Any Code of Which We Cannot Think Is Good

JOHN T. COFFEY AND RODNEY M. GOODMAN

Abstract—**A central paradox of coding theory has been noted for many years, and concerns the existence and construction of the best codes. Virtually every linear code is "good" in the sense that it meets the Gilbert–Varshamov bound on distance versus redundancy. Despite the sophisticated constructions for codes derived over the years, however, no-one has succeeded in demonstrating a constructive procedure that yields such codes over arbitrary symbol fields. A quarter of a century ago, Wozencraft and Reiffen, in discussing this problem, stated that "we are tempted to infer that any code of which we cannot think is good." Using the theory of Kolmogorov complexity, we show the remarkable fact that his statement holds true in a rigorous mathematical sense: any linear code that is truly random, in the sense that there is no concise way of specifying the code, is good. Furthermore, random selection of a code that does contain some constructive pattern results, with probability bounded away from zero, in a code that does not meet the Gilbert–Varshamov bound *regardless of the block length of the code*. In contrast to the situation for linear codes, it is shown that there are effectively random nonlinear codes which have no guarantee on distance. In addition, it is shown that the techniques of Kolmogorov complexity can be used to derive typical properties of classes of codes in a novel way.**

I. Introduction

Ever since the pioneering work of Shannon, the existence of good codes for arbitrary information channels has been known. Shannon's proof relies on the idea of picking a code at random, and showing that such a code is good with high probability; unfortunately, this gives us no indication of how such codes are to be constructed. For the more restricted case where we use Hamming distance to measure the "goodness" of a code, an early result of Gilbert shows that a certain tradeoff of distance versus rate is possible [1], [2]. Asymptotically, we have

$$H_q(d/n) \geq 1 - R + o(1)$$

for the best codes over GF(q), where $H_q(x)$ is the q-ary entropy function. Indeed, it can be shown that virtually every linear code satisfies the Gilbert–Varshamov bound—a code picked "at random" satisfies the bound with probability asymptotically approaching one. The work of constructive coding theory starts with this premise and seeks to synthesize the codes. However, the task seems extraordinarily difficult. Although it is possible to construct infinite families of codes that have both rate and relative distance bounded away from zero, there has until relatively recently been no known constructive procedure for obtaining codes which meet the Gilbert–Varshamov bound over any symbol field. The recent breakthrough in codes obtained from algebraic geometry has given such constructions for relatively large symbol fields ($q \geq 49$) but so far there has been no corresponding progress for smaller symbol fields. This phenomenon has often been noted as a paradox of coding theory. Writing a quarter of a century ago, Wozencraft and Reiffen summed up the attitude of many information and coding theorists [3]:

"It is unfortunately true that the search for good codes with large $|S|$ has thus far been unrewarding. However, as we have seen, almost all codes are good. Thus we are tempted to infer that any code of which we cannot think is good."

In this correspondence, we demonstrate the remarkable fact that the last statement can be shown to be true in a strict formal sense in the case of linear codes. Using the theory of Kolmogorov complexity, we show that those codes which are truly "random," in the sense that there is no method for specifying the code that is significantly more concise than simply writing out the symbols of the generator matrix, must meet the Gilbert–Varshamov bound. It follows that virtually all linear codes meet the bound, because virtually all such codes are effectively patternless.

Another consequence of this result concerns the probability of picking a bad code. (Henceforth a code is "good" if it meets the Gilbert–Varshamov bound and "bad" otherwise). If we pick a code at random, the probability of picking a bad code goes to zero exponentially. If we insist that the code has some minimum amount of structure, and then make a random selection from such codes, the probability of picking a bad code is much greater than in the case of random selection from all codes. This is because we have excluded very many (in fact, virtually all) codes which are good, without excluding any bad codes. We show that the probability of picking a bad code, given a random selection from the set of codes that have some minimum amount of structure, is bounded away from zero regardless of the block length. Thus random selection from the codes we are most likely to think of is "quite likely" to produce a bad code. These results hold even if we add the condition that the code is recoverable from its compressed specification in polynomial time.

A natural asymptotic form of the encoding problem is to synthesize an *infinite family* of good codes using some fixed procedure: we should be able to specify some fixed list of instructions, which, together with an integer m, would be sufficient to generate the mth code in the infinite sequence. We will say that any such family of codes is *computable*. If the procedure is executed in a time upper-bounded by a polynomial in m, we will say that the family of codes is *practically computable*. It is a consequence of our results that although virtually all infinite families of codes are good, virtually all are also uncomputable in the above sense. In addition, there is no reason to believe that any infinite family of codes is practically computable.

The general problem of deciding the complexity of producing the best of various classes of codes (using various other measures of complexity) has attracted much interest [22]–[25]—indeed, Bassalygo *et al.* [23] assert that these "can now rightly be regarded as pivotal problems in the theory of correcting codes." We feel that the application of the techniques of Kolmogorov complexity opens up a new avenue of inquiry into the encoding problem. In addition, the techniques provide a novel and intuitively appealing way of analyzing the typical behavior of classes of codes.

In Section II, we discuss the basic axioms of complexity theory. In Section III, we outline the conventional proofs of the Gilbert–Varshamov bound and discuss some basic related results. In Section IV, the main results are obtained in which the distance and randomness properties of a code are shown to be related. It is shown that for linear codes, all effectively random codes must meet the Gilbert–Varshamov bound, and that a weaker converse also holds. In Section V, we discuss the case of general nonlinear codes, and show that the result does not apply to this class: there are effectively random nonlinear codes that

Manuscript received July 14, 1988; revised March 20, 1990.

J. T. Coffey was with the California Institute of Technology. He is now with the Department of Electrical Engineering and Computer Science, University of Michigan, 1301 N. Beal Ave, Ann Arbor, MI 48109-2122.

R. M. Goodman is with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125.

IEEE Log Number 9036750.

have no guarantee on distance. In Section VI, we show how some aspects of the typical behavior of classes of codes can be derived from a characterization of codes in terms of their complexity.

II. Kolmogorov Complexity

We begin by discussing a general model of computation [4], [5]. Informally, a *Turing machine* (TM) consists of a finite state machine, a read–write head, and a two-way infinite tape. The tape is ruled into cells, and each cell is occupied by a symbol from a fixed alphabet $\Sigma = \{\sigma_1, \cdots, \sigma_K\}$ or else the cell is blank. We denote the blank by σ_0. The fixed control performs one of the following actions: it can erase the current symbol on the tape; it can overprint a new symbol; or it can move right or move left one cell. The states of the finite control are $\{q_0, q_1, \cdots, q_P\}$, and two states are distinguished: q_0 is the starting state, and q_P is the halting state. The computation continues until the state q_P is reached. Then the computation is over and the output is whatever is written on the tape. The action of the Turing machine is specified by its "next move" function that specifies, given a state and a symbol, what state the finite control moves to and what action it takes. More formally, a Turing machine is defined to be a triple $M = (P, K, \delta)$ where P and K are positive integers and δ is a function

$$\delta: \{q_0, \cdots, q_{P-1}\} \times \{\sigma_0, \cdots, \sigma_K\} \rightarrow \{q_0, \cdots, q_P\} \times \{\sigma_0, \cdots, \sigma_K, L, R\}.$$

We characterize the current status of a Turing machine by its "instantaneous description."

Definition: The *instantaneous description* (ID) of a Turing machine is a quadruple $(q_{p_1}, \boldsymbol{u}, \sigma_{k_1}, \boldsymbol{v})$.

