by showing that, for at least half of the choices for $w = w_1 w_2 \ldots w_m$, any circuit that can output this many different w_r's must have size larger than K'.

We show that I) and II) hold, for some K' computable in polynomial time. We first show I), and in the process define K'. Assume that $\mathrm{MinSC}(\mathcal{F}) \leq K$, and let $S \subseteq \{1, 2, \ldots, m\}$ be a set cover for $\mathcal{F}$ with $|S| = K$. It follows that for each $\alpha \in \{0, 1\}^n$, there exists an $r \in S$ and a j with $1 \leq j \leq m$ such that $D_{r,j}$ is satisfied by α. The block codebook $\mathcal{C}$ contains, for each $\alpha \in \{0, 1\}^n$, a codeword y of the first type where $\beta(y) = 0^m$, $\psi(y) = w_r$, $\mathrm{num}(\eta(y)) = j$, $\nu(y) = \alpha$, and α satisfies $D_{r,j}$. In addition, $\mathcal{C}$ contains all codewords of the second type. So $|\mathcal{C}| = 2^n + (2^m - 1)2^n = 2^p$, and the decoder of size zero maps the codeword $\beta\psi\eta\nu$ to the dataword $\beta\nu$.

We now place an upper bound on the size of an encoder for $\mathcal{C}$. Write a dataword x as $\beta(x)\nu(x)$ where $|\beta(x)| = m$ and $|\nu(x)| = n$. The encoder computes $\beta(y) = \beta(x)$ and $\nu(y) = \nu(x)$ using no gates. If $\beta(x) \neq 0^m$, then $\psi(y) = 0^z$ and $\eta(y) = 0^{\ell}$; this can be implemented using $O(m + z + \ell)$ gates. We show how the encoder computes $\psi(y)$ and $\eta(y)$ from $\nu(x)$ when $\beta(x) = 0^m$. It first computes bits $b_{r,j}$ for $r \in S$ and $1 \leq j \leq m$, where $b_{r,j} = 1$ iff $\nu(x)$ satisfies $D_{r,j}$; because each $D_{r,j}$ has at most three literals, these bits can be computed using $O(mK)$ gates. Because S is a set cover, for each $\nu(x) \in \{0, 1\}^n$ there is at least one choice for r and j such that $b_{r,j} = 1$. The encoder then computes bits $b'_{\tilde{r},\tilde{j}}$ for $\tilde{r} \in S$ and $1 \leq \tilde{j} \leq m$ such that $b'_{\tilde{r},\tilde{j}} = 1$ iff $(\tilde{r}, \tilde{j})$ is the lexicographically smallest (r, j) such that $b_{r,j} = 1$. The only property of the bits $b'_{r,j}$ used is that for each $\nu(x)$: $b'_{r,j} = 1 \Rightarrow b_{r,j} = 1$,

[5] An argument of this type (that the information in the output of a deterministic device cannot be more than the information in the input plus the information in the description of the device) was used previously by Vazirani and Vazirani [35].

and there is exactly one (r, j) such that $b'_{r,j} = 1$. The bits $b'_{r,j}$ can be computed using a number of gates proportional to the number of bits $b_{r,j}$, that is, $O(mK)$ gates. The encoder then computes bits $d_r = \bigvee_{j=1}^{m} b'_{r,j}$ for $r \in S$ and $e_j = \bigvee_{r \in S} b'_{r,j}$ for $1 \leq j \leq m$; this is done using $O(mK)$ gates. Thus, for each $\nu(x) \in \{0, 1\}^n$, there is exactly one $r \in S$ with $d_r = 1$ and exactly one $1 \leq j \leq m$ with $e_j = 1$, and this unique (r, j) has the property that $\nu(x)$ satisfies $D_{r,j}$. The encoder now computes $\psi(y)$ by $\psi(y) = w_r$ where $d_r = 1$; because $|w_r| = z$ for all r and there are $|S| = K$ bits d_r for $r \in S$, $\psi(y)$ can be computed with $O(zK)$ additional gates. The encoder computes $\eta(y)$ by $\mathrm{num}(\eta(y)) = j$ where $e_j = 1$; because $|\eta(y)| = \ell$ and there are m bits e_j for $1 \leq i \leq m$, $\eta(y)$ can be computed with $O(m\ell)$ additional gates. The total number of gates in the encoder circuit is $O(mK + zK + m\ell)$. Because $n \leq m$, $z = O(m + n)$, $\ell = O(\log m)$, and $\log m \leq K$, the total size of the encoder is $O(mK)$. Choose the constant C such that the size of this encoder is at most CmK, and let $K' = CmK$. So I) holds for this choice of K'.

We now prove that II) holds for this choice of K' and for all sufficiently large n (and recall that $m \geq n$). Assume $\mathrm{MinSC}(\mathcal{F}) > m^{\varepsilon}K$. For $w \in \{0, 1\}^{mz}$, let the transformation map $((\mathcal{F}, K), w)$ to (G_w, p, q, K') (p, q, K' do not depend on w). Let $\mathcal{C}_w$ be an arbitrary block codebook for (G_w, p, q), and let $(\hat{\mathcal{E}}_w, \hat{\mathcal{D}}_w)$ be an arbitrary encoder/decoder circuit pair for $\mathcal{C}_w$, such that $\hat{\mathcal{D}}_w$ has size zero. The proof has two parts. The first part is to show that for each fixed $w = w_1 \cdots w_m$, there are at least $m^{\varepsilon}K$ values of r such that a codeword y with $\psi(y) = w_r$ is an output $\hat{\mathcal{E}}_w(x)$ for some input dataword x. Intuitively, for a randomly chosen w, these $m^{\varepsilon}K$ w_rs contain $(m^{\varepsilon}K)z$ bits of information. The intuition behind the second part is that for a randomly chosen w, the description of $\hat{\mathcal{E}}_w$ must also contain about $m^{\varepsilon}zK$ bits of information, and so the size of $\hat{\mathcal{E}}_w$ is about $m^{\varepsilon}zK > m^{\varepsilon}mK$, which exceeds CmK for all sufficiently large m.

The formal statement of the first part is as follows.

Claim 7: For each $w = w_1 \cdots w_m \in \{0, 1\}^{mz}$ there is a set $S_w \subseteq \{1, 2, \ldots, m\}$ with $|S_w| > m^{\varepsilon}K$, such that for each $r \in S_w$ there exists a dataword $x_r \in \{0, 1\}^p$ such that $\psi(\hat{\mathcal{E}}_w(x_r)) = w_r$.

Proof: We argue as in the proof of Theorem 3.1 that the range Φ of the output mapping of $\hat{\mathcal{D}}_w$ contains exactly the coordinates of $\beta(y)$ and $\nu(y)$. First note that Φ cannot contain any coordinate of $\psi(y)$ or $\eta(y)$ because each such coordinate takes the value 1 for at most $m^2 2^n$ codewords (the codewords of the first type), and $m^2 2^n < 2^{p-1}$ if $m^2 < 2^{m-1}$, which holds for all sufficiently large m. The conclusion about Φ now holds as before because $p = m + n$.

It follows that for each $\alpha \in \{0, 1\}^n$, there exists a $y \in \mathcal{C}_w$ such that $\beta(y) = 0^m$ and $\nu(y) = \alpha$. Let $S_w = \{r \,|\, \exists y \in \mathcal{C}_w$ with $\psi(y) = w_r$ and $\beta(y) = 0^m\}$. It follows by the definition of the codewords that S_w is a set cover for $\mathcal{F}$. The claim is now immediate from the assumption $\mathrm{MinSC}(\mathcal{F}) > m^{\varepsilon}K$. □

The formal statement of the second part is as follows.

Claim 8: For at least half of the $w \in \{0, 1\}^{mz}$, the size of $\hat{\mathcal{E}}_w$ is greater than K'.

Proof: Assume for contradiction that there is a set $W \subseteq \{0, 1\}^{mz}$ with $|W| = 2^{mz-1}$ such that the size of $\hat{\mathcal{E}}_w$ is at most K' for all $w \in W$. Let $M = \lceil m^{\varepsilon}K \rceil$. We now show that an arbitrary word $w = w_1 \cdots w_m \in W$ can be identified by giving four pieces of information. We also give an upper bound u_i on the number of choices for the ith piece of information for $i = 1, 2, 3, 4$.

1) A set $S_w \subseteq \{1, 2, \ldots, m\}$ with $|S_w| = M$, such that for each $r \in S_w$ there exists an $x_r \in \{0, 1\}^p$ such that $\psi(\hat{\mathcal{E}}_w(x_r)) = w_r$. A set of this type exists by Claim 7. The number of sets of this type is at most $u_1 = m^M$.
2) The M datawords x_r for $r \in S_w$. The number of choices is $u_2 = 2^{pM}$.
3) The circuit $\hat{\mathcal{E}}_w$, whose size is at most K' by assumption, restricted to its outputs in $\psi(y)$. A rough upper bound on the number of circuits with p inputs, z outputs, and size K' is $u_3 = 16^{K'}(p + K' + 2)^{2K'+z}$.
4) The value of w_r for each $r \notin S_w$. The number of choices is $u_4 = 2^{(m-M)z}$.

We will have a contradiction if $u_1 u_2 u_3 u_4 < 2^{mz-1}$ for all sufficiently large m. In the remainder of the proof, the verification of this inequality is outlined.

Substituting the values of u_i, taking logs, and doing a little rearranging, the inequality becomes

$$Mz - M(p + \log m) > 4K' + (2K' + z)\log(p + K' + 2) + 1.$$

Because z, $p = O(K')$, it is sufficient to show that for each C' there is an m_0 such that for all $m \geq m_0$

$$Mz - M(p + \log m) > C'K' \log K'.$$

Recall that $z \geq c(m + n) = cp$ where $c > 1$, so

$$Mz - M(p + \log m) = \Omega(Mz).$$

Therefore, it is sufficient to show that, for each constant C'', $Mz > C''K' \log K'$ for all sufficiently large m. Recalling that $z \geq m$ and substituting the values of M and K', it is sufficient that

$$m^{1+\varepsilon}K > C''CmK \log(CmK).$$

Because $K \leq m$, it is sufficient that $m^{\varepsilon} > 2C''C \log(Cm)$. For any choice of constants C, C'', there is an m_0 such that this holds for all $m \geq m_0$. □

This completes the proof of Theorem 3.3. □

E. Some Additional Technical Machinery

In the proof of Theorem 3.4 and Lemma 4.4, we will need DIFs that have more than one synchronizing state. In this case, Claim 3 does not hold, so we must consider codewords of the form $y'2y''$ where possibly $|y'|, |y''| > 0$. If $y''y'$ is "important" in the proof, then the proof becomes more complicated because we must deal with concatenations of the form $y'_1 2 y''_1 y'_2 2 y''_2$ where the important parts are "split." To avoid bad splits, we use special DIFs G such that, if y is a q-bit word that is generated from a synchronizing state, then y has the form

$$y = 2\gamma I(y)\gamma$$

where $\gamma \in \{0, 1\}$. We call $I(y)$ the *important part of* y. From each synchronizing state, G produces a "2," remembers the next bit that is produced (the bit γ), and after producing $q-3$ more bits (depending on other details of G), the next bit produced must be γ.

The purpose of introducing the matching γs is to avoid rotations that split the important part of y. Let $\mathcal{C}$ be a block codebook for (G, p, q), for an arbitrary p, q. For ease of explanation, we work with G, $\mathcal{C}$, and codewords $y \in \mathcal{S}_q(G)$, rather than $\theta(G)$, $\theta(\mathcal{C})$, and $\theta(y)$. The interested reader can check that everything translates in a straightforward way to the "θ world." Call a rotation $y'2y''$ of $y = 2\gamma I(y)\gamma$ *bad* if $I(y)$ is split in $y'2y''$, formally, if $2 \le |y'| \le |y| - 3$. Recall from Claim 1 that all words in $\mathcal{C}$ have the same rotation. The next claim states that if the rotation of $\mathcal{C}$ is bad, then γ can have only one value.

Claim 9: Let $G, q, \mathcal{C}$ be as above. Suppose that the (common) rotation of words in $\mathcal{C}$ is bad. Then there is a $\gamma_0 \in \{0, 1\}$ such that, for each $y \in \mathcal{C}$, $y = w\gamma_0 2\gamma_0 w'$ for some $w, w' \in \{0, 1\}^+$.

Proof: Let y_1 and y_2 be arbitrary words in $\mathcal{C}$. Because the split is bad, we can write $y_i = w_i\gamma_i 2\gamma_i' w_i'$, for $i = 1, 2$. Then

$$y_1 y_1 = w_1\gamma_1 2\gamma_1' w_1' w_1\gamma_1 2\gamma_1' w_1'$$

and

$$y_1 y_2 = w_1\gamma_1 2\gamma_1' w_1' w_2\gamma_2 2\gamma_2' w_2'$$

are both in $\mathcal{S}(G)$. Because G only produces words of the form $2\gamma I(y)\gamma$ from a synchronizing state, we have $\gamma_1 = \gamma_1'$ and $\gamma_1' = \gamma_2$. □

To use this lemma, we choose the instance (G, p, q, K) so that both values for γ must be used in forming any block codebook for (G, p, q). For example, a reason could be that if only one γ is used then $|\mathcal{C}|$ is too small. It then follows from Claims 1 and 9 that there is an integer ρ such that, for each $y \in \mathcal{C}$, $y = wI(y)w'$ for some w, w' with $|w| = \rho$. In other words, for each $y \in \mathcal{C}$, its important part is a subword. We can then ignore w, w' and work just with the important parts.

