ON COMPLEXITY AND EFFICIENCY IN ENCODING AND DECODING ERROR-CORRECTING CODES

Thesis by

John Timothy Coffey

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1989

(Submitted May 18, 1989)

Acknowledgements

I wish to thank my advisor, Prof. Rod Goodman, for his encouragement and generous support throughout my stay at Caltech. He provided an excellent environment for research and allowed me the independence to pursue various (and sometimes tangential) topics. Sincere thanks must also go to Prof. Paddy Farrell of the University of Manchester for many creative ideas, and for his hospitality in inviting me to visit Manchester in the summer of 1987. I wish to thank Prof. Yaser Abu-Mostafa, Prof. Bob McEliece and Prof. Ed Posner for their great help during my stay at Caltech; I was also fortunate to get to know Dr. Gus Solomon of Hughes Aircraft, who provided much interest and advice. All the above are also due thanks as members of my thesis committee.

To my many friends and relatives in Los Angeles, who made my four years at Caltech so enjoyable: thank you!

Most of all, thanks to my father, who first suggested that I should do a postgraduate degree, and to my mother.

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 which yields such codes over arbitrary symbol fields. A quarter of a century ago, Wozencraft & 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 this statement holds true in a rigorous mathematical sense: any linear code which is truly random, in the sense that there is no concise way of specifying the code, is good. Furthermore, random selection of a code which does contain some constructive pattern results, with probability bounded away from zero, in a code which does not meet the Gilbert-Varshamov bound *regardless of the block length of the code.* In contrast to the situation for linear codes, we show that there are effectively random non-linear codes which have no guarantee on distance, and that over all rates, the average non-linear code has much lower distance than the average linear code.

These techniques are used to derive original results on the performance of various classes of codes, including shortened cyclic, generalized Reed-Solomon, and general non-linear codes, under a variety of decoding strategies involving mixed burst- and random-error correction.

The second part of the thesis deals with the problem of finding decoding algorithms for general linear codes. These algorithms are capable of full hard decision decoding or bounded soft decision decoding, and do not rely on any rare structure for their effectiveness.

After a brief discussion of some aspects of the theory of NP-completeness as it relates to coding theory, we propose a simple model of a general decoding algorithm which is sufficiently powerful to be able to describe most of the known approaches to the problem. We provide asymptotic analysis of the complexity of various approaches to the problem under various decoding strategies (full hard decision decoding and bounded hard- and soft-decision decoding) and show that a generalization of information set decoding gives more efficient algorithms than any other approach known.

Finally, we propose a new type of algorithm that synthesizes some of the advantages of information set decoding and other algorithms that exploit the weight structure of the code, such as the zero neighbours algorithm, and discuss its effectiveness.

Contents

List of Figures

List of Tables

To my father and mother.

Chapter 1

Kolmogorov Complexity in Coding Theory

1.1 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 with increasing blocklength [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 which 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 & 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 chapter, 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. Sometimes a code may contain a 'pattern' that cannot be exploited in constructing the code; to deal with this, we discuss the *time-bounded* Kolmogorov complexity. We show that any code that is random in the wider sense of having no efficiently computable pattern must meet the Gilbert-Varshamov bound.

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 which 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 also hold when we add the condition that the code be 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 good 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 analysing the typical behaviour of classes of codes.

In Section 1.2, we discuss the basic axioms of complexity theory. In Section 1.3, we outline the conventional proofs of the Gilbert-Varshamov bound and discuss some basic related results. In Section 1.4, 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 1.5, we discuss the case of general non-linear codes, and show that the result does not apply to this class: there are effectively random non-linear codes which have no guarantee on distance. In Section 1.6, we show how some aspects of the typical behaviour of classes of codes can be derived from a characterization of codes in terms of their complexity. In Section 1.7, we discuss and contrast corresponding results for different error-correction strategies, such as burst-error and combined random- and burst-error correction.

Summary of Original Contributions

The main results, those of Section 1.4, are entirely original. The idea of using Kolmogorov complexity to analyse the average properties of various classes of codes is also original. Various results exist for the average properties of codes using other techniques (these results have been cited where appropriate); however, the results here have been derived independently, and many are new: combined burst- and random-error correction for linear codes, shortened cyclic codes and generalized Reed-Solomon codes, and burst- and random-error correction for the average non-linear code.