Informally, the machine is in state q_{p_1}, the symbol under the read–write head is σ_{k_1}, the string $\boldsymbol{u}$ is on the tape to the left of the read–write head, and the string $\boldsymbol{v}$ is to the right of the read–write head. (The string $\boldsymbol{u}$ is taken to begin at the leftmost nonblank symbol on the tape; the string $\boldsymbol{v}$ is taken to end at the rightmost nonblank symbol.)

Given a certain instantaneous description, ID_1, the next-move function of the Turing machine determines uniquely what the next instantaneous description ID_2 will be. We write

$$(q_{p_1}, \boldsymbol{u}_1, \sigma_{k_1}, \boldsymbol{v}_1) \rightarrow_M (q_{p_2}, \boldsymbol{u}_2, \sigma_{k_2}, \boldsymbol{v}_2).$$

We define the relation $\rightarrow_M^t$ on the set of IDs for $t \in Z^+$ recursively: $ID_1 \rightarrow_M^{t+1} ID_2$ for $t > 1$ iff there is an ID such that $ID_1 \rightarrow_M ID$ and $ID \rightarrow_M^t ID_2$.

Informally, $ID_1 \rightarrow_M^t ID_2$ if we go from ID_1 to ID_2 in t steps on the machine M. The relation $\rightarrow_M^*$ is defined on IDs as $ID_1 \rightarrow_M^* ID_2$ iff there exists $t \in Z^+$ such that $ID_1 \rightarrow_M^t ID_2$.

The importance of Turing machines is evident from Church's thesis: *Any algorithm can be rendered as a Turing machine.* Although this statement is unprovable, relating as it does a mathematical concept to the nonmathematical concept of "computability," it is virtually universally accepted as a *de facto* definition of computability.

Definition: A function $f: \Sigma_1^* \rightarrow \Sigma_2^*$ is said to be computable iff there exists a Turing machine $M = (P, K, \delta)$ such that $\Sigma_1 \cup \Sigma_2 \subseteq \{\sigma_0, \cdots, \sigma_K\}$ and for every $\boldsymbol{x} \in \Sigma_1^*$, we have $(q_0, \lambda, \sigma_0, \boldsymbol{x}) \rightarrow_M^* (q_p, \lambda, \sigma_0, \boldsymbol{y})$ where $\boldsymbol{y} = f(\boldsymbol{x})$ and λ is the null string.

A *universal Turing machine* (UTM) is, informally, a general purpose Turing machine. It takes as input a string $\boldsymbol{x} = \rho(M)\rho(\boldsymbol{w})$ where $\rho(M)$ is an encoding of a Turing machine M, and $\rho(\boldsymbol{w})$ is an encoding of the input to that Turing machine, and simulates the action of M on $\boldsymbol{w}$. More exactly, the UTM takes the input string, checks to see if it is of the form $\rho(M)\rho(\boldsymbol{w})$, (if not, it goes into an infinite loop), simulates the action of M on $\boldsymbol{w}$, and if M would halt with output $\boldsymbol{y}$, then U also halts with the same output.

For convenience, we fix a universal Turing machine that accepts inputs in an alphabet of size q, and that has the lowest possible number of states in the finite control. Clearly, $\rho(M)$ must have a certain structure if it is to represent a Turing machine. The q-ary input $\rho(\boldsymbol{w})$ needs no such structure, however, so $\rho(\boldsymbol{w}) = \boldsymbol{w}$ in this formulation.

Suppose that on a given input $\boldsymbol{x}$, the UTM halts, leaving the string $\boldsymbol{y}$ to the right of the read–write head. We say that $\boldsymbol{v}$ is computed by U on $\boldsymbol{x}$. We define the *Kolmogorov* (or *Kolmogorov–Chaitin*) *complexity* [6], [7] of a string $\boldsymbol{s}$ to be the length of the shortest input to the universal Turing machine U such that U accepts the input string and eventually halts leaving $\boldsymbol{s}$ on the tape to the right of the read–write head.

Definition: The Kolmogorov (or Kolmogorov–Chaitin) complexity of a string $\boldsymbol{s}$ is a function K: $\{0, 1, \cdots, q-1\}^* \rightarrow Z$ defined by

$$K(\boldsymbol{s}) = \min\{|\boldsymbol{p}| \| (q_0, \lambda, \sigma_0, \boldsymbol{p}) \rightarrow_U^* (q_P, \lambda, \sigma_0, \boldsymbol{s})\}.$$

The following theorem summarizes the main properties of this function.

Theorem 1:

1) There exists a constant c_0 such that $K(\boldsymbol{s}) \leq n + c_0$ for any $\boldsymbol{s}$ and $n = |\boldsymbol{s}|$.
2) The fraction of n-tuples $\boldsymbol{s}$ with $K(\boldsymbol{s}) < n - c_1$ is less than q^{-c_1}.
3) The Kolmogorov complexity of a string is, in general, uncomputable.
4) If $\boldsymbol{s}$ has length n and weight λn, then $K(\boldsymbol{s}) \leq nH_q(\lambda) + o(n)$ for large n, where $H_q(x)$ is the q-ary entropy function $-x \log_q x - (1-x)\log_q(1-x) + \log_q(q-1)$ for $0 < x < 1$.

Proof:

a) Consider the everhalting machine E that goes directly from the starting state to the halting state. Let $|\rho(E)| = C$. Then the input $\rho(E)\boldsymbol{s}$ to the UTM produces the output $\boldsymbol{s}$, so $K(\boldsymbol{s}) \leq |\boldsymbol{s}| + C$.
b) Consider the q-ary strings of length less than $n - c_1$. For every string of length n that has Kolmogorov complexity less than $n - c_1$, there is by definition at least one corresponding q-ary string of length less than $n - c_1$ associated with it. The number of programs of length less than $n - c_1$ cannot be greater than the total number of q-ary strings with less than this length, and that total is $(q^{n-c_1} - 1)/(q-1)$. Thus no more than $(q^{n-c_1} - 1)/(q-1)$ strings of length n can have such a low complexity.
c) Let L be an arbitrary natural number. Consider the following program that generates a string $\boldsymbol{s}_L$ of Kolmogorov complexity $K(\boldsymbol{s}_L) > L$ using the algorithm for computing the Kolmogorov complexity.

- Generate all strings lexicographically: $0, 1, \cdots, q-1, 00, \cdots$.
- Compute the Kolmogorov complexity of each.
- Stop when the first string $\boldsymbol{s}_L$ with complexity greater than L is found.
- Report $\boldsymbol{s}$ and halt.

The length of this program is $B + \log L$ for some constant $B = |\rho(M)|$. For large L, $B + \log L < L$, so for large L, the string can be computed by a valid program of length less than L, which contradicts the condition that $K(s) > L$.

d) Take n and λn and generate lexicographically all strings of length n with weight λn.
Specify which one of these is the string s. The program takes

$$\log_q \left\{ \binom{n}{\lambda n} (q-1)^{\lambda n} \right\} + O(\log_q n)$$

symbols. Using Stirling's formula for $n!$ we can derive

$$\log_q \left\{ \binom{n}{\lambda n} (q-1)^{\lambda n} \right\} = nH_q(\lambda) - O(\log_q n),$$

and so this quantity represents an upper bound on the complexity of any sequence of length n and weight λn, as claimed. □

Following Kolmogorov and Chaitin [6]–[9], we say that a string is *random* if its complexity is at least equal to its length. If this is so, there is no concise way of specifying the string—no procedure is much better than simply writing out all the symbols. A sequence which contains a pattern or obeys some law, on the other hand, can be expressed by a relatively short sequence of instructions to a universal Turing machine. The shorter the program, the less random is the string. The crucial point is that, as previously shown, *virtually all* strings of a given length are almost totally random (where the meaning of the terms "virtually all" and "almost totally" are obvious from property b)). By discussing the properties of sequences of high complexity, we are in effect discussing the properties of *typical* sequences, and it is this fact we will exploit later.