F. Proof of Theorem 3.4

Here we prove that the minimum block decoder problem is Σ_3^p-hard under $\le_m^p$. To do this, it is sufficient to show $B \le_m^p$ MBD where the decision problem B is known to be Σ_3^p-hard under $\le_m^p$. We use the following problem, called $\mathrm{QBF}_{3,\exists}$; it is the obvious generalization of the satisfiability problem (which has quantifier structure $\exists$) to the structure $\exists\forall\exists$ ("$\exists$" in the subscript indicates that the leading quantifier is $\exists$). If $u = \{u_1, \ldots, u_n\}$, $v = \{v_1, \ldots, v_n\}$, and $w = \{w_1, \ldots, w_n\}$ are sets of Boolean variables, we let $F(u, v, w)$ denote a Boolean formula whose variables are those in $u \cup v \cup w$. We let $\exists u$ abbreviate $\exists u_1 \cdots \exists u_n$, and, similarly, $\forall v$ and $\exists w$. If $\tilde{u}, \tilde{v}, \tilde{w} \in \{0, 1\}^n$, then $F(\tilde{u}, \tilde{v}, \tilde{w}) = 1$ means that F is satisfied (is true) when the assignment $(\tilde{u}, \tilde{v}, \tilde{w})$ is substituted for the variables (u, v, w).

$\mathrm{QBF}_{3,\exists}$
Instance: A CNF formula $F(u, v, w)$.
Question: $\exists u\, \forall v\, \exists w\, F(u, v, w) = 1$?

It is proved by Stockmeyer [12] and Wrathall [34] that $\mathrm{QBF}_{3,\exists}$ is Σ_3^p-complete (under $\le_m^p$).[6]

Theorem 3.4: Let $R < 1$. MBD (rate $\ge R$, FM) is Σ_3^p-hard under $\le_m^p$.

Proof: We describe a polynomial-time transformation from an arbitrary instance $F(u, v, w)$ of $\mathrm{QBF}_{3,\exists}$ to a (G, p, q) such that G is a special DIF having finite memory, (G, p, q) is block feasible, $p/\theta(q) \ge R$, and $\exists u\, \forall v\, \exists w F(u, v, w) = 1$ iff (G, p, q) has a decoder of size zero.

The rough idea of the proof is given first. Three of the fields of a codeword are μ, ν, ω, all of length n; they are identified with assignments to u, v, w, respectively. Assume for the moment that I) we can construct G so that in every codeword the values of the μ, ν, ω fields correspond to a satisfying assignment of F. Assume also that II) we can ensure that the projection function computed by the decoder "uses" (that is, is dependent on) all of field ν but none of field ω. For a fixed $\tilde{u}$, this is close to $\forall v\, \exists w F(\tilde{u}, v, w) = 1$, that is, for each choice of the bits in field ν there must exist a codeword in the codebook (for some choice of the bits in field ω). A more difficult part of the proof, which we do not attempt to sketch here, is to simulate the quantifier $\exists u$ by the choice of bit positions that the projection function uses in the fields other than ν, ω. Returning to the assumption I) that y is a codeword iff the μ, ν, ω fields of y satisfy F, our solution uses $\mathcal{S}(G)$-concatenability of the codebook. We include another field η in the codewords, which contains a binary number between 1 and $m =$ the number of clauses. When G produces an η field with value j, the next μ, ν, ω fields that are produced must satisfy the jth clause. We must then ensure that for each $1 \le j \le m$ there must be a codeword in the codebook whose η-field is j. This is done by arguing that if there is some j that does not appear in the η-field of any codeword in the codebook, then the codebook does not contain enough codewords to achieve rate $p : q$. Assumption II) is handled by the following observation. Suppose the projection function (the decoder) is a function of the bit positions in the set S ($S \subseteq \{1, 2, \ldots, q\}$ and $|S| = p$); if we restrict the codewords in the codebook to any subset of s bit positions in S ($1 \le s \le |S| = p$), then each s-bit word must appear 2^{p-s} times. The codewords are constructed so that this is violated if S contains some bit position of the field ω or does not contain some bit position of the field ν.

Let C_j for $1 \le j \le m$ be the conjuncts of F; that is, $F = C_1 \wedge \cdots \wedge C_m$. Let $\ell = \lceil \log(m+1) \rceil$, $p = k + 2n + \ell + 1$, and $q = k + 4n + \ell + 3$, where k is chosen such that $k \ge 2n + 2$ and $p/\theta(q) \ge R$. As in the proof of Theorem 3.1, we can take $k = O(q - p) = O(n)$.

The DIF G has m synchronizing states, s_j for $1 \le j \le m$. A q-bit word y generated from a synchronizing state has the form

$$y = 2\gamma\beta\mu_1\mu_1' \cdots \mu_n\mu_n'\nu_1 \cdots \nu_n\omega_1 \cdots \omega_n\eta\gamma' \qquad (8)$$

where $\gamma, \gamma' \in \{0, 1\}$, $\beta \in \{0, 1\}^k$, $\mu_i, \mu_i', \nu_i, \omega_i \in \{0, 1\}$ for $1 \le i \le n$, and $\eta \in \{0, 1\}^\ell$. Define $\mu(y) = \mu_1\mu_2 \cdots \mu_n$, and $\gamma(y)$, $\gamma'(y)$, $\beta(y)$, $\nu(y)$, $\omega(y)$, $\eta(y)$ as before.

[6] More generally, for $k \in \boldsymbol{N}^+$, defining $\mathrm{QBF}_{k,\exists}$ as $\mathrm{QBF}_{3,\exists}$ but with k alternating quantifiers, and replacing "CNF" with "DNF" if k is even, it is shown in [12], [34] that $\mathrm{QBF}_{k,\exists}$ is Σ_k^p-complete; this was previously known for $k = 1$.

We next describe the set of words (8) that are generated from the synchronizing state s_j, for a fixed j with $1 \le j \le m$

1) $\beta(y) = 0^k$, $\gamma(y) = \gamma'(y) \in \{0, 1\}$, $\eta(y) \in \{0, 1\}^\ell$, and either
   a) $\mu_i(y) = \mu'_i(y)$ for $1 \le i \le n$ and $(\mu(y), \nu(y), \omega(y))$ satisfies the conjunct C_j, or
   b) $\mu_i(y)\mu'_i(y) \in \{00, 01, 11\}$ for $1 \le i \le n$, there exists at least one i such that $\mu_i(y)\mu'_i(y) = 01$, and $\nu(y), \omega(y) \in \{0, 1\}^n$;
2) $\beta(y) \in \{0, 1\}^k - \{0^k\}$, $\gamma(y) = \gamma'(y) \in \{0, 1\}$, $\eta(y) \in \{0, 1\}^\ell$, $\mu_i(y)\mu'_i(y) \in \{00, 11\}$ for all $1 \le i \le n$, $\nu(y) \in \{0, 1\}^n$, and $\omega(y) = 0^n$.

Moreover, after producing a y of this form, G is in the synchronizing state s_z where $z = \mathrm{num}(\eta(y))$ (we define $s_0 = s_1$ and $s_j = s_m$ for all $j > m$).

There is a DIF G of size polynomial in n and m having this property, and, moreover, G has memory at most q. To achieve this memory bound we design G to have exactly two states, say g_0 and g_1, in the level $L_g \stackrel{\text{def}}{=} k + 3n + 2$ just after ω_n is produced and just before the first symbol of η is produced. The only information that G must "remember" at this point is the value of the most recently produced γ (see (8)) so that it will produce the next γ' equal to this value. Let $\hat{y}$ be a codeword of length q that is generated from some state s of G, and let L be the level containing s. If $L_g < L \le L_g + \ell$ (that is, $\hat{y}$ is a rotation of a y as in (8) where $\eta\gamma'$ is "split") then $\gamma(y)$ determines the state g_0 or g_1 entered at level L_g. Otherwise, $\eta(y)$ determines the synchronizing state that is entered at level 0. In either case, $\delta_G(s, \hat{y})$ is determined independently of s.

Note that codeword types 1a), 1b), and 2) are pairwise disjoint. In the notation of Section III-E, $y = 2\gamma I(y)\gamma$ where the important part of y is $I(y) = \beta\mu\nu\omega\eta$.

We show that (G, p, q) is block feasible. The number of codewords of type 2) is $2(2^k - 1)2^n 2^n 2^\ell$, where the five terms count the number of choices for γ, β, μ, ν, and η, respectively. For codewords of type 1b), if $\beta = 0^k$ and $\mu_1\mu'_1 = 01$ then γ, ν, ω, and η can be arbitrary. Thus, the number of codewords of type 1b) is at least $2^{2n+\ell+1}$. This gives a total of at least $2^{k+2n+\ell+1} = 2^p$ codewords, and they form an $\mathcal{S}(G)$-concatenable set.

To complete the proof, we show that

$$\exists\, u\, \forall\, v\, \exists\, w F(u, v, w) = 1$$

iff (G, p, q) has a decoder of size zero. As in the proof of Theorem 3.1, a decoder of size zero is specified by its output mapping, a one-to-one mapping

$$\phi : \{1, 2, \ldots, p\} \to \{1, 2, \ldots, q - 1\}.$$

As before, let Φ denote the range of ϕ, so $|\Phi| = p$.

(*Only if*) Say that $\exists u \forall v \exists w F(u, v, w) = 1$. We describe a block codebook $\mathcal{C}$ for (G, p, q). Let the assignment $\tilde{u} \in \{0, 1\}^n$ be such that $\forall v \exists w F(\tilde{u}, v, w) = 1$. For each $\tilde{v} \in \{0, 1\}^n$, there exists at least one $\tilde{w} \in \{0, 1\}^n$ such that $F(\tilde{u}, \tilde{v}, \tilde{w}) = 1$; choose an arbitrary one of these $\tilde{w}$ and denote it $\tilde{w}(\tilde{v})$, so $F(\tilde{u}, \tilde{v}, \tilde{w}(\tilde{v})) = 1$ for each $\tilde{v} \in \{0, 1\}^n$.

$\mathcal{C}$ contains the y of type 1a) such that $\gamma(y) = \gamma'(y) \in \{0, 1\}$, $\beta(y) = 0^k$, $\eta(y) \in \{0, 1\}^\ell$, $\mu_i(y) = \mu'_i(y) = \tilde{u}_i$ for $1 \le i \le n$, $\nu(y) \in \{0, 1\}^n$, and $\omega(y) = \tilde{w}(\nu(y))$. By assumption, the assignment $(\mu(y), \nu(y), \omega(y))$ satisfies all conjuncts of F, so this is a valid codeword of type 1a) for each starting (synchronizing) state $s_1, \ldots, s_m$. The number of such y is $2^{\ell+1}2^n$.

$\mathcal{C}$ contains the y of type 1b) such that $\gamma(y) = \gamma'(y) \in \{0, 1\}$, $\beta(y) = 0^k$, $\eta(y) \in \{0, 1\}^\ell$, $\nu(y) \in \{0, 1\}^n$, $\omega(y) = \tilde{w}(\nu(y))$, and

$$\mu_i(y)\mu'_i(y) \in \{\tilde{u}_i\tilde{u}_i, 01\}, \qquad \text{for } 1 \le i \le n$$

except that the y with

$$\mu_1(y)\mu'_1(y) \cdots \mu_n(y)\mu'_n(y) = \tilde{u}_1\tilde{u}_1 \cdots \tilde{u}_n\tilde{u}_n$$

are not in $\mathcal{C}$ because these are not codewords of type 1b). The number of such y is $2^{\ell+1}(2^n - 1)2^n$.

$\mathcal{C}$ contains the y of type 2) such that $\gamma(y) = \gamma'(y) \in \{0, 1\}$, $\beta(y) \in \{0, 1\}^k - \{0^k\}$, $\eta(y) \in \{0, 1\}^\ell$, $\mu(y)$, $\nu(y) \in \{0, 1\}^n$, $\mu'(y) = \mu(y)$, and $\omega(y) = 0^n$. The number of such y is $2^{\ell+1}(2^k - 1)2^{2n}$.

Therefore,

$$|\mathcal{C}| = 2^{\ell+1}2^n((2^k - 1)2^n + (2^n - 1) + 1) = 2^{\ell+1}2^k 2^{2n} = 2^p.$$

Define the output mapping so that Φ contains all coordinates of γ, β, ν, and η, and in addition for $1 \le i \le n$, Φ contains μ_i if $\tilde{u}_i = 1$, or μ'_i if $\tilde{u}_i = 0$. The number of coordinates in Φ is $k + 2n + \ell + 1 = p$. We must show that as y ranges over $\mathcal{C}$, the p-bit word defined by these coordinates ranges over all of $\{0, 1\}^p$. For each of the three types of codewords in $\mathcal{C}$, $\gamma(y) \in \{0, 1\}$, $\nu(y) \in \{0, 1\}^n$, and $\eta(y) \in \{0, 1\}^\ell$ can be arbitrary, regardless of the value of $\beta(y)$, $\mu(y)$, $\mu'(y)$. So we can concentrate on the coordinates of β, μ and μ' in Φ. As y ranges over all codewords of type 1a) and 1b) in $\mathcal{C}$ (where necessarily $\beta(y) = 0^k$)

$$\mu_1(y)\mu'_1(y)\mu_2(y)\mu'_2(y) \cdots \mu_n(y)\mu'_n(y)$$

assumes all values in

$$\{\tilde{u}_1\tilde{u}_1, 01\} \times \{\tilde{u}_2\tilde{u}_2, 01\} \times \cdots \times \{\tilde{u}_n\tilde{u}_n, 01\}.$$

Because Φ contains the coordinate μ''_i defined as μ_i (resp., μ'_i) if $\tilde{u}_i = 1$ (resp., $\tilde{u}_i = 0$), $\mu''_1 \cdots \mu''_n$ assumes all values in $\{0, 1\}^n$. Now for codewords y of type 2) in $\mathcal{C}$, $\beta(y)$ can have any value in $\{0, 1\}^k - \{0^k\}$ and $\mu_1(y)\mu'_1(y) \cdots \mu_n(y)\mu'_n(y)$ can have any value in $\{00, 11\}^n$; the latter statement implies that $\mu''_1(y) \cdots \mu''_n(y)$ can have any value in $\{0, 1\}^n$.

(*If*) Say now that $\mathcal{C}$ is a block codebook for (G, p, q) and let Φ be the range of the output mapping of a decoder of size zero for $\mathcal{C}$. By Claim 1, all $y \in \mathcal{C}$ have the same rotation. If $y = y'2y'' \in \mathcal{C}$, then $yy = y'2y''y'2y'' \in \mathcal{S}(G)$, so $2y''y'$ is generated from at least one synchronizing state. By writing $2y''y'$ in the form (8) we identify the subwords $\gamma(y)$, $\beta(y)$, etc., and their coordinates.