1.2 Kolmogorov Complexity

The discussion here is based on material from [105], which should be consulted for further details. 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 $\Sigma = \{\sigma_1, \ldots, \sigma_K\}$ 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 $\{q_0, q_1, \ldots, q_P\}$, 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 which 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

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

We characterize the current status of a Turing Machine by its 'instantaneous description.'

Definition 1.1 *The instantaneous description (ID) of a Turing Machine is a quadruple* $(q_{p_1}, \mathbf{u}, \sigma_{k_1}, \mathbf{v})$.

Informally, the machine is in state q_{p_1}, the symbol under the read-write head is σ_{k_1}, the string **u** is on the tape to the left of the read-write head, and the string **v** is to the right of the read-write head. (The string **u** is taken to begin at the leftmost non-blank symbol on the tape; the string **v** is taken to end at the rightmost non-blank 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

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

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 non-mathematical concept of 'computability,' it is virtually universally accepted as a *de facto* definition of computability.

Definition 1.2 *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* $\Sigma_1 \cup \Sigma_2 \subseteq \{\sigma_0, \ldots, \sigma_K\}$ *and for every* $\mathbf{x} \in \Sigma_1^*$, *we have* $(q_0, \lambda, \sigma_0, \mathbf{x}) \rightarrow_M^* (q_p, \lambda, \sigma_0, \mathbf{y})$ *where* $\mathbf{y} = f(\mathbf{x})$ *and* λ *is the null string.*

A *Universal Turing Machine* (UTM) is, informally, a general purpose Turing Machine. It takes as input a string $\mathbf{x} = \rho(M)\rho(\mathbf{w})$ where $\rho(M)$ is an encoding of a

Turing machine M, and $\rho(\mathbf{w})$ is an encoding of the input to that Turing Machine, and simulates the action of M on $\mathbf{w}$. More exactly, the UTM takes the input string, checks to see if it is of the form $\rho(M)\rho(\mathbf{w})$, (if not, it goes into an infinite loop), simulates the action of M on $\mathbf{w}$, and if M would halt with output $\mathbf{y}$, then U also halts with the same output.

For convenience, we fix a Universal Turing Machine which accepts inputs in an alphabet of size q, and which 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 $\rho(\mathbf{w})$ needs no such structure, however, so $\rho(\mathbf{w}) = \mathbf{w}$ in this formulation.

Suppose that on a given input $\mathbf{x}$, the UTM halts, leaving the string $\mathbf{v}$ to the right of the read-write head. We say that $\mathbf{v}$ is computed by U on $\mathbf{x}$. We define the *Kolmogorov (or Kolmogorov-Chaitin) complexity* [6, 7] of a string $\mathbf{s}$ 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 $\mathbf{s}$ on the tape to the right of the read-write head. This quantity is also called the *algorithmic information content* of $\mathbf{s}$ [50].

Definition 1.3 *The Kolmogorov (or Kolmogorov-Chaitin) complexity of a string* $\mathbf{s}$ *is a function* $K : \{0, 1, \ldots, q-1\}^* \to Z$ *defined by*

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

The following theorem summarizes the main properties of this function.

Theorem 1.1 *(i) There exists a constant* c_0 *such that* $K(\mathbf{s}) \leq n + c_0$ *for any* $\mathbf{s}$ *and* $n = |\mathbf{s}|$.

(ii) The fraction of $n-$*tuples* $\mathbf{s}$ *with* $K(\mathbf{s}) < n - c_1$ *is less than* q^{-c_1}.

(iii) The Kolmogorov complexity of a string is, in general, uncomputable.

(iv) If $\mathbf{s}$ *has length* n *and weight* λn*, then* $K(\mathbf{s}) \leq nH_q(\lambda) + o(n)$ *for large* n*, where* $H_q(x)$ *is the* q*-ary entropy function* $-x\log_q x - (1-x)\log_q(1-x) + x\log_q(q-1)$ *for* $0 < x < 1$.

(v) The running time of the shortest program for an arbitrary string is not bounded by any computable function of the length of the string.

Proof: (i) Consider the everhalting machine E which goes directly from the starting state to the halting state. Let $|\rho(E)| = C$. Then the input $\rho(E)$s to the UTM produces the output s, so $K(\mathrm{s}) \leq |\mathrm{s}| + C$.

(ii) 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 less than q^{n-c_1} strings of length n can have such a low complexity.