In speaking of the class of "low complexity" strings of a given length, we do not mean that the *complexity* of such strings is insignificant compared with that of the most random strings. Rather, we imply that the *number* of strings with such a low complexity is insignificant compared to the total number of strings. We also note that property c) implies that, in general, we can only acquire *upper* bounds on the Kolmogorov complexity of a string. Thus if a string of length n is shown to have a valid computing program of length $n - c_1$ for reasonably large c_1, we can definitely say that it has lower complexity than all but a tiny minority of strings. However, it may or may not have a complexity that is insignificant with respect to n.

Note also that in this formulation we are not concerned with the running time of the shortest program. If we wish to allow only programs that run efficiently, we can consider the *time-bounded* Kolmogorov complexity, defined to be the shortest program that will run on the universal Turing machine, halting within $\tau(n)$ steps, leaving the desired output string on the tape. The previous properties a), b), and d) still apply (the time bounded Kolmogorov complexity is computable if the function $\tau(n)$ is computable). Typically, the function $\tau(n)$ is set to some polynomial function of n. We will discuss this function later.

III. The Gilbert–Varshamov Bound

We being by discussing the basic statement of the bound, with some related facts. Our treatment follows that of MacWilliams and Sloane [10].

Theorem 2: There exists a linear (n, k) code over GF(q) with minimum distance at least d, where d is the greatest integer satisfying

$$1 + \binom{n-1}{1}(q-1) + \cdots + \binom{n-1}{d-2}(q-1)^{d-2} < q^{n-k}.$$

Proof: Given the $(n-k)\times n$ parity check matrix H with entries from GF(q), the code is defined as all those n-tuples $\boldsymbol{x}$ for which $H\boldsymbol{x}^T = 0$. If all combinations of $d-1$ or fewer columns of H are linearly independent, no nonzero vector of weight less than d can satisfy this, so the minimum distance is at least d. Thus it suffices to show that we can build an $(n-k)\times n$ parity check matrix with this property provided d is as given in the theorem. Suppose we have chosen i columns with the property that no combinations of $d-1$ or fewer of them are linearly dependent. There are at most $\sum_{j=0}^{d-2}\binom{i}{j}(q-1)^j$ distinct linear combinations of these i columns taken $d-2$ or fewer at a time. Provided this number is less than q^{n-k} we can add another column and still maintain the property that any $d-1$ or fewer columns of the new matrix are independent. We can keep adding columns as long as i is such that

$$1 + \binom{i}{1}(q-1) + \cdots + \binom{i}{d-2}(q-1)^{d-2} < q^{n-k},$$

i.e., as long as $i \le n-1$. □

Asymptotically we have

$$H_q(d/n) \ge 1 - R + o(1)$$

where

$$H_q(x) = x\log_q(q-1) - x\log_q x - (1-x)\log_q(1-x)$$

for $0 < x < 1$.

Many classes of codes have been shown to be asymptotically good in the sense that they meet the Gilbert–Varshamov bound. These include linear codes, alternant codes [11], generalized BCH codes [12], Goppa codes [13], double circulant (quasi-cyclic) codes [14], shortened cyclic codes [15], and self-dual codes [16].

In many cases, the following result [10] suffices to show that most members of a class of codes meet the Gilbert–Varshamov bound.

Theorem 3 [10]: Let $\boldsymbol{\Phi} = \{\Phi_1, \Phi_2, \cdots\}$ be an infinite family of linear codes over GF(q), where Φ_i is a set of (n_i, k_i) codes such that 1) $k_i/n_i > R$ and 2) each nonzero vector of length n_i belongs to the same number of codes in Φ_i. Then there are codes in this family which asymptotically meet the Gilbert–Varshamov bound.

Proof: Let N_0 be the total number of codes in Φ_i, and let N_1 be the number that contain a given nonzero vector. If we write out all the nonzero words in all codes in the class and count them in two different ways, we get $(q^k - 1)N_0 = (q^n - 1)N_1$. The number of nonzero vectors of weight d or less is $\sum_{i=1}^{d}\binom{n}{i}(q-1)^i$ and so the number of codes with minimum distance d or less is at most $N_1\sum_{i=1}^{d}\binom{n}{i}(q-1)^i$ and the fraction of codes for which this is true is at most

$$N_1 \sum_{i=1}^{d}\binom{n}{i}(q-1)^i \Big/ N_0 = \left[\sum_{i=1}^{d}\binom{n}{i}(q-1)^i\right] \cdot (q^k - 1)/(q^n - 1) = q^{n[H_q(d/n) - (1-R)] + o(n)},$$

and this fraction goes to zero exponentially as $n \to \infty$ if $H_q(d/n) < 1 - R$. □

For the case of linear systematic codes, we can modify this procedure to consider only sequences which are nonzero in the first k bits. We find that the fraction of such codes with minimum distance d or less is at most

$$\sum_{i=1}^{d}\left[\binom{n}{i}-\binom{n-k}{i}\right](q-1)^i q^{-n(1-R)} = q^{n[H_q(d/n)-(1-R)]+o(n)},$$

so again we conclude that for large n, *virtually all* linear systematic codes lie on or above the Gilbert–Varshamov bound.

It can also be shown [17] that virtually no linear codes over any symbol field lie *significantly above* the bound, in the sense that the fraction of codes with $H_q(d/n) \geq 1-R+\epsilon$ tends to zero for any $\epsilon > 0$. The bound is thus important as an indicator of the *average* behavior of codes. It is still not known if the bound is tight; in the binary case there exists a significant gap between the best upper bound on the asymptotic size of a code (the McEliece–Rodemich–Rumsey–Welch linear programming bound [18]) and the Gilbert–Varshamov bound. The resolution of this discrepancy is perhaps the most basic theoretical open problem in coding theory. It has long been conjectured that the Gilbert–Varshamov bound is in fact tight for binary codes (though it is now known to be loose for some larger symbol fields [19]).

To put all this in an information theory context, the bound represents a bound on the *reliability function* for the binary symmetric channel. Following Berlekamp [20], we let $P_e(N, M)$ denote the probability of error of the best code having M codewords of block length N for a given channel. For convenience, we define the rate in natural units as $R_e = (\ln M)/N$, and the reliability function as

$$E(R_e) = \lim_{N\to\infty} -\frac{1}{N}\ln P_e(N, [\exp R_e N])$$

assuming that the limit exists. (This is a special case of the more general reliability function $E(R_e, L)$ that takes advantage of list decoding, with $L = 1$.) We define the average guaranteed error correcting power as

$$e(R_e) = \lim_{n\to\infty} \frac{d(n, [\exp R_e n])}{2n}$$

where $d(n, M)$ is the greatest possible minimum distance in a binary code of length n with M codewords (as before, we assume the limit exists). Now $E(R_e)$ depends implicitly on the channel. For the binary symmetric channel with crossover probability p, we define the reliability function for the BSC, $E(R_e; p)$. Then we have [20]

$$e(R_e) = \lim_{p\to 0} \frac{-E(R_e; p)}{\ln p}.$$

As p goes to zero, the "expurgated" lower bound on $-E(R_e; p)/\ln p$ becomes the Gilbert–Varshamov bound on $e(R_e)$. The conjecture that the Gilbert–Varshamov bound is tight is equivalent to the conjecture that $E(R_e)$ coincides with the expurgated bound for the binary symmetric channel.