We prove a claim that restricts which coordinates can be in Φ. We use that if $S \subseteq \Phi$, then as y ranges over $\mathcal{C}$, the $|S|$-bit word defined by these coordinates must assume every value in $\{0, 1\}^{|S|}$, and each of these values must appear $2^{p-|S|}$ times.

Claim 10:

1) Φ does not contain both coordinates γ and γ'.
2) For each $1 \le i \le n$, Φ does not contain coordinate ω_i.
3) For each $1 \le i \le n$, Φ does not contain both coordinates μ_i and μ'_i.

Proof:

1) Because $\gamma(y) = \gamma'(y)$ for all codewords, $\gamma(y)\gamma'(y)$ cannot have the value 01.
2) This follows because $\omega(y) = 0^n$ for all y of type 2), there are at most $2^{4n+\ell+1}$ y's of type 1a) and 1b), and
$$2^{4n+\ell+1} < 2^{2n+k+\ell} = 2^{p-1}$$
because we have chosen $k \geq 2n+2$.
3) This follows because
$$\mu_i(y)\mu_i'(y) \in \{00, 11, 01\}$$
for all codewords y. □

Because Φ contains $p = k+2n+\ell+1$ of the $q-1 = k+4n+\ell+2$ coordinates, and Claim 10 rules out $2n+1 = q-1-p$ of them, the only choice possible is that Φ contains all coordinates of β, ν, and η, contains exactly one of γ or γ', and for $1 \leq i \leq n$ contains exactly one of μ_i or μ_i'.

Because Φ contains one of γ or γ' (say, without loss of generality, that it contains γ), it follows that $\gamma(y) = 0$ for some $y \in \mathcal{C}$ and $\gamma(y) = 1$ for some (other) $y \in \mathcal{C}$. Notice that we have not used to this point any assumption that the important part $I(y)$ of y is not split. It follows from Claim 9 that $I(y) = \beta\mu\nu\omega\eta$ is not split. In other words, the common rotation ρ of words in $\mathcal{C}$ is either 0, 1, $q-2$, or $q-1$. In the rest of the proof we use only the important part of each $y \in \mathcal{C}$; so to simplify the notation, assume that $\rho = 0$.

Claim 11: Let $y \in \mathcal{C}$ be such that $\beta(y) = 0^k$ and $\mu_i(y) = \mu_i'(y)$ for $1 \leq i \leq n$. Then $F(\mu(y), \nu(y), \omega(y)) = 1$.

Proof: Suppose otherwise that $(\mu(y), \nu(y), \omega(y))$ does not satisfy the conjunct C_j. Let $y' \in \mathcal{C}$ be such that $\eta(y')$ is the ℓ-bit binary representation of j, denoted $\mathrm{bin}_\ell(j)$. Such a y' must exist because $\ell = \lceil \log(m+1) \rceil$ and all coordinates of η are in Φ. Consider the concatenation $y'y$, that is,

$$\overbrace{2\cdots \mathrm{bin}_\ell(j)\gamma'}^{y'}\overbrace{2\gamma 0^k \mu_1(y)\mu_1(y)\cdots\mu_n(y)\mu_n(y)\nu(y)\omega(y)\cdots}^{y}.$$

Because $\beta(y) = 0^k$ and $\mu_i(y) = \mu_i'(y)$ for all i, y must be a codeword of type 1a) generated from s_j. But it cannot be, because $(\mu(y), \nu(y), \omega(y))$ does not satisfy the conjunct C_j. □

We now show that $\exists u \forall v \exists w F(u, v, w) = 1$. For $1 \leq i \leq n$, if the coordinate μ_i (resp., μ_i') is in Φ, then define the coordinate μ_i'' to be μ_i (resp., μ_i'), and define $\tilde{u}_i = 1$ (resp., $\tilde{u}_i = 0$). We show that $\forall v \ \exists w F(\tilde{u}, v, w) = 1$. Let $\tilde{v} \in \{0, 1\}^n$ be arbitrary. Recall that Φ contains the coordinates β, μ'', and ν. Therefore, there must exist a $y \in \mathcal{C}$ with $\beta(y) = 0^k$, $\mu_i'' = \tilde{u}_i$ for $1 \leq i \leq n$, and $\nu(y) = \tilde{v}$. We argue that $\mu_i(y) = \mu_i'(y)$ for all i. Let i with $1 \leq i \leq n$ be arbitrary. There are two cases; they both use the fact that for all codewords y and all $1 \leq i \leq n$ we have $\mu_i(y)\mu_i'(y) \neq 10$. i) if $\tilde{u}_i = 1$, then $\mu_i(y) = \mu_i''(y) = \tilde{u}_i = 1$, so $\mu_i'(y) \neq 0$; ii) if $\tilde{u}_i = 0$, then $\mu_i'(y) = \mu_i''(y) = \tilde{u}_i = 0$, so $\mu_i(y) \neq 1$.

Let $\tilde{w} = \omega(y)$. By Claim 11, $F(\tilde{u}, \tilde{v}, \tilde{w}) = 1$. Because $\tilde{v}$ was arbitrary, we have shown that $\forall v \ \exists w F(\tilde{u}, v, w) = 1$. □

IV. Complexity of Finding the Maximum Block Rate

In this section, we classify the computational complexity of computing $MaxConcat(G, q)$ and the related problem of computing the maximum block rate for (G, q). Before stating the results of this section, we need some additional definitions.

Valiant [7] has defined a fundamental complexity class of functions called #P. While problems in NP ask whether there exists an object having a certain property, such as the existence of a satisfying assignment for a CNF formula, functions in #P *count* the number of objects having the property, such as counting the number of satisfying assignments of a CNF formula.

Formally, a function $f\colon \{0, 1\}^+ \to \boldsymbol{N}$ is in #P iff there is a ptime-relation $R(u, v)$ and a polynomial Q such that, for all $u \in \{0, 1\}^+$
$$f(u) = |\{v \,|\, |v| = Q(|u|) \text{ and } R(u, v) = 1\}|.$$
For example, recalling the example given in Section II-D1 that the decision problem for Hamiltonicity belongs to NP, using a similar relation R shows that, if f is the function that maps a graph H to the number of Hamiltonian cycles in H, then f belongs to #P. If $f \in \#\mathrm{P}$ then $f(u)$ is typically exponential in $|u|$ (as the examples of counting satisfying assignments and counting Hamiltonian cycles demonstrate). Therefore, for computational purposes the value of f is represented in binary notation. This avoids trivial hardness results for functions in #P that are based on the argument that it takes exponential time just to write the output. In this way, we also view f as a function from $\{0, 1\}^+$ to $\{0, 1\}^+$. Toda [46] has proved that $\Sigma_k^p \subseteq \mathrm{P}^{\#\mathrm{P}}$ for all k (thus, $\mathrm{P}^{\#\mathrm{P}}$ is a relatively "high" complexity class compared to NP), but it is not known whether or not $\mathrm{P} = \mathrm{P}^{\#\mathrm{P}}$. Further information about #P can be found in [27, Sec. 4].

One of the main goals of this section is to classify the complexity of computing $MaxConcat(G, q)$, where G and q are inputs to the function. We do not know whether this function belongs to #P, although we do show that it belongs to a "higher" class called Max#P, defined as follows. The definition is similar to that of #P, but the relation R has an additional argument w, and $f(u)$ is the maximum over w of length $Q(|u|)$, of the number of v such that $R(u, v, w) = 1$. Formally, the function $f\colon \{0, 1\}^+ \to \boldsymbol{N}$ is in Max#P iff there is a ptime-relation $R(u, v, w)$ and a polynomial Q such that, for all $u \in \{0, 1\}^+$
$$f(u) = \max_{|w| = Q(|u|)} |\{v \,|\, |v| = Q(|u|) \text{ and } R(u, v, w) = 1\}|.$$
As far as we know, the class Max#P of functions has not been formally defined before; although the closely related class $\mathrm{NP}^{\#\mathrm{P}}$ of decision problems, which can also be defined by a "max, counting" quantification, has been defined and used, for example, in [22], [25], [47].

By definitions, Toda's result [46], and the inclusion $\mathrm{NP}^{\#\mathrm{P}} \subseteq \mathrm{P}^{\mathrm{Max}\#\mathrm{P}}$ which follows from a result of Torán [47, Theorem 4.1], the complexity classes we have introduced have the inclusion structure
$$\mathrm{P} \subseteq \mathrm{NP} \subseteq \Sigma_2^p \subseteq \Sigma_3^p \subseteq \cdots \subseteq \mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{NP}^{\#\mathrm{P}} \subseteq \mathrm{P}^{\mathrm{Max}\#\mathrm{P}}.$$
If $\mathrm{P} = \mathrm{NP}$ then the polynomial hierarchy collapses ($\mathrm{P} = \Sigma_k^p$ for all $k \geq 1$), but it does not contradict current knowledge that $\mathrm{P} = \mathrm{NP}$ and $\mathrm{P} \neq \mathrm{P}^{\#\mathrm{P}} \neq \mathrm{NP}^{\#\mathrm{P}}$.[7]

[7]The classes $\mathrm{P}^{\#\mathrm{P}}$ and $\mathrm{NP}^{\#\mathrm{P}}$ are equal to P^{PP} and $\mathrm{NP}^{\mathrm{PP}}$, respectively, where the class PP is defined by Gill [48]; the latter names are often used in the literature, e.g., [25], [46].

In addition to the "upper bound" that $MaxConcat$ belongs to Max#P, we also prove a corresponding "lower bound" by showing that every function in Max#P is reducible to $MaxConcat$. The type of reduction that is typically used in #P-hardness and #P-completeness results is *polynomial-time Turing reduction*, denoted $\leq_T^p$. If f and g are functions, then $f \leq_T^p g$ if f can be computed in polynomial time using an oracle for g; that is, in the notation of Section II-D3, $f \in \mathrm{P}^g$. We use a stronger reduction, introduced by Krentel [49], called metric reduction. Let f, g: $\{0,1\}^+ \to \{0,1\}^+$. Then f is *metric reducible* to g, which we write $f \leq_{mtrc} g$, if there is a pair P_1, P_2 of polynomial-time computable functions such that, for all $u \in \{0,1\}^+$, $f(u) = P_2(g(P_1(u)))$. Using metric reduction in place of $\leq_T^p$ in a hardness proof gives a stronger result, because $f \leq_{mtrc} g$ implies $f \leq_T^p g$, but the converse implication is not known to hold. Metric reduction between functions is close in spirit to polynomial-time transformation between decision problems. Both $\leq_{mtrc}$ and $\leq_T^p$ are transitive relations.

In classifying the problem of computing the maximum block rate, we use the class $\mathrm{FP}^{\mathrm{Max\#P}}$, that is, the class of functions that can be computed in polynomial time using an arbitrary function in Max#P as an oracle.

Because it is more typical in complexity theory to prove completeness (under $\leq_m^p$) of decision problems rather than functions, we also consider the following "decision versions" of $MaxConcat$ and $MaxBlkRate$.

MAXCONCAT
Instance: (G, q, K) where G is a DIF and q, K are positive integers.
Question: Is $MaxConcat(Q, q) \geq K$?

In this problem, K may be written in binary notation, because $MaxConcat(Q, q)$ may be exponentially larger than $|G|$ and q.

MAXBLKRATE
Instance: (G, q, K) where G is a DIF and q, K are positive integers.
Question: Is $MaxBlkRate(Q, q) \geq K/q$?

We next state the main results of this section, and then give their proofs. Recall that the finite memory restriction means that the input is restricted to (G, q) where G has finite memory.

Theorem 4.1:

1) MAXCONCAT and MAXBLKRATE are $\mathrm{NP}^{\#\mathrm{P}}$-complete (under $\leq_m^p$).
2) $MaxConcat$ is Max#P-complete under $\leq_{mtrc}$.
3) $MaxConcat$ and $MaxBlkRate$ are $\mathrm{FP}^{\mathrm{Max\#P}}$-complete under $\leq_T^p$.

Moreover, these statements hold with the finite memory restriction.

Recall that the maximum block rate of (G, q) is

$$MaxBlkRate(G, q) = \lfloor \log MaxConcat(G, q) \rfloor / q$$

(Definition 2.2). On the surface, this function might be easier to compute than $MaxConcat$, because $MaxBlkRate(G, q)$ does not have to compute $MaxConcat(G, q)$ exactly. However, Theorem 4.1 part 3) implies that $MaxConcat$ and $MaxBlkRate$ are reducible to one another by $\leq_T^p$.

The final issue we consider in the section is related to the question of whether a "compact representation" of a block code of maximum rate can be found in polynomial time. We have shown in Lemma 2.2 that for each $G = (V, E)$ and q, there exists a $\tau \subseteq V$ such that there is a block codebook $\mathcal{C}$ having maximum rate (that is, $|\mathcal{C}| = 2^{q \cdot MaxBlkRate(G,q)}$) and $\mathcal{C} \subseteq \mathcal{T}_q(\tau)$ (recall that $\mathcal{T}_q(\tau)$ is the set of q-bit words y such that, for all $t \in \tau$, y is generated from t and $\delta(t, y) \in \tau$). Is there a polynomial-time computable function that takes (G, q) where G has finite memory, and outputs a set τ having this property? Call such a function a *τ-finder*. We do not know the answer, but we show that a "yes" answer implies a previously unknown equality between classes.

Theorem 4.2: If there is a polynomial-time computable τ-finder, then $\mathrm{P}^{\#\mathrm{P}} = \mathrm{NP}^{\#\mathrm{P}}$ and $\mathrm{FP}^{\#\mathrm{P}} = \mathrm{FP}^{\mathrm{Max\#P}}$.

We now give the proofs of these theorems. The next lemma gives the "upper bound" parts of Theorem 4.1.

Lemma 4.3: $MaxConcat$ belongs to Max#P; $MaxConcat$ and $MaxBlkRate$ belong to $\mathrm{FP}^{\mathrm{Max\#P}}$; and MAXCONCAT and MAXBLKRATE belong to $\mathrm{NP}^{\#\mathrm{P}}$.