(iii) Let L be an arbitrary natural number. Consider the following program that generates a string s_L of Kolmogorov complexity $K(\mathrm{s}_L) > L$ using the algorithm for computing the Kolmogorov complexity:

— Generate all strings lexicographically: $0, 1, \ldots, q-1, 00, \ldots$.

— Compute the Kolmogorov complexity of each.

— Stop when the first string s_L with complexity greater than L is found.

— Report 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 $K(\mathrm{s}) > L$.

(iv) Take n and λn and generate lexicographically all strings of length n with weight λn. Specify which one of these is the string s. This 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.
(v) Suppose this is false, and that there exists a computable function $\tau(n)$ such that the shortest program for a string of length n halts in at most $\tau(n)$ steps on the Universal Turing Machine. We have the following program for computing $K(\mathrm{s})$, contradicting (iii):

— Take s, find n, and compute $\tau(n)$.

— Generate all programs lexicographically: $0, 1, \ldots, q-1, 00, \ldots$.

— Simulate the Universal Turing Machine on each program for $\tau(n)$ steps.

— Find the first program that halts within $\tau(n)$ steps leaving s on the tape.

— Find and report the length of the program, and halt.

▶

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 shown above, *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 (ii)). 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 (iii) above 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 which is insignificant with respect to n.

Note also that in this formulation we are not concerned with the running time of the shortest program. Indeed, the running time can be arbitrarily large. We wish to allow only programs which run efficiently, and so we concentrate on the *time-bounded* Kolmogorov complexity, defined to be the shortest program which will run on the Universal Turing Machine, halting within $\tau(n)$ steps, leaving the desired output string on the tape. The above properties (i), (ii) and (iv) 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. The effect of introducing the time bound is to declare some strings that are non-random to be random given a certain computing time budget. The following result holds:

Theorem 1.2 *Let $\tau(n)$ be a computable function. Then for any m, there is a string* s *of some length N such that*

$$K_{\tau(n)}(\mathrm{s}) \geq N$$

and

$$K(\mathrm{s}) < \underbrace{\log\log\ldots\log N}_{m \text{ times}}.$$

Proof: Consider the following program that takes n as a parameter:

— Compute
$$N = \left.2^{2^{2^{\cdot^{\cdot^{2^n}}}}}\right\}m.$$

— Compute $T = \tau(N)$.

— Simulate all programs of length $< N$ for T steps each.

— Keep track of those strings of length N that were generated in this simulation.

— Determine the first string s of length N that has not been generated.

— Report the string and halt.

Now $K_{\tau(n)}(\mathbf{s}) \geq N$ by our construction. The program length is $\leq C + \log n = C + \log(\log^{(m)} N) < \log^{(m)} N$ for large enough N. So

$$K(\mathbf{s}) < \underbrace{\log\log\ldots\log N}_{m \text{ times}}.$$

1.3 The Gilbert-Varshamov Bound

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

Theorem 1.3 *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 $\mathbf{x}$ for which $H\mathbf{x}^T = 0$. If all combinations of $d-1$ or fewer columns of H are linearly independent, no non-zero 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 $i \leq n-1$. ▶

Henceforth, we concentrate exclusively on the asymptotic version of the bound given above, which is

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

where

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

for $0 < x < 1$. (This assumes that $d/n \leq (q-1)/q$; the Plotkin bound [10] implies that this holds for the codes we are interested in.)

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 1.4 *Let $\Phi=\{\Phi_1, \Phi_2, \ldots\}$ be an infinite family of linear codes over $GF(q)$, where Φ_i is a set of (n_i, k_i) codes such that (i) $k_i/n_i > R$ and (ii) each non-zero 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 which contain a given non-zero vector. If we write out all the non-zero 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 non-zero 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

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

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 non-zero in the first k symbols. We find that the fraction of such codes with minimum distance d or less is at most

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

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

We show in Appendix A 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* behaviour of codes. It is still not known if the bound is tight: in the binary case there exists a sizeable 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 $E(R_e, L)$ which 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.

1.4 Complexity and the Gilbert-Varshamov Bound

1.4.1 All Random Codes Are Good

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 $(= k/n)$ 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 1.4 *The defining string of a linear systematic code $\mathcal{C}$ over $GF(q)$ is the string*

$$s_{\mathcal{C}} = \{\tilde{p}_1, \ldots, \tilde{p}_i, \ldots, \tilde{p}_k\}$$

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

[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.