IV. Complexity and the Gilbert–Varshamov Bound

It is well known that every linear code is equivalent to a systematic code [21]. Instead of picking the entries of a generator matrix at random, we pick a systematic code according to the following rules: the generator matrix G is assumed to be of the form $(I|P)$, where I is the $k \times k$ identity matrix, and P is an arbitrary $k \times (n-k)$ matrix. We will call P the parity matrix.[1] There are $q^{k(n-k)}$ possible choices for P; each specifies exactly one code. Suppose we know R exactly. Then to specify the code, we can write out the parity matrix P row by row, to get a string of length $k(n-k) = n^2R(1-R)$ symbols. If R is known exactly, such a string can represent exactly one code. Thus the string specifies the code, and to specify the code it is necessary and sufficient that we specify the corresponding binary string.

Definition: The *defining string* of a linear systematic code $\mathscr{C}$ over GF(q) is the string

$$s_{\mathscr{C}} = \{\tilde{p}_1, \cdots, \tilde{p}_i, \cdots, \tilde{p}_k\}$$

where $\tilde{p}_i = \{p_{i,1}, \cdots, p_{i,j}, \cdots, p_{i,n-k}\}$, and where the generator matrix of the code has the form $G = (I|P)$.

In a slight abuse of notation, we say that the *Kolmogorov complexity of a code* is the Kolmogorov complexity of the string that defines the code.

Note that we have assumed a fixed R in this. There are two ways of dealing with this: either the fixed R is assumed to be known always, or we place some unambiguous encoding of the rate before the string of length $n^2R(1-R)$ to specify the code. The second method adds some fixed constant to the complexity; as we will see later, this is not important. We could also regard the length of the string as given, and speak of the *conditional Kolmogorov* complexity of the string. Once again, our conclusions would not be altered by adopting this convention.

Theorem 4: Let Φ be an infinite sequence of codes over some fixed symbol field GF(q) all of which have rate exactly R. Let the jth code have length n, minimum distance d, and defining strings s_j. Then there exists a constant C_0 such that

$$K(s_j) \leq C_0 + n^2R(1-R) - n\left[H_q(d/n) - (1-R)\right] + o(n)$$

for all j.

Proof: We outline a program for a universal Turing machine that calculates the defining string of the code. The parity matrix P consists of k rows, each containing $n-k$ symbols. We label the ith row of the generator matrix $\boldsymbol{r}_i$. The code has minimum distance d; suppose we are given any codeword of weight d. The first k symbols of this n-tuple represent an information sequence $\boldsymbol{i}$, while the last $n-k$ symbols represent the parity check sequence $\boldsymbol{p}$. Let the support of the information vector $\boldsymbol{i}$ be $\{\alpha_1, \cdots, \alpha_m\}$, where m is the number of nonzero symbols in the information sequence. Clearly, the specified codeword is of the form $\boldsymbol{c} = \sum_{j=1}^{m}\beta_j \boldsymbol{r}_{\alpha_j}$, where the β_j's are nonzero elements of GF(q). In other words, $\boldsymbol{c}$ is a sum of multiples of m rows of the generator matrix; which m rows these are can be determined from the first k symbols. Suppose we are given $\boldsymbol{c}$ (the codeword of weight d) and all the rows of the parity matrix except one—that one being the last row involved in the sum that represents $\boldsymbol{c}$ (i.e., row α_m in the notation previously given). Then it is a simple matter to recover $\boldsymbol{r}_{\alpha_m}$ by calculating $\boldsymbol{r}_{\alpha_m} = \beta_m^{-1}(\boldsymbol{c} - \sum_{j=1}^{m-1}\beta_j \boldsymbol{r}_{\alpha_j})$. Hence the defining string of the code is calculable from the given information. We have only to calculate the length of the program. We need an encoding of the Turing machine that performs the calculations, a specification of the low weight word, and specification of all rows of the parity matrix except one. The coding of the machine takes a constant number of symbols; the rows of the parity matrix take $(k-1)(n-k)$ symbols, and for the low weight word,

[1] We apologize for any confusion between the parity matrix P and the parity check matrix H. Also the distinction between the entropy function $H_q(x)$ and the parity check matrix H should be clear from context.

we give the value of d (taking $\lceil \log_q d \rceil$ symbols) and then say *which* word of weight d the low weight word is (taking $\left\lceil \log_q \left(\binom{n}{d} (q-1)^d \right) \right\rceil$ symbols). Using the relation $\binom{n}{\lambda n}(q-1)^{\lambda n} \le q^{nH_q(\lambda)+o(n)}$ we find that the program has length at most

$$C_1 + n^2 R(1-R) - n(1-R) + nH_q(d/n) + o(n)$$

where C_1 is the length of the encoding of the Turing machine, and this is then an upper bound for the Kolmogorov complexity of the code string, as claimed. □

Equivalently, we could use the fact that the parity check matrix H has the form $H = (-P^T | I)$ [21], where P^T is an $(n-k)\times k$ matrix and I is the $(n-k)\times(n-k)$ identity matrix. Given a codeword, we know that the corresponding columns of H sum to 0. We can thus omit one column of H, saving $n-k$ symbols, deriving it from the given codeword.

Corollary: Virtually all long systematic linear codes satisfy the Gilbert–Varshamov bound. More precisely, for any $\sigma > 0$, the fraction of systematic linear codes over GF(q) for which $H_q(d/n) \le 1 - R - \sigma$ is less than $q^{-n\sigma + o(n)}$ for all n.

Proof: From property b) of Kolmogorov complexity, the fraction of codes with complexity $n^2R(1-R) - n(1-R) + nH_q(d/n) + o(n)$ is less than

$$q^{n[H_q(d/n) - (1-R)] + o(n)}$$

and if $H_q(d/n) < 1 - R$, this fraction goes to zero exponentially with increasing n. □

The previous result also implies the following interesting and important observation, the main point of this correspondence.

Theorem 5: For any positive constant C_0 there exists a constant n_0 such that the following statement holds true: any linear code of block length $n > n_0$ and rate R over GF(q) that has Kolmogorov complexity no less than $n^2R(1-R) - C_0$ symbols must have minimum distance d satisfying $H_q(d/n) \ge 1 - R + o(1)$, i.e., must satisfy the Gilbert–Varshamov bound.

The term "random coding" is particularly apt: random selection is virtually certain to produce a good code; however, it is also virtually certain to produce a "random" code! The two classes turn out to be correlated. In the spirit of Wozencraft and Reiffen, we assert that "*any code which is sufficiently random is good.*"

Comparing our derivation to the standard one given in Section III, we see that we have effectively followed the same method: the pigeonhole principle guarantees that most codes are good. Now, however, we have simultaneously classified the codes according to complexity, and have found that it is precisely the set of high complexity codes that provides the guaranteed good codes.

The standard for a code being patternless is that a factor linear in the block length cannot be saved. This corresponds, for example, to saying that there is no row in the parity matrix, no diagonal, no column, which can be compressed down to any given percentage of its length. Our standard for "randomness" is thus in some ways quite generous: we allow the random selection of all but a linear number of bits, then insist only that the remaining linear number should allow compression to a percentage of its length. We need to ask about the converse of the previous result: given a code of low complexity, what is the probability that the code is bad?

We consider a modified random coding argument. We are given some *budget* B, a constant which is arbitrary but fixed, with which we are to design a procedure for constructing a code. A procedure is judged to be within budget if the shortest encoding $\rho(M)$ of a Turing machine M that will carry out the procedure has length no greater than the budget. To compensate for the extra overhead involved in specifying the Turing machine, the procedure must be able to save at least a linear amount of complexity in the specification of the code, i.e., we must be able to generate a code string of length $n^2R(1-R)$ given no more than $n^2R(1-R) - n\sigma$ symbols, for some constant σ; we will take σ to be at most $1-R$. We randomly select a procedure which obeys these rules (we avoid the halting problem by either randomly selecting from those programs that halt inside some given computation time or use an "oracle" to decide which programs halt leaving a valid codestring on the tape) and refer to the result as a random $C(B,\sigma)$ code. We have the following result.