Proof: By Lemma 2.2, if $G = (V, E)$, $\mathcal{C} \subseteq \{0,1\}^q$, $\mathcal{C}$ is $\mathcal{S}(G)$-concatenable, and $|\mathcal{C}| = MaxConcat(G, q)$, then there exists a $\tau \subseteq V$ such that $\mathcal{C} = \mathcal{T}_q(\tau)$. Therefore,

$$MaxConcat(G, q) = \max_{\tau \subseteq V} |\{y \,|\, y \in \mathcal{T}_q(\tau)\}|.$$

Because the relation "$y \in \mathcal{T}_q(\tau)$" can be decided in polynomial time, $MaxConcat$ belongs to Max#P by definition. Obviously, $\mathrm{Max\#P} \subseteq \mathrm{FP}^{\mathrm{Max\#P}}$.

Similarly, (G, q, K) is a "yes" instance of MAXCONCAT iff

$$(\exists \tau \subseteq V)[|\{y \,|\, y \in \mathcal{T}_q(\tau)\}| \geq K]$$

so MAXCONCAT belongs to $\mathrm{NP}^{\#\mathrm{P}}$ by definition.

The upper bounds on $MaxBlkRate$ and MAXBLKRATE follow from the above and

$$MaxBlkRate(G, q) = \lfloor \log MaxConcat(G, q) \rfloor / q. \qquad \square$$

The "hardness" parts of Theorem 4.1 will follow easily from Lemma 4.4 given next. This lemma is also used later in the proof of Theorem 5.1. The proof of Theorem 3.4 has many similarities with the proof of Lemma 4.4, and familiarity with the former proof would be helpful in reading the latter.

For Boolean variables $v = \{v_1, \ldots, v_n\}$ and $w = \{w_1, \ldots, w_n\}$, let $F = F(v, w)$ denote a formula of these variables. Define

$$Max\#(F) = \max_{w \in \{0,1\}^n} |\{v \in \{0,1\}^n \,|\, F(v, w) = 1\}|.$$

Lemma 4.4: There is a constant c such that, for each $n, m \in \boldsymbol{N}^+$ there are $B_1(n, m)$, $B_2(n, m) \in \boldsymbol{N}^+$ such that $B_1(n, m)$, $B_2(n, m) \leq 2^{c(n+m)}$ and such that the following holds. There is a polynomial-time transformation with the following input–output behavior.

Input: a CNF formula $F(v, w)$ having m conjuncts and variables $v = \{v_1, \ldots, v_n\}$ and $w = \{w_1, \ldots, w_n\}$, and an integer Z with

$$0 \le Z \le (B_1(n, m) \cdot 2^n + B_2(n, m))/2.$$

Output: (G, q) such that G is a DIF having finite memory, $q = O(n)$, and

$$MaxConcat(G, q) = B_1(n, m) \cdot Max\#(F) + B_2(n, m) + 2Z.$$

(The numbers Z, $B_1(n, m)$ and $B_2(n, m)$ are written in binary notation, so they can be exponential in $n + m$.)

Proof: In the proof of Theorem 3.4, we used $\eta(y)$ with $\mathrm{num}(\eta(y)) = j$ to place the DIF in its synchronizing state s_j so that the satisfaction of conjunct C_j is tested in the codeword immediately following y. Here we use $\eta(y)$ in the same way, and we also use $\zeta(y)$ to enforce that in any $\mathcal{S}(G)$-concatenable codebook $\mathcal{C}$ of q-bit words, all $y \in \mathcal{C}$ with $\beta(y) = 0^k$ must have the same $\omega(y)$.

For each n and m, let $\ell = \lceil \max(\log(n + 1), \log(m + 1)) \rceil$ and $k = \ell + 2$. Define

$$B_1(n, m) = 2^{2\ell+1} \text{ and } B_2(n, m) = (2^{k-1} - 1)2^{2n+2\ell+1}.$$

Fix an input $F(v, w)$ and Z. Because n and m are fixed for the rest of the proof, abbreviate $B_1 = B_1(n, m)$ and $B_2 = B_2(n, m)$. Let $C_1, \ldots, C_m$ be the conjuncts of F. Let $q = k + 2n + 2\ell + 3$.

We describe a special DIF G. A q-bit word y generated from a synchronizing state of G has the form

$$y = 2\gamma\beta\nu\omega\eta\zeta\gamma' \tag{9}$$

where $\gamma, \gamma' \in \{0, 1\}$, $\beta \in \{0, 1\}^k$, $\nu, \omega \in \{0, 1\}^n$, and $\eta, \zeta \in \{0, 1\}^\ell$.

To enforce that all $y \in \mathcal{C}$ with $\beta(y) = 0^k$ must have the same $\omega(y)$, G has $3mn^2$ synchronizing states: $s(i_1, i_2, b, j)$ where $1 \le i_1, i_2 \le n$, $1 \le j \le m$, and $b \in \{0, 1, *\}$. Here, $*$ is a "don't care" symbol such that the tests "$0 = *$?" and "$1 = *$?" are defined as true. Suppose that y is generated from state $s(i_1, i_2, b, j)$. If $\beta(y) = 0^k$, then G will test that $\omega_{i_2}(y) = b$ (and not produce any more symbols if the test fails), it will remember $b' = \omega_{i_1}(y)$, it will test C_j as before, and it will end in state $(\mathrm{num}(\zeta(y)), i_1, b', \mathrm{num}(\eta(y)))$ (as before, ζ and η that are outside the range $[1, n]$ and $[1, m]$, respectively, are replaced by some (arbitrary) number in the range). As an example, let $y, y', y'' \in 2 \cdot \{0, 1\}^{q-1}$ with $\beta(y') = \beta(y'') = 0^k$. Let $j = \mathrm{num}(\eta(y))$ and $i = \mathrm{num}(\zeta(y))$. As G produces the concatenation $yy'y''$, conjunct C_j must be satisfied by $\nu(y')\omega(y')$, and $\omega_i(y') = \omega_i(y'')$ must hold.

We next describe the set of words (9) that are generated from the synchronizing state $s(i_1, i_2, b, j)$:

1) $\beta(y) = 0^k$, $\gamma(y) = \gamma'(y) \in \{0, 1\}$, $\eta(y), \zeta(y) \in \{0, 1\}^\ell$, $(\nu(y), \omega(y))$ satisfies conjunct C_j, and $\omega_{i_2}(y) = b$;
2) $\beta(y) \in 0 \cdot \{0, 1\}^* \cdot 1 \cdot \{0, 1\}^*$, $\gamma(y) = \gamma'(y) \in \{0, 1\}$, $\eta(y), \zeta(y) \in \{0, 1\}^\ell$, and $\nu(y), \omega(y) \in \{0, 1\}^n$;
3) $\beta(y) = 1\beta'(y)$ where $\beta'(y) \in \{0, 1\}^{k-1}$, $\gamma(y) = \gamma'(y) \in \{0, 1\}$, $\eta(y), \zeta(y) \in \{0, 1\}^\ell$, $\nu(y), \omega(y) \in \{0, 1\}^n$, and $\mathrm{num}(\beta'(y)\nu(y)\omega(y)\eta(y)\zeta(y)) < Z$.

Moreover, if $\beta(y) = 0^k$, then G ends in state

$$s(\mathrm{num}(\zeta(y)), i_1, \omega_{i_1}(y), \mathrm{num}(\eta(y)))$$

otherwise, G ends in state

$$s(\mathrm{num}(\zeta(y)), i_1, *, \mathrm{num}(\eta(y))).$$

Notice that the number of codewords of type 2) is B_2 and the number of type 3) is $2Z$. To verify that the number of type 3) is $2Z$ (that is, two choices for γ and Z choices for $\beta'\nu\omega\eta\zeta$), it must be verified that the upper bound on Z (namely, $(B_1 2^n + B_2)/2$) can be represented by a $(k - 1 + 2n + 2\ell)$-bit number $\beta'\nu\omega\eta\zeta$, which is true because

$$(B_1 2^n + B_2)/2 = 2^{2\ell}2^n + (2^{k-1} - 1)2^{2n+2\ell} < 2^{k-1+2n+2\ell}.$$

There is a DIF G of size polynomial in n and m having this property, and, moreover, G has memory at most $2q + \ell + 1$. The memory bound holds because, given any $z \in \mathcal{S}(G)$ with $|z| \ge 2q + \ell + 1$, we can write

$$z = \cdots \zeta_1\gamma_1' 2\gamma_2\beta_2\nu_2\omega_2\eta_2\zeta_2\gamma_2' \cdots .$$

Now ζ_1, β_2, ω_2, η_2, and ζ_2 determine the (synchronizing) state that G is in after producing $\zeta_2\gamma_2'$.

We show that

$$MaxConcat(G, q) = B_1 \cdot Max\#(F) + B_2 + 2Z.$$

We first show

$$\begin{aligned} &MaxConcat(G, q) \\ &\quad \ge B_1 \cdot Max\#(F) + B_2 + 2Z \\ &\quad \stackrel{\text{def}}{=} 2^{2\ell+1} \cdot Max\#(F) + (2^{k-1} - 1)2^{2n+2\ell+1} + 2Z. \end{aligned} \tag{10}$$

Let $\tilde{w} \in \{0, 1\}^n$ be such that $Max\#(F) = |\{v \,|\, F(v, \tilde{w}) = 1\}|$. Let $\mathcal{C}$ be the set of codewords $y \in 2 \cdot \{0, 1\}^{q-1}$ such that $\mathcal{C}$ contains all codewords of types 2) and 3), and contains the codewords of type 1) such that

$$\begin{aligned} \beta(y) &= 0^k \qquad \omega(y) = \tilde{w} \\ F(\nu(y), \tilde{w}) &= 1 \qquad \eta(y), \zeta(y) \in \{0, 1\}^\ell \end{aligned}$$

and

$$\gamma(y) = \gamma'(y) \in \{0, 1\}.$$

That $\mathcal{C}$ is $\mathcal{S}(G)$-concatenable follows from two observations. i) For codewords y of the first type, $\omega(y)$ is constant $(\tilde{w})$ and $F(\nu(y), \omega(y)) = 1$ (that is, $(\nu(y), \omega(y))$ satisfies all conjuncts of F). ii) The purpose of using the "don't care" symbol $*$ is to allow $yy' \in \mathcal{S}(G)$ when y is not of type 1) and y' is of type 1). In this case, G is in a state of the form $(i_1, i_2, *, j)$ after producing y, so G will not test that $\omega_{i_2}(y) = \omega_{i_2}(y')$ (this test would fail for this $\mathcal{C}$ because $\omega(y) \ne \tilde{w}$ is possible if y is not of type 1)). It is straightforward to see that (10) holds, because the three summands of (10) are the number of $y \in \mathcal{C}$ of types 1), 2), and 3), respectively. We now show that

$$MaxConcat(G, q) \le B_1 \cdot Max\#(F) + B_2 + 2Z. \tag{11}$$

Let $\mathcal{C} \subseteq \{0, 1\}^q$ be $\mathcal{S}(G)$-concatenable and $|\mathcal{C}| = MaxConcat(G, q)$. Again by Claim 1, all words in $\mathcal{C}$ have the same rotation. The set of all codewords of types 2) and 3) is an $\mathcal{S}(G)$-concatenable set of size $B_2 + 2Z$. Therefore,

$$|\mathcal{C}| \ge B_2 + 2Z = (2^{k-1} - 1)2^{2n+2\ell+1} + 2Z. \tag{12}$$

If $y'2y'' \in \mathcal{C}$ then $y'2y''y'2y'' \in \mathcal{S}(G)$, so we can identify the coordinates of $\beta(y)$, $\nu(y)$, etc. We show that each one of the

2^ℓ (resp., 2^ℓ, 2) possibilities for $\eta(y)$ (resp., $\zeta(y)$, $\gamma(y)$) must appear for at least one $y \in \mathcal{C}$. We prove this for η. The same proof works for ζ and γ, because the only fact about $|\eta|$ that we use is $k \geq |\eta| + 2$, and the same is true for $|\zeta| = \ell$ and $|\gamma| = 1$, because $k = \ell + 2$. We show that for each $\hat{\eta} \in \{0, 1\}^\ell$, there exists a $y \in \mathcal{C}$ such that $\eta(y) = \hat{\eta}$. Suppose otherwise. Then

$$|\mathcal{C}| \leq 2^{\ell+1}(2^\ell - 1)(2^{2n} + (2^{k-1} - 1)2^{2n}) + 2Z$$

(the two terms in parentheses count the maximum number of choices for β, ν, ω in codewords of type 1) and 2), respectively). A straightforward calculation, using $k \geq \ell + 2$, shows that this upper bound on $|\mathcal{C}|$ is strictly less than the expression on the right-hand side in (12). This is a contradiction.

In particular, the fact that both values for γ must appear implies by Claim 9 that the rotation of $\mathcal{C}$ is not bad, so the important part $I(y) = \beta\nu\omega\eta\zeta$ is not split for $y \in \mathcal{C}$.

Claim 12: There exists a $\tilde{w} \in \{0, 1\}^n$ such that the following holds. If $y \in \mathcal{C}$ and $\beta(y) = 0^k$, then: 1) $(\nu(y), \omega(y))$ satisfies F, and 2) $\omega(y) = \tilde{w}$.

Proof: The proof of 1) is identical to the proof of Claim 11, because we have shown that for each j with $1 \leq j \leq m$, there is a $y' \in \mathcal{C}$ with $\mathrm{num}(\eta(y')) = j$. The proof of 2) is similar. Suppose that there are $y, y' \in \mathcal{C}$ such that $\beta(y) = \beta(y') = 0^k$ and $\omega(y) \neq \omega(y')$; say that $\omega(y)$ and $\omega(y')$ differ in the ith bit. We have argued that there is a $y'' \in \mathcal{C}$ with $\mathrm{num}(\zeta(y'')) = i$. By definition of G, the concatenation $y''yy'$ does not belong to $\mathcal{S}(G)$. □

It follows from Claim 12 that the number of $y \in \mathcal{C}$ of type 1) is at most

$$2^{2\ell+1} \cdot Max\#(F) = B_1 \cdot Max\#(F).$$

Because the number of codewords of type 2) and 3) is B_2 and $2Z$, respectively, the bound (11) is immediate. □

We will also need the following Max#P-complete function and NP$^{\#\mathrm{P}}$-complete decision problem.