In a slight abuse of notation, we say that the *Kolmogorov complexity of a code* is the Kolmogorov complexity of the string which 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 1.5 *Let $\mathbf{\Phi}$ 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 string $\mathbf{s}_j$. Then there exists a constant C_0 such that*

$$K(\mathbf{s}_j) \leq C_0 + n^2R(1-R) + n\Big[H_q(d/n) - (1-R)\Big] + O(\log_q n)$$

for all j.

Proof: We outline a program for a Universal Turing Machine which 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 $\mathbf{r}_i$. 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 $\mathbf{i}$, while the last $n-k$ symbols represent the parity check sequence $\mathbf{p}$. Let the support of the information vector $\mathbf{i}$ be $\{\alpha_1, \ldots, \alpha_m\}$, where m is the number of non-zero symbols in the information sequence. Clearly, the specified codeword is of the form $\mathbf{c} = \sum_{j=1}^{m} s_j \mathbf{r}_{\alpha_j}$, where $s_j \in GF(q)$. In other words, $\mathbf{c}$ is a sum of m rows of the generator matrix; which m rows these are can be determined from the first k symbols. Suppose we are given $\mathbf{c}$ (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 which represents $\mathbf{c}$ (*i.e.*, row α_m in the notation given above). Then it is a simple matter to recover $\mathbf{r}_{\alpha_m}$ by calculating $\mathbf{r}_{\alpha_m} = \mathbf{c} - \sum_{j=1}^{m-1} s_j \mathbf{r}_{\alpha_j}$. 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 which performs the calculations, a specification of the low weight word, and specification of all rows of the parity matrix except one. The encoding 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, 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 $\lceil \log_q \left(\binom{n}{d} (q-1)^d \right) \rceil$ symbols[2] Using the relation $\binom{n}{\lambda n}(q-1)^{\lambda n} \leq q^{nH_q(\lambda)+o(n)}$ we find that the program has length at most

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

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 $\mathbf{0}$. We can thus omit one column of H, saving $n-k$ symbols, deriving it from the given codeword.

Corollary 1.6 *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 $H_q(d/n) \leq 1 - R - \sigma$ is less than $q^{-n\sigma + C\log_q n}$ for some constant C and for all n.*

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

$$q^{n\left[H_q(d/n)-(1-R)\right]+O(\log_q n)}$$

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

▶

The result above also implies the following interesting and important observation:

[2]Note that we can recover the word of weight d from its specification in polynomial time. We interpret the $\lceil \log_q d \rceil$ symbols as a number; the first symbol must be zero if the number is less than $\binom{n-1}{d}(q-1)^d$, and so on. Then there is an obvious recursion for the other symbols. This will be important later.

Theorem 1.7 *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 $GF(q)$ which has Kolmogorov complexity (or polynomial-time-bounded Kolmogorov complexity) no less than $n^2R(1-R) - C_0$ symbols must have minimum distance d satisfying $H_q(d/n) \geq 1 - R + o(1)$, 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 & Reiffen, we assert that *"any code which is sufficiently random is good."*

Comparing our derivation to the standard one given in Section 1.3, 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 which 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 symbols, then insist only that the remaining linear number should allow compression to a percentage of its length.

1.4.2 A Probabilistic Converse

We need to ask about the converse of the above 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 which 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 which 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 1.8 *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*

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

regardless of the block length n.

Proof: The restrictions on the codes imply that each code string has complexity $K(\mathrm{s}) \leq B + n^2R(1-R) - n\sigma$ 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 which have sufficiently low complexity. For a given n, pick d^* to be the largest integer such that

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

From our previous arguments, a code with a minimum distance $\leq d^*$ 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

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

We take the "sufficiently large" condition on the budget B to mean that $B \geq C_0$. 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 1.3 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 non-zero $n-$tuples of weight d^* or less, where a choice is valid if there is any systematic linear code which contains that set of codewords.[3] Let $S_i(n)$ be the number of systematic linear codes containing a given valid set of i non-zero codewords, averaged over all valid sets of i non-zero 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

$$N_2(n,d^*) = \binom{\sum_{j=1}^{d^*}\left(\binom{n}{j} - \binom{n-k}{j}\right)(q-1)^j}{2}\Big/(q-1)^2 - E_2(n,d^*),$$

where $E_2(n, d^*)$ represents the number of unordered pairs of non-zero codewords each of weight $\leq d^*$ which have the same non-zero sequence in their first k symbols. We also have $S_1(n) = q^{(k-1)(n-k)}$ and $S_2(n) = q^{(k-2)(n-k)}$. Thus