Theorem 6: For sufficiently high budgets, random selection of a $C(B,\sigma)$ code results with probability $p > q^{-(B+1)} > 0$ in a code which has minimum distance d satisfying

$$H_q(d/n) \le 1 - R - \sigma + o(1)$$

regardless of the block length n.

Proof: The restrictions on the codes imply that each code string has complexity $K(s) \le B + n^2R(1-R) - n\sigma$ symbols. The number of codes with this complexity is certainly less than $q^{B + n^2R(1-R) - n\sigma}$ by the pigeonhole principle. We now count the number of bad codes that have sufficiently low complexity. For a given n, pick d^* to be the largest integer such that

$$\sum_{i=1}^{d^*} \binom{n}{i} (q-1)^i \le q^{n[1-R-\sigma]}.$$

From our previous arguments, a code with a minimum distance $\le d^*$ can be calculated by giving an encoding of a Turing machine M (containing the values of both R and σ), the specification of the lowest weight word (taking $\log_q \sum_{i=1}^{d^*} \binom{n}{i} (q-1)^i$ symbols) and the remaining words of the parity matrix P. (Note that the values of n and d^* are implicit from the length of the input and the values of R and σ). The program thus takes

$$C_0 + \log_q \sum_{i=1}^{d^*} \left\{ \binom{n}{i} (q-1)^i \right\} + n^2R(1-R) - n(1-R)$$
$$\le C_0 + n^2R(1-R) - n\sigma \text{ symbols.}$$

We take the "sufficiently large" condition on the budget B to mean that $B \ge C_0$. Now every code with minimum distance less than d^* is representable by a program meeting the requirements set out in the statement of the theorem. The familiar argument given in Section III yields an *upper* bound on the number of codes with distance less than the Gilbert–Varshamov bound. We seek a *lower* bound on the number of such codes.

Let $N_i(n, d^*)$ be the number of "valid" distinct choices of i distinct nonzero n-tuples, each of which has weight d^* or less and a leading nonzero coefficient of 1, where a choice is valid if there is any systematic linear code which contains that set of codewords. Let $S_i(n)$ be the number of systematic linear codes containing a given valid set of i nonzero codewords, averaged over all valid sets of i nonzero codewords. Let $T(n, d^*)$ be the number of codes with minimum distance d^* or less. By the

principle of inclusion and exclusion, we have

$$T(n,d^*) = N_1(n,d^*)S_1(n) - N_2(n,d^*)S_2(n) + N_3(n,d^*)S_3(n) - \cdots$$

where the sum is overestimated by taking an odd number of terms and underestimated by taking an even number of terms. We need only N_1, N_2, S_1, and S_2.

We have

$$N_1(n,d^*) = \sum_{j=1}^{d^*} \left[\binom{n}{j} - \binom{n-k}{j} \right] (q-1)^{j-1},$$

and

$$N_2(n,d^*) = \left(\sum_{j=1}^{d^*} \left(\binom{n}{j} - \binom{n-k}{j} \right) (q-1)^{j-1} \right) - E_2(n,d^*),$$

where $E_2(n,d^*)$ represents the number of unordered pairs of nonzero codewords each of weight $\le d^*$ that have the same nonzero sequence in their first k symbols. We also have

$$S_1(n) = q^{(k-1)(n-k)} \quad \text{and} \quad S_2(n) = q^{(k-2)(n-k)}.$$

Thus

$$\begin{aligned} T(n,d^*) &\ge N_1(n,d^*)S_1(n) - N_2(n,d^*)S_2(n) \\ &\ge \sum_{i=1}^{d^*} \left[\left\{ \binom{n}{i} - \binom{n-k}{i} \right\} (q-1)^i \right] q^{(k-1)(n-k)} \\ &\quad - \frac{1}{2} \left[\sum_{i=1}^{d^*} \left\{ \binom{n}{i} - \binom{n-k}{i} \right\} (q-1)^i \right]^2 q^{(k-2)(n-k)} \\ &\quad + \frac{1}{2} \sum_{i=1}^{d^*} \left[\left\{ \binom{n}{i} - \binom{n-k}{i} \right\} (q-1)^i \right] q^{(k-2)(n-k)}. \end{aligned}$$

Noting that $\binom{n-k}{d} / \binom{n}{d} \le (1-R)^d$, we have

$$\begin{aligned} T(n,d^*) &\ge q^{n^2R(1-R)-n\sigma}(1+o(1)) \\ &> q^{n^2R(1-R)-n\sigma-1} \end{aligned}$$

for sufficiently large n. Thus picking a $C(B,\sigma)$ code at random gives a probability of at least q^{-B-1} of picking a bad code. □

Of course, there are codes of low Kolmogorov complexity that do meet the Gilbert–Varshamov bound. The simplest example is the code produced by the following program: for a given n and k, generate all codes lexicographically; for each code, determine the minimum distance; stop when we find the first code that meets the bound, report that code and halt. The problem, however, is that the running time of this program is exponential in the block length. There are more efficient algorithms [2] based on the same idea, but none with subexponential running time. Our result shows that even with unconstrained running time, random selection of a low complexity code has a certain minimum likelihood of producing a bad code. Moreover, by considering the *time-bounded* Kolmogorov complexity mentioned earlier, and setting $\tau(n)$ equal to some polynomial function n^3 or greater, we have the same lower bound on the probability of selecting a bad code (because our described procedure for bad codes has running time which is $o(n^3)$) while we have now no reason to believe that the *upper* bound for this probability is less than one.

It is possible to reverse our argument, and thus to derive many of the same conclusions in a different way. From Section III, we know that the fraction of linear codes over GF(q) with $H_q(d/n) \le 1-R-\sigma$ is at most $q^{-n\sigma+o(n)}$. Thus it is possible to write a program that indicates that the string to be specified represents a bad code, and then say *which* of the strings representing bad codes is the one to be specified. As there are no more than $q^{n^2R(1-R)-n\sigma+o(n)}$ systematic linear codes over GF(q) satisfying $H_q(d/n) \le 1-R-\sigma$, the Kolmogorov complexity of the defining string of any such code is upper-bounded by

$$K(\mathscr{C}) \le n^2R(1-R) - n\sigma + o(n).$$

Thus, again, any code that is effectively random must satisfy the Gilbert–Varshamov bound. Conversely, assume that the fraction of codes with $H_q(d/n) \le 1-R-\sigma$ is exactly $q^{-g(n,R,\sigma)}$ where $g(n,R,\sigma)$ is defined appropriately. Any such bad code can then be represented by a program of length $\le C_0 + n^2R(1-R) - \lceil g(n,R,\sigma) \rceil$ symbols, where C_0 is the length of the formal description of the Turing machine. Then suppose we have a budget $B > C_0$ as before, and that we select codes at random from the set of codes with Kolmogorov complexity at most $B + n^2R(1-R) - \lceil g(n,R,\sigma) \rceil$ symbols. There are at most $q^{B+n^2R(1-R)-\lceil g(n,R,\sigma) \rceil}$ strings of the required complexity, and there are exactly $q^{n^2R(1-R)-g(n,R,\sigma)}$ bad codes among them. Thus the probability of picking a bad code is at least $q^{-(B+1)}$ as before.