$Max\#SAT$
Input: A CNF formula $F(v, w)$.
Output: $Max\#(F)$ $(= \max_w |\{v \mid F(v, w) = 1\}|)$.

MAX#SAT
Instance: A CNF formula $F(v, w)$ and a positive integer K.
Question: Is $Max\#(F) \geq K$?

The proof of the following lemma is very similar to the proof of Theorem 4.1 of Simon [50], which itself follows the proof of Cook's theorem [40], [26, Sec. 2.6]. Part 2) is stated (without proof) by Wagner [22, Theorem 7] with Wagner's class $\bigvee$CP in place of NP$^{\#\mathrm{P}}$; Torán [47, Theorem 4.1] shows that $\bigvee \mathrm{CP} = \mathrm{NP}^{\#\mathrm{P}}$.

Lemma 4.5:
1) $Max\#SAT$ is Max#P-complete under $\leq_{mtrc}$ and FP$^{\mathrm{Max}\#\mathrm{P}}$-complete under $\leq^p_T$.
2) MAX#SAT is NP$^{\#\mathrm{P}}$-complete (under $\leq^p_m$).

Proof of Theorem 4.1: The upper bounds are given by Lemma 4.3, so it suffices to show the reductions. We first consider $MaxConcat$. To show (part 1) that

$$\mathrm{NP}^{\#\mathrm{P}} \leq^p_m \mathrm{MAXCONCAT\ (FM)}$$

it suffices to show that

$$\mathrm{MAX\#SAT} \leq^p_m \mathrm{MAXCONCAT\ (FM)}$$

by Lemma 4.5 part 2). Let $(F(v, w), K)$ be an instance of MAX#SAT. Using Lemma 4.4 with $Z = 0$, transform F to (G, q) such that G has finite memory and

$$MaxConcat(G, q) = B_1 \cdot Max\#(F) + B_2.$$

The output of the reduction is (G, q, K') where $K' = B_1K + B_2$. To show (part 2) that

$$\mathrm{Max\#P} \leq_{mtrc} MaxConcat\ \text{(FM)}$$

it suffices to show that

$$Max\#SAT \leq_{mtrc} MaxConcat\ \text{(FM)}$$

by Lemma 4.5(1) and the transitivity of metric reduction. Let $F(v, w)$ be an input to $Max\#SAT$. Using Lemma 4.4 with $Z = 0$, transform F to (G, q) such that G has finite memory and

$$MaxConcat(G, q) = B_1 \cdot Max\#(F) + B_2.$$

Then transform $i = MaxConcat(G, q)$ to $j = Max\#(F)$ by $j = (i - B_2)/B_1$. (These two transformations are the P_1 and P_2 in the definition of metric reductions.) In part 3),

$$\mathrm{FP}^{\mathrm{Max}\#\mathrm{P}} \leq^p_T MaxConcat\ \text{(FM)}$$

is immediate from

$$\mathrm{FP}^{\mathrm{Max}\#\mathrm{P}} \leq^p_T \mathrm{Max\#P}$$

(by definition)

$$\mathrm{Max\#P} \leq^p_T MaxConcat\ \text{(FM)}$$

(because $\leq_{mtrc}$ implies $\leq^p_T$), and the transitivity of $\leq^p_T$.

We now consider $MaxBlkRate$. To show (part 1) that

$$\mathrm{NP}^{\#\mathrm{P}} \leq^p_m \mathrm{MAXBLKRATE\ (FM)}$$

it suffices to show that

$$\mathrm{MAX\#SAT} \leq^p_m \mathrm{MAXBLKRATE\ (FM)}.$$

Define the *maximum dataword length* by

$$\begin{aligned} MaxDWL(G, q) &\stackrel{\mathrm{def}}{=} q \cdot MaxBlkRate(G, q) \\ &= \lfloor \log MaxConcat(G, q) \rfloor \end{aligned}$$

so $MaxDWL$ takes only integral values. Let $(F(v, w), K)$ be an input to MAX#SAT, let n be the number of v-variables and w-variables, and m be the number of conjuncts. Let $B_1 = B_1(n, m)$ and $B_2 = B_2(n, m)$ be as in Lemma 4.4. By Lemma 4.4, there is a polynomial-time transformation f that transforms (F, Z) to $(G, q) = f(F, Z)$, where G has finite memory and

$$MaxConcat(G, q) = B_1 \cdot Max\#(F) + B_2 + 2Z \qquad (13)$$

provided that $0 \leq Z \leq (B_1 2^n + B_2)/2$. Because B_1 and B_2 are even, we can define the integers $B'_1 = B_1/2$ and $B'_2 = B_2/2$. Applying the definition of $MaxDWL$ to (13) gives

$$MaxDWL(f(F, Z)) = \lfloor \log(B'_1 \cdot Max\#(F) + B'_2 + Z) \rfloor + 1 \qquad (14)$$

provided that $0 \leq Z \leq B'_1 2^n + B'_2$. Recall that we want to decide whether $Max\#(F) \geq K$. If $K > 2^n$ then the answer is

clearly "no," so we can assume that $K \leq 2^n$. As Z ranges from 1 to $B_1'2^n+B_2'$, there must exist a Z_0 such that $B_1'K+B_2'+Z_0$ is an integral power of two, say 2^d. Taking $(G, q) = f(F, Z_0)$, it follows from (14) that $Max\#(F) \geq K$ iff $MaxDWL(G, q) \geq d+1$. Therefore, the output of the reduction (the instance of MAXBLKRATE (FM)) is $(G, q, d+1)$.

To show (part 3) that

$$\mathrm{FP}^{\mathrm{Max\#P}} \leq_T^p MaxBlkRate \text{ (FM)}$$

it suffices to prove

$$Max\#SAT \leq_T^p MaxBlkRate \text{ (FM)}$$

because

$$\mathrm{FP}^{\mathrm{Max\#P}} \leq_T^p Max\#SAT$$

by Lemma 4.5 part 1). To show this, we describe a polynomial-time algorithm $\mathcal{A}$, using an oracle for $MaxBlkRate$ (FM), that computes $Max\#(F)$. The oracle for $MaxBlkRate$ (FM) gives also an oracle for $MaxDWL$ (FM). Let $F(v, w)$ be an input to $\mathcal{A}$, and let n, m, f, B_1, B_2, B_1', and B_2' be as in the preceding paragraph. The algorithm $\mathcal{A}$, using the oracle for $MaxDWL$ (FM), first finds $p \stackrel{\text{def}}{=} MaxDWL(f(F, 0))$. Because $Max\#(F) \leq 2^n$, as Z ranges from 1 to $B_1'2^n + B_2'$, there must exist a Z_0, in particular a smallest Z_0, such that $B_1' \cdot Max\#(F)+ B_2' + Z_0$ is an integral power of two. This Z_0 is the smallest Z such that $MaxDWL(f(F, Z)) = p+1$. Thus, using the oracle for $MaxDWL$ (FM), the algorithm can find Z_0 in time $O(n+m)$ by binary search. Then from (14)

$$2^{MaxDWL(f(F, Z_0))} = 2(B_1' \cdot Max\#(F) + B_2' + Z_0)$$

from which $Max\#(F)$ can be computed in time polynomial in n and m. This completes the proof of Theorem 4.1.

Proof of Theorem 4.2: Assume that f is a polynomial-time τ-finder. To show that $\mathrm{FP}^{\#\mathrm{P}} = \mathrm{FP}^{\mathrm{Max\#P}}$, it suffices to show that $MaxBlkRate$ (FM) belongs to $\mathrm{FP}^{\#\mathrm{P}}$ because

$$\mathrm{FP}^{\mathrm{Max\#P}} \leq_T^p MaxBlkRate \text{ (FM)}$$

that is, it suffices to show that $MaxBlkRate$ (FM) can be computed in polynomial time using an oracle for some function in $\#\mathrm{P}$. Let (G, q) be an input to $MaxBlkRate$ where G has finite memory. Let $\tau = f(G, q)$. Because f is a τ-finder

$$MaxBlkRate(G, q) = \lfloor \log |\mathcal{T}_q(\tau)| \rfloor / q.$$

That the function $g(\tau, q) \stackrel{\text{def}}{=} |\mathcal{T}_q(\tau)|$ belongs to $\#\mathrm{P}$ is shown by the ptime-relation R where $R((\tau, q), y) = 1$ iff $y \in \mathcal{T}_q(\tau)$.

By considering only 0–1 valued functions, $\mathrm{FP}^{\#\mathrm{P}} = \mathrm{FP}^{\mathrm{Max\#P}}$ implies $\mathrm{P}^{\#\mathrm{P}} = \mathrm{P}^{\mathrm{Max\#P}}$. It is noted above that

$$\mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{NP}^{\#\mathrm{P}} \subseteq \mathrm{P}^{\mathrm{Max\#P}}$$

so $\mathrm{P}^{\#\mathrm{P}} = \mathrm{NP}^{\#\mathrm{P}}$, which completes the proof.

Remark: The proof of Theorem 4.1 shows that it holds when the memory of G is finite (and at most $2q + \log q$), but it does not prove the stronger restriction that the memory of G is at most q (as was done for the results of Section III). However, $MaxConcat$ and $MaxBlkRate$ are $\mathrm{FP}^{\#\mathrm{P}}$-hard and their decision versions are $\mathrm{P}^{\#\mathrm{P}}$-hard with this stronger restriction. This can be shown by a simple modification to Lemma 4.4. Now the input formula F has variables v, $Max\#(F)$ is replaced by $|\{v \mid F(v) = 1\}|$, the substring ω is eliminated from the proof and, as in the proof of Theorem 3.4, the synchronizing states are s_i for $1 \leq i \leq m$ and the memory of G is shown to be at most q.

V. Polynomial-Size Encoders and Maximum Block Rate

In this section we consider the question of whether the maximum block rate for (G, q) can always be achieved using encoder and decoder circuits of size polynomial in $|G|$ and q. This is formalized as follows.

Definition 5.1: The statement "max-rate block codes have polynomial-size encoders (resp., encoders/decoders)" means that there exists a polynomial Q such that, for each DIF G and $q \in \boldsymbol{N}^+$, there exists an encoder/decoder circuit pair $(\hat{\mathcal{E}}, \hat{\mathcal{D}})$ for (G, p, q) where $p = q \cdot MaxBlkRate(G, q)$ and $|\hat{\mathcal{E}}| \leq Q(|G|, q)$ (resp., $|\hat{\mathcal{E}}| + |\hat{\mathcal{D}}| \leq Q(|G|, q)$). This statement is abbreviated "poly-size encoders (resp., encoders/decoders)."

We show that the answer to this question is "yes" iff $\mathrm{Max\#P}$ lies lower in the complexity-theoretic structure than is currently believed. To state the result, we must define two more complexity classes.

First, define $\Delta_3^p = \mathrm{P}^{\Sigma_2^p}$. It follows from definitions that $\Sigma_2^p \subseteq \Delta_3^p \subseteq \Sigma_3^p$.[8] The second additional complexity class we need is called $\mathrm{P/poly}$. It is similar to P, but uses families of circuits rather than finite algorithms as the model of computation. Let $A \subseteq \{0, 1\}^+$ be a decision problem. Then A belongs to $\mathrm{P/poly}$ if there is a polynomial Q and a sequence $\{U_n \mid n \geq 1\}$ where U_n is an n-input 1-output circuit of size at most $Q(n)$, and U_n computes the function $\mathcal{F}_n$ where $\mathcal{F}_n(u) = 1$ if $u \in A$ or $\mathcal{F}_n(u) = 0$ if $u \notin A$. It is known that $\mathrm{P} \subseteq \mathrm{P/poly}$. The class $\mathrm{P/poly}$ is more general than P because for $\mathrm{P/poly}$ we can use a different polynomial-size deciding circuit U_n for each input length n, but for P we must use a single polynomial-time algorithm (having a *finite* description) that works for all n.

Because $\Sigma_k^p \subseteq \mathrm{P}^{\#\mathrm{P}}$ for all $k \geq 1$ [46] and $\mathrm{P} \subseteq \mathrm{P/poly}$, proving either that i) $\mathrm{P}^{\#\mathrm{P}} = \Delta_3^p$ or that ii) $\mathrm{P}^{\#\mathrm{P}} \not\subseteq \mathrm{P/poly}$ would be a major breakthrough in complexity theory. In particular, $\mathrm{P}^{\#\mathrm{P}} = \Delta_3^p$ implies that $\Sigma_k^p \subseteq \Delta_3^p$ for all k, that is, the polynomial hierarchy collapses to the third level. The next result shows that if the statement in Definition 5.1 is true (resp., false) then i) (resp., ii)) is true.

Theorem 5.1:

$$\begin{aligned} \mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{P/poly} &\Rightarrow \text{poly-size encoders/decoders} \\ &\Rightarrow \text{poly-size encoders} \\ &\Rightarrow \mathrm{P}^{\mathrm{Max\#P}} = \mathrm{P}^{\#\mathrm{P}} = \Delta_3^p. \end{aligned}$$

(We actually show that the final implication holds when restricted to G having finite memory.)

Definition 5.1 is a statement only about the *existence* of polynomial-size encoders/decoders; it says nothing about the time required to *find* a polynomial-size encoder/decoder. There is a version of Definition 5.1 and Theorem 5.1 that requires the encoder (resp., encoder/decoder) to be constructible by a polynomial-time algorithm.

[8] In general, for $k \geq 2$, $\Delta_k^p = \mathrm{P}^{\Sigma_{k-1}^p}$ and $\Sigma_{k-1}^p \subseteq \Delta_k^p \subseteq \Sigma_k^p$ [12].