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

[3]A codeword c has the same weight as λc for any non-zero $\lambda \in GF(q)$. To avoid counting multiple copies of the same code, we normalize each of the i distinct non-zero n-tuples to have 1 as the first component.

$$+\frac{1}{2}\sum_{i=1}^{d^*}\left[\left\{\binom{n}{i}-\binom{n-k}{i}\right\}(q-1)^i\right]q^{(k-2)(n-k)-2}.$$

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

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

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

1.4.3 Discussion

Of course, there are codes of low Kolmogorov complexity which 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 which 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 sub-exponential running time. Our result above shows that even with unconstrained running time, random selection of a low complexity code has a certain minimum likelihood of producing a bad code. More importantly, taking the *time-bounded* Kolmogorov complexity mentioned earlier, and setting $\tau(n)$ equal to an appropriate polynomial function, we have the same lower bound on the probability of selecting a bad code (because our described procedure for bad codes runs in polynomial time) 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 1.3, we know that the fraction of linear codes over $GF(q)$ with $H_q(d/n) \leq 1-R-\sigma$ 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 $H_q(d/n) \leq 1-R-\sigma$, the Kolmogorov complexity of the defining string of any such code is upper-bounded by

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

Thus, again, any code which is effectively random must satisfy the Gilbert-Varshamov bound. Conversely, assume that the fraction of codes with $H_q(d/n) \leq 1 - R - \sigma$ 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 $\leq C_0 + n^2R(1-R) - \lceil g(n,R,\sigma)\rceil$ 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 1.7, and in Chapter 2, Section 2.5, we use the same idea to analyse 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 1.8. 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 polynomial time, whereas in the second case, there is no reason to believe that the program does so. This means that in the results in Section 1.4, 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 $K(\mathrm{s}) < K_u(\mathrm{s}) - (C_0 + 1) - C_1$ is less than q^{-C_1} for any constant C_1, where $K_u(\mathrm{s})$ 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 which have any non-zero 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

$$\leq n^2R(1-R) - 2n(1-R) + 2nH_q(d/n) + o(n) = n^2R(1-R) - 2n\sigma + o(n)$$

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

It is also clear from the discussion of Theorem 1.5 that the *diameter, r,* of virtually all linear codes over $GF(q)$ satisfies $H_q(r/n) \geq 1 - R + o(1)$, where the diameter of a code is defined [10] as the *maximum* distance between any two codewords.

1.5 Non-linear Codes

In the case of non-linear 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. Although the average behaviour of random non-linear codes has been studied extensively [21, 61], there are no results on the average distance of such codes known to us. We shall derive results dealing with this problem in this section.

We have the following argument. An $[n, M]$ non-linear code over $GF(q)$ is a collection of M distinct q-ary $n-$tuples. Let the function $f(n, M)$, for arbitrary but fixed q, be such that the number of $[n, M]$ non-linear codes over $GF(q)$ is $q^{f(n,M)}$. Then $\lceil f(n, M) \rceil$ q-ary symbols is the minimum amount required in order 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 $\mathcal{C}$. Then there are two codewords $\mathbf{w}_1$ and $\mathbf{w}_2$ which are such that $\text{wt}(\mathbf{w}_1 - \mathbf{w}_2) = d$. We can specify the code using the following procedure: specify the $[n, M-1]$ code obtained by deleting $\mathbf{w}_2$ from $\mathcal{C}$, specify which word in the expurgated code is $\mathbf{w}_1$, give the $n-$tuple $\mathbf{w}_1 + \mathbf{w}_2$ of weight d, and reconstruct $\mathbf{w}_2$ 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 $\mathbf{w}_1$, 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

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

and we find that

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

for most non-linear 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 non-linear code with at most M codewords. If code C has minimum distance d, there are two codewords $\mathbf{c}_1$ and $\mathbf{c}_2$ which 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