Although this argument is briefer than the one already given, it obscures some points we want to make. First, we wish to show that many of the main characteristics of a class of codes can be derived in a simple and intuitive way from the consideration of the Kolmogorov complexity of the defining strings of the codes (where the defining strings are defined in a way appropriate for the class). We illustrate the point in Section VI, and in [26], we use the same idea to analyze the complexity of a decoding procedure for general linear codes. Second, we wish to show that in the random selection from relatively low complexity codes, the "badness" of a fraction of the resulting codes is proportional to the amount of complexity we save, in the sense given in Theorem 6. Third, we recall that our main concern lies in discovering the polynomial-time-bounded Kolmogorov complexity of codes. Comparing the two arguments given, we note that in the first case, our Turing machine program runs in $o(n^3)$ time, whereas in the second case, there is no reason to believe that the program runs in polynomial time. This means that in Theorems 4–6, we can substitute "polynomial-time-bounded" Kolmogorov complexity for unrestricted Kolmogorov complexity without altering the validity of the results.

It may appear that our suggested program for calculating the code string of a bad code is quite a loose upper bound, but in fact this is not so: our upper bound for the complexity of a bad code is tight for virtually all such codes. By an elementary application of the pigeonhole principle, we see that the fraction of codes for which $K(s) < K_u(s) - (C_0+1) - C_1$ is less than q^{-C_1} for any constant C_1, where $K_u(s)$ is the upper bound and the constant C_0 is the length of the encoding of the Turing machine M already described.

The following result is now obvious: most codes that have any nonzero vector of weight less than the Gilbert–Varshamov bound have exactly one such vector. This follows from the way $T(n,d^*)$ is derived, but also because a code with *two* words of weight less than $nH_q^{-1}(1-R-\sigma)$ for any $\sigma > 0$ can be represented by a program of length

$$\begin{aligned} &\le n^2R(1-R) - 2n(1-R) + 2nH_q(d/n) + o(n) \\ &= n^2R(1-R) - 2n\sigma + o(n) \end{aligned}$$

and is thus of significantly lower complexity than even the average code that does not meet the bound.

It is also clear from the discussion of Theorem 4 that the *radius* r of virtually all linear codes over GF(q) satisfies $H_q(r/n) \geq 1 - R + o(1)$, where the radius of a code is defined [10] as the *maximum* distance between any two codewords.

V. Nonlinear Codes

In the case of nonlinear codes, we find that a new formulation is needed to represent the codes. The lack of structure also manifests itself in the much higher Kolmogorov complexity of most of these codes. Once again, when we speak of "high-" or "low-complexity" codes, we are implicitly using these terms in a relative way.

We have the following argument. An $[n, M]$ nonlinear code over GF(q) is a collection of M distinct q-ary n-tuples. Let the function $f(n, M)$ be such that the number of $[n, M]$ nonlinear codes over GF(q) is $q^{f(n,M)}$. Then $\lceil f(n, M) \rceil$ q-ary symbols is the minimum amount required to be able to specify any of the codes. The complexity of the code is defined to be the complexity of the string of $\lceil f(n, M) \rceil$ symbols which specifies it. Let d be the minimum distance of the code $\mathscr{C}$. Then there are two codewords $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ which are such that $\mathrm{wt}(\boldsymbol{w}_1 - \boldsymbol{w}_2) = d$. We can specify the code using the following procedure: specify the $[n, M-1]$ code obtained by deleting $\boldsymbol{w}_2$ from $\mathscr{C}$, specify which word in the expurgated code is $\boldsymbol{w}_1$, give the n-tuple $\boldsymbol{w}_1 + \boldsymbol{w}_2$ of weight d, and reconstruct $\boldsymbol{w}_2$ from that information. We need $\lceil f(n, M-1) \rceil$ symbols to specify the expurgated code, $\lceil \log_q M \rceil$ symbols to specify which codeword is $\boldsymbol{w}_1$, and at most $nH_q(d/n) + o(n)$ symbols to give the n-tuple of weight d. This must be close to $f(n, M)$. So

$$nH_q(d/n) + nR + o(n) \geq f(n, M) - f(n, M-1)$$

for most codes. Now $f(n, M) = \log_q \binom{q^n}{M}$, so

$$\begin{aligned} f(n, M) - f(n, M-1) &= \log_q \frac{\binom{q^n}{M}}{\binom{q^n}{M-1}} \\ &= \log_q ((q^n - M + 1)/M) \\ &= n(1-R)(1 + o(1)), \end{aligned}$$

and we find that

$$H_q(d/n) \geq 1 - 2R + o(1)$$

for most nonlinear codes.

Alternatively, we can specify the code by writing out each codeword in turn to get a string of length nM symbols. If we view the *order* in which we place the codewords as significant, any string of length nM symbols represents a nonlinear code with at most M codewords. If code $\mathscr{C}$ has minimum distance d, there are two codewords $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ that are separated by an n-tuple of weight d. To compute the code string, it is sufficient to do the following: specify the code string corresponding to the code $\mathscr{C}_1$ obtained by removing word $\boldsymbol{c}_2$ from $\mathscr{C}$. Specify which word in this subcode is $\boldsymbol{c}_1$, give the n-tuple of weight d, which is $\boldsymbol{c}_1 - \boldsymbol{c}_2$, then say *where* $\boldsymbol{c}_2$ is located in the code string for $\mathscr{C}$. We have

$$n(M-1) + k + nH_q(d/n) + k + o(n) \geq nM$$

for most codes (where $k = nR = \log_q M$), or $H_q(d/n) \geq 1 - 2R + o(1)$ as before.

In contrast to the situation for linear codes, the Gilbert–Varshamov bound does not seem to be automatically met by high complexity codes—indeed, for $R > 1/2$, we have no guarantee of distance at all.

Of course, we have merely derived a lower bound on distance for most codes, and we should ask how tight this bound is. This question can be interpreted two ways: 1) are there really binary nonlinear codes that are almost totally random that have $H_q(d/n) \approx 1 - 2R$ for $R < 1/2$ and $d \to 0$ for $R \geq 1/2$? 2) is the average behavior of nonlinear codes really below the Gilbert–Varshamov bound? The answer to both questions is yes.

For the first question, we apply the pigeonhole principle again. Assume that this is false for all high complexity $[n, M-1]$ codes, i.e., $H_q(d/n) \geq 1 - 2R + \epsilon$ for $R < 1/2$, and $H_q(d/n) \geq \epsilon$ for $R \geq 1/2$, for some $\epsilon > 0$.

We take a high complexity nonlinear $[n, M-1]$ code over GF(q), a word from that code, and another n-tuple at distance $d^* \leq d$ from the selected codeword. The total number of results is

$$q^{f(n, M-1) - C} q^{nR} q^{nH_q(d^*/n) + C_1 \log_q n}.$$

Each result can arise in at most two ways from the above constructions, because $d^* \leq d$, so the total number of $[n, M]$ codes we get is

$$q^{f(n,M) - n(1-R) + nR + nH_q(d^*/n) + o(n)}.$$

If $R \geq 1/2$, we find that the number of distinct $[n, M]$ codes over GF(q) is greater than $q^{f(n,M) + n(2R-1)}$, contradicting the definition of the function $f(n, M)$. Similarly, for $R < 1/2$, if $H_q(d/n) = 1 - 2R + \epsilon$ for all high complexity codes, we would find that the number of $[n, M]$ codes over GF(q) is greater than $q^{f(n,M) + n\epsilon + o(n)}$, a contradiction for positive ϵ.

We should now suspect that this bound is tight for a significant fraction of nonlinear codes. Indeed, suppose that the fraction of codes over GF(q) for which this is tight is $\alpha(n, R)$. Then, as Martin-Löf has pointed out [27], we can save at least $\log_q \alpha(n, R) + O(1)$ symbols in the specification of any code for which the bound is tight. If $\alpha(n, R)$ tends to zero with increasing n, then the bound cannot be tight for any random code, contradicting the aforementioned 1). Thus we conclude that the bound is tight for a fraction of nonlinear codes that is bounded away from zero.