Definition 5.2: The statement "max-rate block codes have polynomial-time-constructible encoders (resp., encoders/decoders)" means that there exists a polynomial-time algorithm $\mathcal{A}$ and a polynomial Q such that, for each DIF G and $q \in \boldsymbol{N}^+$, $\mathcal{A}(Q, q)$ produces an encoder circuit $\hat{\mathcal{E}}$ (resp., an encoder/decoder circuit pair $(\hat{\mathcal{E}}, \hat{\mathcal{D}})$) for (G, p, q) where $p = q \cdot MaxBlkRate(G, q)$ and $|\hat{\mathcal{E}}| \leq Q(|G|, q)$ (resp., $|\hat{\mathcal{E}}| + |\hat{\mathcal{D}}| \leq Q(|G|, q)$). This statement is abbreviated "poly-time-construct encoders (resp., encoders/decoders)."

Theorem 5.2:

$$\mathrm{P}^{\mathrm{Max\#P}} = \mathrm{P} \Leftrightarrow \text{poly-time-construct encoders/decoders}$$
$$\Leftrightarrow \text{poly-time-construct encoders.}$$

If we believe that the polynomial hierarchy does not collapse, then maximum-rate block codes do not always have polynomial-size encoders. However, the next result gives a condition under which polynomial-size encoders/decoders suffice by increasing q to decrease the rate. The increase in q occurs for two reasons: first, we assume $q \geq m$ where G has memory m; second, q is multiplied by a factor k depending on p and $|G|$ (for example, if $|G| \leq 2^{p/2}$ then $k = 2$).

Theorem 5.3: There is a constant c such that the following holds. Let G be a DIF, $q \in \boldsymbol{N}^+$, and $p = q \cdot MaxBlkRate(G, q)$, where G has finite memory m and $q \geq m$. Let integer $k \geq 2$ be such that $|G| \leq 2^{p(1-\frac{1}{k})}$. Then there is a rate $p : kq$ encoder/decoder circuit pair $(\hat{\mathcal{E}}, \hat{\mathcal{D}})$ for (G, p, kq) such that $|\hat{\mathcal{E}}| + |\hat{\mathcal{D}}| \leq c|G|(kq)^2$. Moreover, there is an algorithm that produces $(\hat{\mathcal{E}}, \hat{\mathcal{D}})$ in time $O(q|G|(|G| + qk^2))$.

The proof of Theorem 5.3 uses another result that says essentially that polynomial-size encoders/decoders always exist for sets of codewords that do not have to be $\mathcal{S}(G)$-concatenable and that are generated from a single state of G. For a DIF $G = (V, E)$, $q \in \boldsymbol{N}^+$, $s \in V$, and $T \subseteq V$, define $\mathcal{S}_q(G, s, T)$ to be the set of q-bit words y such that y is generated from s and $\delta(s, y) \in T$. The following result was shown independently by Kumar and Sivakumar [51]. It also has application to the design of encoder/decoders for individual states in state-dependent encodings (see Section VI).

Theorem 5.4: There is a constant c such that the following holds. Let $G = (V, E)$ be a DIF, $p, q \in \boldsymbol{N}^+$, $s \in V$, and $T \subseteq V$, where $|\mathcal{S}_q(G, s, T)| \geq 2^p$. There exists a $\mathcal{C} \subseteq \mathcal{S}_q(G, s, T)$ with $|\mathcal{C}| = 2^p$ and circuits $\hat{\mathcal{E}}$ and $\hat{\mathcal{D}}$ with $|\hat{\mathcal{E}}| + |\hat{\mathcal{D}}| \leq c|G|q^2$ such that the function $\mathcal{E}$ computed by $\hat{\mathcal{E}}$ is a one-to-one function from $\{0, 1\}^p$ onto $\mathcal{C}$, and the function computed by $\hat{\mathcal{D}}$ is the inverse of $\mathcal{E}$ on $\mathcal{C}$. Moreover, there is an algorithm that, when given input (G, p, q, s, T), runs within time $O(|G|q^2)$ and either outputs "not feasible" if $|\mathcal{S}_q(G, s, T)| < 2^p$, or otherwise outputs an $\hat{\mathcal{E}}$ and $\hat{\mathcal{D}}$ as above.

We now give the proofs of these results.

Proof of Theorems 5.1 and 5.2: Because the proofs of the two theorems are very similar, we first prove one implication of both and then prove the other implication of both. We first show that

$$\mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{P/poly} \Rightarrow \text{poly-size encoders/decoders.}$$

For a fixed q, we let "$\leq$" denote lexicographic ordering on q-bit words, that is, $y \leq y'$ iff $\mathrm{num}(y) \leq \mathrm{num}(y')$. If $S \subseteq \{0, 1\}^q$ and $y \in \{0, 1\}^q$, we define the *rank of y in S*, denoted $\mathrm{rank}(y, S)$, to be the number of $y' \in S$ with $y' \leq y$; thus, $0 \leq \mathrm{rank}(y, S) \leq |S|$ for all y and S.

Let tau-rank be the function with inputs of the form (y, G, τ) where $G = (V, E)$ is a DIF and $\tau \subseteq V$; the output is

$$\text{tau-rank}(y, G, \tau) \stackrel{\text{def}}{=} \mathrm{rank}(y, \mathcal{T}_{|y|}(\tau)).$$

We will use that this function belongs to #P; by the definition of rank and the definition of #P, this is shown by the ptime-relation R where $R((y, G, \tau), y') = 1$ iff $y' \leq y$ and $y' \in \mathcal{T}_{|y|}(\tau)$.

Let B be the decision problem whose instances are of the form (x, j, p, q, G, τ) where p, q, G, τ are as above, $x \in \{0, 1\}^p$, and $1 \leq j \leq q$. This is a "yes" instance iff $|\mathcal{T}_q(\tau)| \geq 2^p$ and (informally) the jth bit of the lexicographic encoding of x in $\mathcal{T}_q(\tau)$ equals 1; (formally) $\hat{y}_j = 1$ where $\hat{y} \in \mathcal{T}_q(\tau)$ is such that $\text{tau-rank}(\hat{y}, G, \tau) = \mathrm{num}(x) + 1$. The next goal is to show that B belongs to $\mathrm{P}^{\#\mathrm{P}}$. We describe a polynomial-time algorithm $\mathcal{A}$ that uses an oracle for tau-rank and decides B. Let (x, j, p, q, G, τ) be an instance of B. The algorithm $\mathcal{A}$ first asks the oracle for the value of $\text{tau-rank}(1^q, G, \tau)$; note that this equals $|\mathcal{T}_q(\tau)|$. If $|\mathcal{T}_q(\tau)| < 2^p$, then the answer is "no." Otherwise, $\mathcal{A}$ finds the (unique) lexicographically minimum $\hat{y} \in \{0, 1\}^q$ having $\text{tau-rank}(\hat{y}) = \mathrm{num}(x) + 1$; it does this by binary search over $y \in \{0, 1\}^q$, using the oracle to determine for a given y the relation between $\text{tau-rank}(y, G, \tau)$ and $\mathrm{num}(x) + 1$. Having found $\hat{y}$, $\mathcal{A}$ answers "yes" if $\hat{y}_j = 1$, or "no" otherwise.

There is a similar decision problem B' for decoding. Here, instances are of the form (y, i, p, q, G, τ) where $y \in \{0, 1\}^q$ and $1 \leq i \leq p$, and this is a "yes" instance iff $|\mathcal{T}_q(\tau)| \geq 2^p$ and the ith bit in the lexicographic decoding of y equals 1. To show membership of B' in $\mathrm{P}^{\#\mathrm{P}}$, the algorithm asks the oracle for $r = \text{tau-rank}(y, G, \tau)$, lets x be such that $\mathrm{num}(x) = r - 1$, and tests the ith bit of x.

Choose a fixed representation of instances (x, j, p, q, G, τ) and (y, i, p, q, G, τ) as binary words such that if $u \in \{0, 1\}^n$ represents such an instance, then $n = c(|G| + q)$ where c is a fixed constant (this is possible because $|x|, |y|, i, j \leq q$ and $|\tau| \leq |G|$). Because B and B' belong to $\mathrm{P}^{\#\mathrm{P}}$, they belong to P/poly by assumption. Let the polynomial $P(n)$ and the sequences $\{U_n : n \geq 1\}$ and $\{U'_n : n \geq 1\}$ of circuits be such that $|U_n|, |U'_n| \leq P(n)$ for all $n \geq 1$, $u \in B$ iff $U_{|u|}(u) = 1$, and $u \in B'$ iff $U'_{|u|}(u) = 1$ for all $u \in \{0, 1\}^+$. Define the polynomial Q by $Q(|G|, q) = 2q \cdot P(c(|G| + q))$.

We show that this Q satisfies the definition of "max-rate block codes have polynomial-size encoders/decoders." Fix an arbitrary DIF $\tilde{G}$ and $\tilde{q} \geq 1$. We have to describe an encoder/decoder pair $(\hat{\mathcal{E}}, \hat{\mathcal{D}})$ of circuits for $(\tilde{G}, \tilde{p}, \tilde{q})$ where $\tilde{p} \stackrel{\text{def}}{=} \tilde{q} \cdot MaxBlkRate(\tilde{G}, \tilde{q})$ and $|\hat{\mathcal{E}}| + |\hat{\mathcal{D}}| \leq Q(|\tilde{G}|, \tilde{q})$.

The circuit $\hat{\mathcal{E}}$ is obtained from $U_{\tilde{n}}$ where $\tilde{n} = c(|\tilde{G}| + \tilde{q})$. By Lemma 2.2, there exists a $\tilde{\tau}$ such that $\lfloor \log |\mathcal{T}_{\tilde{q}}(\tilde{\tau})| \rfloor = \tilde{p}$. For $1 \leq \tilde{\jmath} \leq \tilde{q}$, let $U^{(\tilde{\jmath})}$ be the circuit obtained from $U_{\tilde{n}}$ by setting the inputs for j, p, q, G, τ to the fixed values $\tilde{\jmath}, \tilde{p}, \tilde{q}, \tilde{G}, \tilde{\tau}$. Thus,

$U^{(\tilde{\jmath})}$ has $\tilde{p}$ inputs (the bits of x) and one output (the $\tilde{\jmath}$th bit in the lexicographic encoding of x in $\mathcal{T}_{\tilde{q}}(\tilde{\tau})$). Also,

$$|U^{(\tilde{\jmath})}| \leq |U_{\tilde{n}}| \leq P(\tilde{n}) = P(c(|\tilde{G}| + \tilde{q})) = Q(|\tilde{G}|, \tilde{q})(2\tilde{q}).$$

Combining $U^{(\tilde{\jmath})}$ for $1 \leq \tilde{\jmath} \leq \tilde{q}$ gives an encoder circuit $\hat{\mathcal{E}}$ of size at most $Q(|\tilde{G}|, \tilde{q})/2$. The decoder $\hat{\mathcal{D}}$ is obtained from $U'_{\tilde{n}}$ in a similar way, and $|\hat{\mathcal{E}}| + |\hat{\mathcal{D}}| \leq Q(|\tilde{G}|, \tilde{q})$.

For Theorem 5.2 we show

$$\mathrm{FP}^{\mathrm{Max\#P}} = \mathrm{FP} \Rightarrow \text{poly-time-construct encoders/decoders.}$$

We can replace "P" with "FP" because it is well known that for each oracle h, $\mathrm{FP}^h = \mathrm{FP}$ iff $\mathrm{P}^h = \mathrm{P}$. The proof is the same as above, except that there must be a fixed polynomial-time algorithm $\mathcal{A}$ that maps an arbitrary (G, q) to the circuits $(\hat{\mathcal{E}}, \hat{\mathcal{D}})$. The assumption $\mathrm{P}^{\mathrm{Max\#P}} = \mathrm{P}$ implies that B and B' belong to P, from which it follows that there is an algorithm that takes n to U_n and U'_n in time polynomial in n. The final parts of $\hat{\mathcal{E}}$ and $\hat{\mathcal{D}}$ that must be constructible in polynomial time are $\tilde{p}$ and $\tilde{\tau}$. Let $MaxDWL'(G, q, \tau')$ be the function whose value is $\lfloor \log MaxConcat'(G, q, \tau') \rfloor$, where $MaxConcat'(G, q, \tau')$ equals the maximum over all τ with $\tau' \subseteq \tau \subseteq V$ of $|\mathcal{T}_q(\tau)|$. $MaxDWL'$ belongs to $\mathrm{FP}^{\mathrm{Max\#P}}$, so it belongs to FP. Obviously, $\tilde{p} = MaxDWL'(\tilde{G}, \tilde{q}, \emptyset)$. Now $\tilde{\tau}$ can be found using a standard technique. Let $V = \{v_1, \ldots, v_{|V|}\}$ and start with $\tau' = \emptyset$. For $i = 1, 2, \ldots, |V|$ do: if $MaxDWL'(\tilde{G}, \tilde{q}, \tau' \cup \{v_i\}) = \tilde{p}$ then $\tau' \leftarrow \tau' \cup \{v_i\}$. Finally, $\tilde{\tau} = \tau'$.

We now show that

$$\text{poly-size encoders} \Rightarrow \mathrm{P}^{\mathrm{Max\#P}} \subseteq \Delta_3^p.$$

This suffices to prove the last implication in Theorem 5.1 because then

$$\mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{P}^{\mathrm{Max\#P}} \subseteq \Delta_3^p \subseteq \mathrm{P}^{\#\mathrm{P}}.$$

Let the polynomial Q be such that, for each G and q (where G can be restricted to have finite memory), there exists an encoder circuit $\hat{\mathcal{E}}$ of size at most $Q(|G|, q)$ that computes a (maximum-rate) encoder for (G, p, q), where

$$p = q \cdot MaxBlkRate(G, q).$$

Recall (Theorem 4.1) that the function $MaxBlkRate$ is $\mathrm{FP}^{\mathrm{Max\#P}}$-hard under $\leq_T^p$. Thus, to show $\mathrm{P}^{\mathrm{Max\#P}} \subseteq \Delta_3^p$ it suffices to show that $MaxBlkRate$ belongs to $\mathrm{F}\Delta_3^p$.

Recall from the remark following the proof of Theorem 2.3 that the decision problem GMBE is defined like MBE but without the promise that (G, p, q) is block feasible. That is, (G, p, q, K) is a "yes" instance of GMBE iff there is an encoder circuit of size $\leq K$ for (G, p, q).

We describe a polynomial-time algorithm $\mathcal{A}$ that uses an oracle for GMBE and computes the function $MaxBlkRate$. Because GMBE belongs to Σ_2^p, this shows by definition that $MaxBlkRate$ belongs to $\mathrm{F}\Delta_3^p$. Let (G, q) be an input to $MaxBlkRate$. Using the oracle, $\mathcal{A}$ finds the largest p such that $(G, p, q, Q(|G|, q))$ is a "yes" instance of GMBE. Because we have defined the size of an instance (G, p, q, K) to be greater than K, it is important that $\mathcal{A}$ calls the oracle only on instances where K is polynomially bounded in $|G|$ and q, and this is where we use the assumption that max-rate block codes have polynomial-size encoders. Now $MaxBlkRate(G, q) =$ this largest p divided by q.

For Theorem 5.2 we show

$$\text{poly-time-construct encoders} \Rightarrow \mathrm{P}^{\mathrm{Max\#P}} = \mathrm{P}.$$

By Theorem 4.1 it suffices to show that $MaxBlkRate$ belongs to FP. Given arbitrary (G, q), use the assumed polynomial-time algorithm to find a maximum-block-rate encoder for (G, q). The number of inputs to this encoder is $q \cdot MaxBlkRate(G, q)$.

Proof of Theorem 5.4: We describe an algorithm $\mathcal{A}$ that, when given (G, p, q, s, T), determines whether $|\mathcal{S}(G, q, s, T)| \geq 2^p$, and if so outputs an encoder and decoder of the required size for the lexicographic encoding.

Recall that if x is a p-bit word, then $\mathrm{num}(x)$ is the integer whose binary representation is x, so $0 \leq \mathrm{num}(x) < 2^p$. The *lexicographic encoder* maps x to the $(\mathrm{num}(x) + 1)$th lexicographically smallest word in $\mathcal{S}_q(G, s, T)$; that is, y where $\mathrm{rank}(y, \mathcal{S}_q(G, s, T)) = \mathrm{num}(x) + 1$.

Let $G = (V, E)$. The first phase of algorithm $\mathcal{A}$ is to find, for each $v \in V$ and each k with $0 \leq k \leq q$, the number of words y such that $|y| = k$, y is generated from v, and $\delta(v, y) \in T$; denote this number $\eta(v, k)$. Note that for each $v \in V$, there are at most 2^k paths of length k starting at v. Therefore, $\eta(v, k) \leq 2^k \leq 2^q$, so $\eta(v, k)$ can be written as a binary integer of length at most q. Because q can be arbitrarily large, we will charge the algorithm time ℓ for operations on binary integers of length ℓ.

The numbers $\eta(v, k)$ are computed iteratively. First, set $\eta(v, 0) = 1$ for all $v \in T$, and $\eta(v, 0) = 0$ for all $v \notin T$. Then there are q iterations numbered $k = 1, 2, \ldots, q$. At iteration k, do for all $v \in V$: let $\mathrm{next}(v)$ be the set of states reachable from v by following exactly one edge (so $0 \leq |\mathrm{next}(v)| \leq 2$); set

$$\eta(v, k) = \sum_{v' \in \mathrm{next}(v)} \eta(v', k-1)$$

(in particular, $\eta(v, k) = 0$ if $\mathrm{next}(v) = \emptyset$). Each iteration takes time $O(|G|)$, so the total time is $O(|G|q)$. If $\eta(s, q) < 2^p$, then $\mathcal{A}$ outputs "not feasible" and halts. Otherwise, $\mathcal{A}$ continues by constructing an encoder and decoder circuit for the lexicographic encoding. We first define the encoder and decoder as algorithms that run in time $O(|G|q^2)$, and then note that they can be implemented as circuits of size $O(|G|q^2)$.

For $1 \leq k \leq q$, let $\eta_0(v, k)$ be the number of k-bit words $y = 0y'$ generated from v. If there is an edge labeled 0 from state v to some state v', then $\eta_0(v, k) = \eta(v', k-1)$. If there is no such edge, then $\eta_0(v, k) = 0$. For $b = 0, 1$, recall that $\delta(v, b)$ is the state reached by following the edge labeled b from v. The encoding and decoding algorithms are described next. In the uses of $\delta(v, b)$ in these algorithms, there is guaranteed to be an edge (v, v', b).

Lexicographic Encoding Algorithm

Input: $x = x_1 \cdots x_p \in \{0, 1\}^p$.
Output: $y = y_1 \cdots y_q \in \{0, 1\}^q$.
1. $v \leftarrow s$ and $d \leftarrow \mathrm{num}(x)$.
2. Do for $i = 1, 2, \ldots, q$:
  a) Set $k = q - i + 1$.
  b) If $\eta_0(v, k) > d$, then set $y_i = 0$ and $v \leftarrow \delta(v, 0)$.
  c) Else, set $y_i = 1$, $d \leftarrow d - \eta_0(v, k)$, and $v \leftarrow \delta(v, 1)$.
3. Output $y = y_1 \cdots y_q$.

Lexicographic Decoding Algorithm
Input: $y = y_1 \cdots y_q \in \{0,1\}^q$.
Output: $x = x_1 \cdots x_p \in \{0,1\}^p$.
1. $v \leftarrow s$ and $d \leftarrow 0$.
2. Do for $i = 1, 2, \ldots, q$:
  a) Set $k = q - i + 1$.
  b) If $y_i = 0$, then set $v \leftarrow \delta(v, 0)$.
  c) If $y_i = 1$, then set $d \leftarrow d + \eta_0(v, k)$ and $v \leftarrow \delta(v, 1)$.
3. Output $x = x_1 \cdots x_p$ where $\mathrm{num}(x) = d$.

Both of these algorithms can be implemented by circuits of size $O(|G|q^2)$. Each circuit has q stages, where the input (resp., output) of the ith stage is v and d before (resp., after) the ith iteration. The state v can be represented as a binary word of length $\lceil \log |G| \rceil$, and the integer d can be represented as a binary integer of length q. It is not hard to see that a stage can be implemented using $O(|G|q)$ gates. So the total circuit size is $O(|G|q^2)$.

Remark: For both the encoder and the decoder circuit, the circuitry has the same form at each stage, but it must be copied q times because our model of circuits does not allow cycles. In the model of synchronous cyclic circuits, the same circuitry can be reused at each stage, so the circuit size is $O(|G|q)$.

Proof of Theorem 5.3: Let c be the constant of Theorem 5.4. Let G be a DIF, $q \in \boldsymbol{N}^+$, and

$$p = q \cdot MaxBlkRate(G, q)$$

where G has memory m and $q \geq m$. Let integer $k \geq 2$ be such that $|G| \leq 2^{p(1-\frac{1}{k})}$. We show that there is a rate $p : kq$ encoder/decoder circuit pair $(\hat{\mathcal{E}}, \hat{\mathcal{D}})$ for (G, p, kq) such that $|\hat{\mathcal{E}}| + |\hat{\mathcal{D}}| \leq c|G|(kq)^2$. Moreover, there is an algorithm that produces $(\hat{\mathcal{E}}, \hat{\mathcal{D}})$ in time $O(q|G|(|G| + qk^2))$, given G, q, k.

For each $v \in V$ compute

$$\eta(v) \stackrel{\mathrm{def}}{=} |\{y \in \{0,1\}^q \mid \delta(v, y) = v\}|.$$

As in the proof of Theorem 5.4, $\eta(v)$ for a fixed v can be computed in time $O(|G|q)$, so the total time to compute $\eta(v)$ for all $v \in V$ is $O(|G|^2 q)$. Let $\hat{v} = \arg\max_{v \in V} \eta(v)$. We claim that $\eta(\hat{v}) \geq 2^{p/k}$. To prove this, again by Lemma 2.2, let τ be such that $|\mathcal{T}_q(\tau)| \geq 2^p$.[9] Because $q \geq m$, for each q-bit y there is a unique t_y such that, for all $v \in V$, if y is generated from v then $\delta(v, y) = t_y$. So there must exist a set $S \subseteq \mathcal{T}_q(\tau)$ and a $\hat{t} \in \tau$ such that $|S| \geq 2^p/|\tau|$ and $t_y = \hat{t}$ for all $y \in S$. All words in S are generated from all states in τ, so in particular they are all generated from $\hat{t}$. Therefore,

$$\eta(\hat{t}) \geq |S| \geq 2^p/|\tau| \geq 2^p/|G| \geq 2^{p/k},$$

the last inequality following from the assumption $|G| \leq 2^{p(1-\frac{1}{k})}$. Because $\hat{v}$ was chosen to maximize $\eta(v)$, we have $\eta(\hat{v}) \geq 2^{p/k}$.

Define

$$C' = \{y \in \{0,1\}^q \mid \delta(\hat{v}, y) = \hat{v}\}$$

and

$$C = \{y_1 y_2 \cdots y_k \mid y_1, y_2, \ldots, y_k \in C'\}.$$

[9]In the case $q \geq m$, Lemma 2.2 has a shorter proof, given in [28].

Now $|C'| = \eta(\hat{v}) \geq 2^{p/k}$ and $|C| = |C'|^k \geq 2^p$. It follows that $|\mathcal{S}_{kq}(G, \hat{v}, \{\hat{v}\})| \geq 2^p$, and obviously every subset of $\mathcal{S}_{kq}(G, \hat{v}, \{\hat{v}\})$ is a $\mathcal{S}(G)$-concatenable set of kq-bit words. The proof is now complete by applying Theorem 5.4.

VI. Conclusion and Open Questions

We have classified the computational complexities of several basic problems from the field of constrained block coding. The classification was done in terms of the complexity classes Σ_2^p, Σ_3^p, and $\mathrm{NP}^{\#\mathrm{P}}$, which lie "higher" in the complexity class structure than NP. The classes Σ_2^p, Σ_3^p, and $\#\mathrm{P}$ have been known and studied for over 20 years by researchers in complexity theory, although the "natural" problems known to be Σ_2^p-, Σ_3^p-, or $\mathrm{NP}^{\#\mathrm{P}}$-complete are much rarer than those known to be NP-complete. We have shown also that the question of whether the maximum rate can always be achieved using polynomial-size encoders and decoders is very closely connected to the relationships among certain complexity classes, and that either answer to this question would imply a breakthrough in complexity theory.

Three areas for future work are discussed next.

A. Fixed FSTD

We have pointed out that our hardness results rely on the FSTD G being part of the instance (input). As is usually done in complexity theory, a hardness result for the problem A is proved by showing, for some complexity class $\mathcal{L}$ and some $\mathcal{L}$-hard problem B, that B is reducible to A. For the reducibilities in this paper, this is done by mapping each instance u of B to an instance $(G, \ldots)$ of A, where intuitively the structure of u is encoded in the structure of G; for example, the structure of a Boolean formula is encoded using the graphs shown in Figs. 2 and 3. Because $|u|$ is not bounded, such proofs require that $|G|$ not be bounded. This raises the question of the complexity of these problems when G is fixed to some G_0. Clearly, the answer can depend on G_0; for example, if $\mathcal{S}(G_0) = \{0,1\}^+$ then MBE, MBD, MAXBLKRATE, etc., are trivial. However, for a complexity class $\mathcal{L}$ it is reasonable to ask whether there is some fixed G_0 such that the problem is $\mathcal{L}$-hard or $\mathcal{L}$-complete when instances are restricted to those where the DIF is fixed to G_0. Unfortunately, there are obstacles to proving such a result, which follow easily from known results.

Proposition 6.1: Let G_0 be a DIF and let A be one of the decision problems MBE, MBD, MBE+D, MBE&D, MAXBLKRATE, where the DIF G in the instance is fixed to G_0. If A is NP-hard under $\leq_m^p$ then $\mathrm{P} = \mathrm{NP}$. If A is NP-hard under $\leq_T^p$ then the polynomial hierarchy collapses to the second level.

Proof: In this proof, we will consider decision problems where each instance is represented as a word over an alphabet of size one. Letting σ denote the alphabet symbol and $\ell \geq 1$, σ^ℓ denotes the word of length ℓ in $\{\sigma\}^+$. A subset of $\{\sigma\}^+$ is called an *SLA* (*single letter alphabet*) *problem* or a *tally problem*. Let φ_2 be the "pairing function"

$$\varphi_2(i_1, i_2) = i_1 + (i_1 + i_2)(i_1 + i_2 + 1)/2.$$

It is easy to see that φ_2 is a one-to-one function from $\boldsymbol{N}^2$ onto $\boldsymbol{N}$, and that there is an algorithm that maps (i_1, i_2) to $\sigma^{\varphi_2(i_1,i_2)}$ and runs in time polynomial in i_1 and i_2. Thus, for each $k \geq 3$

$$\varphi_k(i_1, \ldots, i_k) \stackrel{\text{def}}{=} \varphi_2(\varphi_{k-1}(i_1, \ldots, i_{k-1}), i_k)$$

is a one-to-one, polynomial-time computable function from $\boldsymbol{N}^k$ onto $\boldsymbol{N}$.

Note that when the DIF G is fixed, an instance of one of the problems MBE, MBD, etc., consists of a fixed number of (at most four) nonegative integers. In the case of MBE, for example, there are three integers p, q, K. We give the proof for $A =$ MBE; the proofs for the other problems are virtually identical. Define the SLA problem A' to be the set of words σ^ℓ such that $\ell = \varphi_3(p, q, K)$ where (p, q, K) is a "yes" instance of A. Because φ_3 is one-to-one and polynomial-time computable, it is immediate that $A \leq_m^p A'$. Thus, if A is NP-hard under $\leq_m^p$ (resp., $\leq_T^p$) then A' is NP-hard under $\leq_m^p$ (resp., $\leq_T^p$). Berman [52] has shown that if there exists an SLA problem that is NP-hard under $\leq_m^p$, then P = NP. Karp and Lipton [38] have shown that if there exists an SLA problem that is NP-hard under $\leq_T^p$, then the polynomial hierarchy collapses to the second level. □

This proof relies on the convention that the "size" of an integer z is z, rather than $\log z$ (see Section II-C). If the latter convention is used, it is conceivable that one of the problems MBE, MBD, etc., could be proved NP-hard without the consequence of the polynomial hierarchy collapsing.