It is possible to obtain the following stronger result.

Theorem 7: The fraction of nonlinear $[n, q^{nR}]$ codes over GF(q) satisfying

$$H_q(d/n) \geq \max(1 - 2R, 0) + \alpha$$

for any $\alpha > 0$ is less than $q^{-\alpha}$.

Proof: First note that if this is true for $R = 1/2$, it is trivially true for $R > 1/2$. So we take $R \leq 1/2$. Select M codewords at random from all q^n possible sequences, and let $\boldsymbol{X}$ be a random variable denoting the number of unordered pairs of codewords at distance $\leq d$ from each other. We can find $E(\boldsymbol{X})$ and $E(\boldsymbol{X}^2)$ and hence can bound $\Pr(\boldsymbol{X} = 0)$ using Chebyshev's inequality.

Let $\boldsymbol{X}_{ij}$ be a random variable that takes the value 1 if the ith and jth codewords are at distance $\leq d$ from each other, and 0,

otherwise. Then $X = \sum_{i<j} X_{ij}$, and so [28]

$$\mu = E(X) = \sum_{i<j} E(X_{ij}) = \binom{M}{2} \frac{\sum_{l=1}^{d} \binom{n}{l}(q-1)^l}{q^n - 1}.$$

Also

$$E(X^2) = \sum_{\substack{i,j,k,l \\ i<j \\ k<l}} X_{ij}X_{kl}.$$

We find that

$$E(X^2) = \binom{M}{2}\binom{M-2}{2}\left(\frac{\sum_{l=1}^{d}\binom{n}{l}(q-1)^l}{q^n-1}\right)^2 (1+O(q^{-n}))$$
$$+6\binom{M}{3}\left(\frac{\sum_{l=1}^{d}\binom{n}{l}(q-1)^l}{q^n-1}\right)^2 (1+O(q^{-n}))$$
$$+\binom{M}{2}\frac{\sum_{l=1}^{d}\binom{n}{l}(q-1)^l}{q^n-1}$$

where the first term represents the products $X_{ij}X_{kl}$ where i, j, k, and l are all different, the second term represents the products where $\#(i,j,k,l) = 3$, and the final term represents products where $(i,j) = (k,l)$. Then

$$\sigma^2(X) = E(X^2) - E^2(X)$$
$$= \frac{1}{2}M^2 p(1 + O(M^{-1}) + O(p)) + p^2 O(q^{-n}) O(M^4)$$
$$= \frac{1}{2}M^2 p(1 + o(1))$$

where $p = \sum_{l=1}^{d}\binom{n}{l}(q-1)^l/(q^n-1)$. Thus $\sigma^2(X) = O(E(X))$. Chebyshev's inequality then gives

$$\Pr(X = 0) \le \frac{\sigma^2(X)}{\mu^2(X)} = O(E^{-1}(X)).$$

Finally,

$$E(X) = \binom{M}{2}\frac{\sum_{i=1}^{d}\binom{n}{i}(q-1)^i}{q^n-1}$$
$$= q^{n[H_q(d/n)-(1-2R)]+o(n)}$$

and the theorem follows. □

We conclude by showing that the probability that a randomly selected code will be significantly *better* than the Gilbert–Varshamov bound (i.e., $H_q(d/n) = 1 - R + \sigma$ for some $\sigma > 0$) is upper bounded by a function that goes to zero as a double exponential. A necessary condition for the code to have minimum distance d is that any given codeword should have no other codeword within distance $d-1$. We pick an initial codeword at random, and then complete the code by selecting the other $M-1$ codewords. The event that there is no codeword within distance d of the first codeword will have probability

$$\le \left(1 - q^{-n(1-H_q(d/n))+o(n)}\right)^{q^{nR}}$$
$$= \left[\left(1 - q^{-n(1-H_q(d/n))+o(n)}\right)^{q^{n(1-H_q(d/n))+o(n)}}\right]^{q^{nR+o(n)}}$$
$$\to e^{-q^{nR}}$$

as n becomes large.

VI. Other Classes of Codes

We feel that the ideas in Kolmogorov complexity provide a useful and intuitively appealing tool for analyzing various properties of codes. By writing a Turing machine program to calculate the defining string of a code, and by observing that the length of this program cannot be significantly less than the logarithm of the number of codes in the class in most cases, we obtain a simple inequality yielding the typical behavior of the class of codes. We restrict ourselves here to two examples to illustrate the idea. Alternative proofs of the properties below can be found elsewhere [29], [30].

Consider the class of *shortened cyclic* codes over GF(2). An (n,k) shortened cyclic code is defined by a generator polynomial $g(x)$ of degree $n-k$, and the code consists of all n-tuples c that in polynomial representation are of the form $i(x)g(x)$, where $\deg i(x) < k$. If we assume that $g(x)$ is a monic polynomial, the code is uniquely representable (given n and R) by the string $(g_{n-k-1}, \cdots, g_0)$. This is a binary string of length $n(1-R)$; virtually all such binary strings have Kolmogorov complexity close to $n(1-R)$ bits. Suppose the code has minimum distance d. Then we can specify $g(x)$ by giving a codeword of weight d, and then specifying which factor of the codeword is $g(x)$. Piret has shown [29] that a polynomial of degree n over GF(2) can have at most $2^{(n/\log_2 n)(1+o(1))}$ distinct factors, so we need $O(n/\log n)$ bits to specify the generator given a codeword. Overall, to specify the value of d, the codeword of weight d, and the particular factor of the codeword, requires a program of length $nH_2(d/n) + O(n/\log n)$ bits. Now the Kolmogorov complexity of a shortened cyclic code is at least $n(1-R) - C$ for all but a fraction of at most 2^{-C} of all such codes. Thus the fraction of codes for which $H_2(d/n) \le 1 - R - \sigma + o(1)$ for $\sigma > 0$ is less than $2^{-n\sigma + o(n)}$ for all n. Therefore virtually all shortened cyclic codes meet the Gilbert–Varshamov bound.

Note that modified forms of Theorems 4 and 5 apply here also. Bad codes (with $H_2(d/n) \le 1 - R - \sigma$ for $\sigma > 0$) have Kolmogorov complexity at most $n(1 - R - \sigma + o(1))$ bits; conversely, random selection from codes with Kolmogorov complexity at most $n(1 - R - \sigma + o(1))$ bits results with nonzero probability in a code with $H_2(d/n) \le 1 - R - \sigma$.

Now consider the class of linear concatenated codes with Reed–Solomon outer codes and varying *nonsystematic* inner codes. We show that there are codes in this class that meet the Gilbert–Varshamov bound. Let the outer code have block length N and rate R, and let the inner codes have rate $r = 1$. Thus to encode, we form the Reed–Solomon codeword in the usual manner to get $(c_0, \cdots, c_{N-1})$ where the c_j's are elements from GF(2^n), and then apply a "template" $(r_0, \cdots, r_{N-1})$ where the r_j's are also from GF(2^n) to get the resulting codeword $(c_0 r_0, \cdots, c_{N-1} r_{N-1})$. Finally, we interpret each symbol from GF(2^n) as a string of n bits. Clearly, the code has length Nn and rate R. Our choice of template decides the code. We show that virtually any choice yields a code meeting the Gilbert–Varshamov bound.

Clearly, given n and N, the template can be represented by a string of length Nn bits, and any string of this length represents

exactly one template. Most such templates have Kolmogorov complexity close to Nn bits. Now if we are given a codeword of weight d, and the first NnR bits of the template, we can recover the remaining bits: we are given $(c_0 r_0, \cdots, c_{N-1} r_{N-1})$ and $(r_0, \cdots, r_{NR-1})$, from which we calculate $(c_0, \cdots, c_{NR-1})$, then $(c_{NR}, \cdots, c_{N-1})$ and finally $(r_{NR}, \cdots, r_{N-1})$. Thus by the familiar argument, $NnH_2(d/Nn) + NnR + o(Nn) \geq Nn - C$ for all but a fraction of at most 2^{-C} of all such codes, i.e., $H_2(d/Nn) \geq 1 - R - o(1)$ for most such codes. Once again, suitably modified version of Theorems 4 and 5 apply.

VII. Conclusion

We have seen that the fact that most linear codes meet the Gilbert–Varshamov bound is a consequence of the fact that most of these codes are effectively random. Thus the common complaint given in the title of the paper is no mere accident, but a fundamental principle of coding theory. We have also demonstrated that a converse holds: codes that are not effectively random have a certain nonzero probability of lying below the Gilbert–Varshamov bound. Furthermore, in a certain sense, the less random the code is, the further away from the bound it is likely to be.

Sometimes it may be desirable to regard a code as random unless it can be recovered from a significantly compressed specification in polynomial time; even with this interpretation, the previous results hold.

The behavior of nonlinear codes contrasts sharply with that of the linear codes, and the statement of Wozencraft and Reiffen cannot be said to apply to them. In both cases, the behavior of any effectively random string is used to bound the distance of "most" codes in the class. In the case of linear codes, the bound obtained happens to coincide with the best known lower bound for the best codes.

By characterizing the average behavior of a class of codes in terms of the properties of the most random ("typical") codes in the class, we have introduced a novel and intuitively appealing way of analyzing average properties of codes. The approach can be used to derive results on classes other than general linear and nonlinear codes; we intend to report these results in a future communication.

Acknowledgment

The authors wish to thank the anonymous reviewers who made extensive comments that helped improve the exposition. In particular, we wish to thank one of the reviewers who suggested the proof of Theorem 7.

References

[1] E. N. Gilbert, "A comparison of signalling alphabets," *Bell Syst. Tech. J.*, vol. 31, pp. 504–522, 1952.
[2] R. R. Varshamov, "Estimate of the number of signals in error correcting codes," *Dokl. Akad. Nauk. SSSR*, vol. 117, pp. 739–741, 1957.
[3] J. M. Wozencraft and B. Reiffen, *Sequential Decoding*. Cambridge, MA: MIT Press, 1961.
[4] M. Davis, Ed., *The Undecidable*. Hewlett, N.Y.: Raven Press, 1965.
[5] M. Davis, *Computability and Unsolvability*. New York: McGraw-Hill, 1958.
[6] A. N. Kolmogorov, "Three approaches to the definition of the concept 'quantity of information,'" *Probl. Peredach. Inform.*, vol. 1, pp. 3–11, 1965.
[7] G. J. Chaitin, "On the length of programs for computing finite binary sequences," *J. ACM*, vol. 13, pp. 547–569, 1966.
[8] A. N. Kolmogorov, "Logical basis for information theory and probability theory," *IEEE Trans. Inform. Theory*, vol. IT-14, pp. 662–664, 1968.
[9] G. J. Chaitin, "On the difficulty of computations," *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 5–9, 1970.
[10] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam: North-Holland, 1977.
[11] H. J. Helgert, "Alternant codes," *Inform. Contr.*, vol. 26, pp. 369–380, 1974.
[12] R. T. Chien and D. M. Choy, "Algebraic generalization of BCH–Goppa–Helgert codes," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 70–79, 1975.
[13] V. D. Goppa, "A new class of linear error-correcting codes," *Probl. Peredach. Inform.*, vol. 6, pp. 207–212, 1970.
[14] T. Kasami, "A Gilbert–Varshamov bound for quasi-cyclic codes of rate 1/2," *IEEE Trans. Inform. Theory*, vol. IT-20, p. 679, 1974.
[15] T. Kasami, "An upper bound on k/n for affine invariant codes with fixed d/n," *IEEE Trans. Inform. Theory*, vol. IT-15, pp. 174–176, 1969.
[16] F. J. MacWilliams, N. J. A. Sloane, and J. G. Thompson, "Good self–dual codes exist," *Discrete Math.*, vol. 3, pp. 153–162, 1972.
[17] V. M. Blinovskii, "Lower asymptotic bound on the number of linear code words in a sphere of given radius in F_q^n," *Probl. Peredach. Inform.*, vol. 23, pp. 50–53, 1987.
[18] R. J. McEliece, E. R. Rodemich, H. C. Rumsey, Jr., and L. R. Welch, "New upper bounds on the rate of a code via the Delsarte–MacWilliams inequalities," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 157–166, 1977.
[19] M. A. Tsfasman, S. G. Vladuts, and T. Zink, "Modular curves, Shimura curves, and Goppa codes better than the Varshamov–Gilbert bound," *Mathematische Nachrichten*, vol. 104, pp. 13–28, 1982.
[20] E. R. Berlekamp, *Algebraic Coding Theory*. New York: McGraw-Hill, 1968.
[21] R. J. McEliece, *The Theory of Information and Coding*. Reading, MA: Addison-Wesley, 1977.
[22] L. A. Bassalygo, "Formalization of the problem of the complexity of code specification," *Probl. Peredach. Inform.*, vol. 12, pp. 105–106, 1976.
[23] L. A. Bassalygo, V. V. Zyablov, and M. S. Pinsker, "Problems of complexity in the theory of correcting codes," *Probl. Peredach. Inform.*, vol. 13, pp. 5–17, 1977.
[24] V. Yu. Krachkovskii, "Complexity of constructing codes with specified correction properties," *Probl. Peredach. Inform.*, vol. 15, pp. 50–55, 1979.
[25] V. V. Zyablov, "An estimate of the complexity of constructing binary linear cascade codes," *Probl. Peredach. Inform.*, vol. 7, pp. 5–13, 1971.
[26] J. T. Coffey and R. M. F. Goodman, "The complexity of information set decoding," *IEEE Trans. Inform. Theory*, vol. 36, no. 5, pp. 1031–1037, Sept. 1990.
[27] P. Martin-Löf, "The definition of random sequences," *Inform. Contr.*, vol. 9, pp. 602–619, 1966.
[28] W. Feller, *An Introduction to Probability Theory and Its Applications*, vol. 1. New York: Wiley, 1965.
[29] P. Piret, "On the number of divisors of a polynomial over GF(2)," *Second Int. Conf. Appl. Algebra, Algorithmics and Error-Correcting Codes*, Toulouse, France, Oct. 1984.
[30] P. Delsarte, "On subfield subcodes of modified Reed–Solomon codes," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 575–576, 1975.

On the Decoding of Algebraic–Geometric Codes Over $\mathbb{F}_q$ for $q \geq 16$

SERGE VLĂDUŢ

***Abstract*—It is proved that for algebraic–geometric codes on a curve over $\mathbb{F}_q$ for $q \geq 37$ or on a curve of sufficiently large genus over $\mathbb{F}_q$ for $q \geq 16$ there exists a polynomial decoding algorithm up to $(d^* - 1)/2$ errors, d^* being the designed minimum distance.**

I. Introduction

Algebraic–geometric (or AG-) codes were discovered by Goppa [1]. It was found that they have many remarkable properties, many of them described in [2]. Decoding AG-codes has been on open problem for some time. Justesen *et al.* [3] proposed an algorithm for the decoding of AG-codes that was generalized in [5]. This algorithm decodes an AG-code, roughly

Manuscript received June 16, 1989; revised January 12, 1990.
The author is with CEMI, Krasikova, 32, Moscow, USSR.
IEEE Log Number 9036818.