Regarding upper bounds, the G_0-fixed versions of $MaxConcat$ and $MaxBlkRate$ now belong to $\mathrm{P}^{\#\mathrm{P}}$, a small improvement from $\mathrm{P}^{\mathrm{Max}\#\mathrm{P}}$ (the reason is that, letting $G_0 = (V_0, E_0)$, there are a constant number of subsets $\tau \subseteq V_0$). We have not investigated whether this upper bound can be improved further or whether the upper bounds of Theorem 2.3 for MBE, MBD, MBE+D, and MBE&D can be improved in the G_0-fixed case. To summarize the situation for fixed G_0, there are large gaps between known lower and upper bounds, and it is unlikely that currently known methods in complexity theory can improve the lower bounds.

B. Improvements to the Results

A technical question that remains open is the precise complexity classifications of MBE and MBE+D. We have shown a "lower bound" of NP-hardness (Theorem 3.2) and an "upper bound" of Σ_2^p (Theorem 2.3). Because we have shown that MBE&D is Σ_2^p-complete under $\leq_{\mathrm{PR}}$, it is reasonable to conjecture that MBE and MBE+D are Σ_2^p-complete, at least under $\leq_{\mathrm{PR}}$ and possibly under $\leq_m^p$ as well. Another open question is whether MBE&D is Σ_2^p-complete (under $\leq_m^p$).

Another area of open questions is motivated by the fact that, for the hardness results involving the size of a decoder circuit (Theorems 3.1, 3.3, and 3.4), the size of the decoder is restricted to zero, and this restriction plays a crucial role in the proofs. On the one hand, proving hardness of MBD with this restriction is technically stronger than proving hardness of MBD without this restriction. But on the other hand one could argue that this restriction misses practically interesting cases where the size of the decoder is significantly greater than zero. Even in the hardness proof for MBE (Theorem 3.2), in case II) it is shown only that the size of the encoder circuit is at least m', where m' in the proof is significantly smaller than the codeword length q. One could investigate the complexities of MBE, MBD, MBE+D, and MBE&D, where the targets K, K_1, K_2 are restricted to other ranges, say, $K \geq q$.

C. Finite-State Encoding

Taking a wider view, a future direction for research into the complexity of problems in constrained coding is to consider finite-state encoding in general (with block encoding being the special case where the encoder has only one state). We have previously noted that there is an NP-completeness result in this area [13].

Some preliminary results on the complexity of designing finite-state encoders/decoders follow from the results and proofs of this paper. A subproblem in the design of a finite-state encoder for a given (G, p, q) is to design a (block) encoder for a particular state s of G. In this case, an encoder $\mathcal{E}$ is a mapping from the set of p-bit datawords $(\{0, 1\}^p)$ to the set of q-bit codewords that are generated from s $(\mathcal{S}_q(G, s, V))$; the range of $\mathcal{E}$ does not have to be $\mathcal{S}(G)$-concatenable. Theorem 5.4 shows that the maximum p such that $2^p \leq |\mathcal{S}_q(G, s, V)|$ can be found in polynomial time. Moreover, the lexicographic encoder and decoder circuits, between $\{0, 1\}^p$ and a subset of $\mathcal{S}_q(G, s, V)$, have polynomial size.

Some of the hardness results for minimizing the size of encoders and decoders for block codes hold also for problems of minimizing the size of encoders and decoders for individual states in state-dependent encoder/decoders, by essentially the same proofs. The definitions of the problems MBE, MBD, etc., in this case are obtained by including the starting state s in the instance and dropping the requirement that the codebook be $\mathcal{S}(G)$-concatenable. The hardness results that can be translated easily to this case are those where the special DIF described in the proof has only one synchronizing state. These results are Theorems 3.1, 3.2, and 3.3, and parts 1)–4) of Corollary 3.5. We simply let s in the instance be the unique synchronizing state. In fact, the proofs are easier in this case because synchronization is not an issue: we can work directly over the alphabet $\{0, 1\}$, and θ is not needed.

The treatment of state-dependent coding above is oversimplified because the design of a finite-state encoder/decoder is actually a global process that includes all states; optimizing locally at each state individually might give inferior global designs. For example, to achieve the desirable property of sliding-block decodability, the problem of dataword-to-codeword assignment must be solved globally; the assignment at each state must be "consistent" with the assignments at other states. Similarly, when designing a circuit for a finite-state encoder, a mapping from (state, dataword) to codeword, viewing the problem globally allows sharing of hardware among different states and thus can yield smaller circuits than those that contain a separate encoder circuit for each state. Neither the upper bounds nor the lower bounds that we have established, for the complexity of minimizing block encoder/decorder size and computing the maximum block rate, automatically carry over to the case of finite-state encoders/decoders. An interesting area for study is the complexity of these problems for finite-state encoders/decoders

as well as the complexity of other problems that arise in the design of efficient finite-state encoders/decoders.

REFERENCES

[1] K. A. S. Immink, *Coding Techniques for Digital Recorders*. Englewood Cliffs, NJ: Prentice-Hall, 1991.

[2] B. H. Marcus, P. H. Siegel, and J. K. Wolf, "Finite-state modulation codes for data storage," *IEEE J. Select. Areas Commun.*, vol. 10, pp. 5–37, Jan. 1992.

[3] K. A. S. Immink, P. H. Siegel, and J. K. Wolf, "Codes for digital recorders," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2260–2299, Oct. 1998.

[4] G. D. Forney, Jr., B. Marcus, N. T. Sindhushayana, and M. Trott, "Multilingual dictionary: System theory, coding theory, symbolic dynamics, and automata theory," in *Proc. Symp. Applied Mathematics*, vol. 50, 1995, pp. 109–138.

[5] B. Marcus, "Symbolic dynamics and connections to coding theory, automata theory, and system theory," in *Proc. Symp. Applied Mathematics*, vol. 50, 1995, pp. 95–108.

[6] B. H. Marcus, R. M. Roth, and P. H. Siegel, "Constrained systems and coding for recording channels," in *Handbooks of Coding Theory*, V. S. Pless and W. C. Huffman, Eds. Amsterdam, The Netherlands: Elsevier, 1998, ch. 20.

[7] L. G. Valiant, "The complexity of computing the permanent," *Theoret. Comp. Sci.*, vol. 8, pp. 189–202, 1979.

[8] D. S. Modha and B. H. Marcus, "Art of constructing low-complexity encoders/decoders for constrained block codes," *IEEE J. Select. Areas Commun.*, vol. 19, pp. 619–634, Apr. 2001.

[9] J. S. Eggenberger and A. M. Patel, "Method and apparatus for implementing PRML codes," U.S. Patent 4 707 681, Nov. 17, 1987.

[10] B. H. Marcus, A. M. Patel, and P. H. Siegel, "Method and apparatus for implementing a PRML code," U.S. Patent 4 786 890, Nov. 22, 1988.

[11] R. L. Galbraith, "Method and apparatus for implementing PRML codes with maximum ones," U.S. Patent 5 196 849, Mar. 13, 1993.

[12] L. J. Stockmeyer, "The polynomial-time hierarchy," *Theoret. Comp. Sci.*, vol. 3, pp. 1–22, 1977.

[13] J. J. Ashley, R. Karabed, and P. H. Siegel, "Complexity and sliding-block decodability," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1925–1947, Nov. 1996.

[14] E. R. Berlekamp, R. J. McEliece, and H. C. A. van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 384–386, May 1978.

[15] A. Vardy, "Algorithmic complexity in coding theory and the minimum distance problem," in *Proc. 29th ACM Symp. Theory of Computing*, 1997, pp. 92–109.

[16] M. Sudan, "Algorithmic issues in coding theory," in *17th Conf. Foundations of Software Technology and Theoretical Computer Science*, Kharagpur, India, 1997. Invited Paper.

[17] G. B. Horn and F. R. Kschischang, "On the intractability of permuting a block code to minimize trellis complexity," *IEEE Trans. Inform. Theory*, vol. 42, pp. 2042–2048, Nov. 1996.

[18] F. R. Kschischang and V. Sorokine, "On the trellis structure of block codes," *IEEE Trans. Inform. Theory*, vol. 41, pp. 1924–1937, Nov. 1995.

[19] C. Umans, "The minimum equivalent DNF problem and shortest implicants," in *Proc. 39th IEEE Symp. Foundations of Computer Science*, 1998, pp. 556–563.

[20] M. Schaefer, "Graph Ramsey theory and the polynomial hierarchy," in *Proc. 31st ACM Symp. Theory of Computing*, 1999, pp. 592–601.

[21] ——, "Deciding the Vapnik–Červonenkis dimension is Σ_3^p-complete," *J. Comput. Syst. Sci.*, vol. 58, pp. 177–182, 1999.

[22] K. W. Wagner, "The complexity of combinatorial problems with succinct input representation," *Acta Inform.*, vol. 23, pp. 325–356, 1986.

[23] L. J. Stockmeyer and A. R. Meyer, "Word problems requiring exponential time," in *Proc. 5th ACM Symp. Theory of Computing*, 1973, pp. 1–9.

[24] J. L. Bentley, T. Ottmann, and P. Widmayer, "The complexity of manipulating hierarchically defined sets of rectangles," *Adv. Comput. Res.*, vol. 1, pp. 127–158, 1983.

[25] M. Mundhenk, J. Goldsmith, C. Lusena, and E. Allender, "Complexity of finite-horizon Markov decision process problems," *J. Assoc. Comput. Mach.*, vol. 47, pp. 681–720, 2000.

[26] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.

[27] D. S. Johnson, "A catalog of complexity classes," in *Handbook of Theoretical Computer Science, Vol. A, Algorithms and Complexity*, J. V. Leeuwen, Ed. Cambridge, MA: Elsevier/MIT Press, 1990.

[28] C. V. Freiman and A. D. Wyner, "Optimum block codes for noiseless input restricted channels," *Inform. Contr.*, vol. 7, pp. 398–415, 1964.

[29] J. E. Savage, *The Complexity of Computing*. New York: Wiley, 1976.

[30] I. Wegener, *The Complexity of Boolean Functions*. New York: Wiley, 1987.

[31] S. Even and Y. Yacobi, "Cryptography and NP-completeness," in *Proc. 7th Colloq. Automata, Languages, and Programming (Lecture Notes in Computer Science}*. New York: Springer-Verlag, 1980, vol. 85, pp. 195–207.

[32] S. Even, A. L. Selman, and Y. Yacobi, "The complexity of promise problems with applications to public-key cryptography," *Inform. Contr.*, vol. 61, pp. 159–173, 1984.

[33] A. R. Meyer and L. J. Stockmeyer, "The equivalence problem for regular expressions with squaring requires exponential space," in *Proc. 13th IEEE Symp. Switching and Automata Theory*, 1972, pp. 125–129.

[34] C. Wrathall, "Complete sets and the polynomial-time hierarchy," *Theor. Comput. Sci.*, vol. 3, pp. 23–33, 1977.

[35] U. V. Vazirani and V. V. Vazirani, "A natural encoding scheme proved probabilistic polynomial complete," *Theor. Comput. Sci.*, vol. 24, pp. 291–300, 1983.

[36] L. M. Adleman and K. Manders, "Reductions that lie," in *Proc. 20th IEEE Symp. Foundations of Computer Science*, 1979, pp. 397–410.

[37] L. Adleman, "Two theorems on random polynomial time," in *Proc. 19th IEEE Symp. Foundations of Computer Science*, 1978, pp. 75–83.

[38] R. M. Karp and R. J. Lipton, "Some connections between nonuniform and uniform complexity classes," in *Proc. 12st ACM Symp. Theory of Computing*, 1980, pp. 302–309.

[39] L. A. Hemaspaandra, A. Hoene, A. V. Naik, M. Ogihara, A. L. Selman, T. Thierauf, and J. Wang, "Nondeterministically selective sets," *Intl. J. Found. Comput. Sci.*, vol. 6, pp. 403–416, 1995.

[40] S. A. Cook, "The complexity of theorem proving procedures," in *3rd ACM Symp. Theory of Computing*, 1971, pp. 151–158.

[41] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. New York: Plenum, 1972, pp. 85–103.

[42] J. Håstad, "Clique is hard to approximate within $n^{1-\epsilon}$," in *Proc. 37th IEEE Symp. Foundations of Computer Science*, 1996, pp. 627–636.

[43] M. Bellare, O. Goldreich, and M. Sudan, "Free bits, PCP's, and nonapproximability—Towards tight results," *SIAM J. Comput.*, vol. 27, pp. 804–915, 1998.

[44] C. Berge, *Graphs*, 2nd rev. ed. Amsterdam, The Netherlands: Elsevier, 1985.

[45] C. Umans, "Hardness of approximating Σ_2^p minimization problems," in *Proc. 40th IEEE Symp. Foundations of Computer Science*, 1999, pp. 465–474.

[46] S. Toda, "PP is as hard as the polynomial-time hierarchy," *SIAM J. Comput.*, vol. 20, pp. 865–877, 1991.

[47] J. Torán, "Complexity classes defined by counting quantifiers," *J. Assoc. Comput. Mach.*, vol. 38, pp. 753–774, 1991.

[48] J. Gill, "Computational complexity of probabilistic turing machines," *SIAM J. Comput.*, vol. 6, pp. 675–695, 1977.

[49] M. W. Krentel, "The complexity of optimization problems," *J. Comput. Syst. Sci.*, vol. 36, pp. 490–509, 1988.

[50] J. Simon, "On some central problems in computational complexity," Ph.D. dissertation, Dept. Comput. Sci., Cornell Univ., Ithaca, NY, 1975.

[51] R. Kumar, private communication.

[52] P. Berman, "Relationship between density and deterministic complexity of NP-complete languages," in *5th Intl. Colloq. Automata, Languages, and Programming (Lecture Notes in Computer Science)*. Berlin, Germany: Springer-Verlag, 1978, vol. 62, pp. 63–71.