$\mathcal{C}_1$ obtained by removing word $\mathbf{c}_2$ from $\mathcal{C}$. Specify which word in this subcode is $\mathbf{c}_1$, give the n–tuple of weight d which is $\mathbf{c}_1 - \mathbf{c}_2$, then say *where* $\mathbf{c}_2$ is located in the code string for $\mathcal{C}$. 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 from this argument to be met automatically 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: (i) are there really binary non-linear codes which are almost totally random which have $H_q(d/n) \approx 1 - 2R$ for $R < 1/2$ and $d \to 0$ for $R \geq 1/2$? (ii) is the average behaviour of non-linear 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.*, $H_q(d/n) = 1-2R+\epsilon$ for $R < 1/2$, and $H_q(d/n) = \epsilon$ for $R \geq 1/2$, for some $\epsilon > 0$.

We take a high complexity non-linear $[n, M-1]$ 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 non-linear 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 (i) above. Thus we conclude that the bound is tight for a fraction of non-linear codes that is bounded away from zero.

It is possible to obtain a stronger result:

Theorem 1.9 *The fraction of non-linear $[n, q^{nR}]$ codes over $GF(q)$ satisfying*

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

for any $\alpha > 0$ is less than $q^{-n\alpha+o(n)}$.

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 $\mathbf{X}$ be a random variable denoting the number of unordered pairs of codewords at distance $\leq d$ from each other. We can find $E(\mathbf{X})$ and $E(\mathbf{X}^2)$ and hence can bound $\Pr(\mathbf{X} = 0)$ using Chebyshev's inequality.

Let $\mathbf{X}_{ij}$ be a random variable which takes the value 1 if the ith and jth codewords are at distance $\leq d$ from each other, and 0 otherwise. Then $\mathbf{X} = \sum_{i<j} \mathbf{X}_{ij}$, and so by linearity of expectation [28]

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

Also

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

We find that

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

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

where the first term represents the products $\mathbf{X}_{ij}\mathbf{X}_{kl}$ 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 $\sigma^2(\mathbf{X}) = E(\mathbf{X}^2) - E^2(\mathbf{X})$

$$\begin{aligned} &= \frac{1}{2}M^2p\left(1+O(M^{-1})+O(p)\right)+p^2O(q^{-n})O(M^4) \\ &= \frac{1}{2}M^2p\left(1+o(1)\right) \end{aligned}$$

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

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

Finally,

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

and the theorem follows. ▶

Thus most non-linear codes do not satisfy the Gilbert-Varshamov bound. This raises the question of *how many* codewords are at a low distance from the rest of the code. A simple modification of the proof of Theorem 1.8 gives us:

Theorem 1.10 *Let ω and R be such that $1-2R < H_q(\omega) < 1-R$. The number of codewords at distance at most $n\omega$ from another codeword is then*

$$q^{n[H_q(\omega)-(1-2R)]+c^2\sqrt{n}+O(\log n)}$$

for some $c < 1$ for a fraction of at least $1-q^{-\sqrt{n}+o(n)}$ of all non-linear $[n, M = \lceil q^{nR}\rceil]$ codes. The fraction of codewords at distance $\le n\omega$ from another codeword is

$$q^{n[H_q(\omega)-(1-R)]+c^2\sqrt{n}+O(\log n)}$$

for some $c < 1$ for at least the same fraction of $[n, M]$ codes.

Thus deleting an asymptotically insignificant fraction of the codewords results in a code meeting the Gilbert-Varshamov bound. The result above shows exactly what this fraction is in most cases. Note that Shannon's original proof of the channel coding theorem [104] used random non-linear codes without expurgation; to achieve a low probability of error for *every codeword*, rather than a low *average* probability of error over all codewords, requires the appropriate amount of expurgation.

It is intuitively surprising that, despite the use of random non-linear codes in the proof of the channel coding theorem, these codes should have low minimum distance on average; it is even more surprising that they should have lower minimum distance than random linear codes.

To highlight the contrast between the two classes of codes, consider an intermediate type of code consisting of q^{nR_1} linear codes, each containing q^{nR_2} codewords. We label such a code an (n, R_1, R_2) ULS [4] code. This class, non-linear with constraints in general, ranges from linear codes ($R_1 = 0$) to unconstrained non-linear codes ($R_2 = 0$). An example of the intermediate case is provided by the family of *Kerdock* codes $\mathcal{K}(m)$, which consist of the first-order Reed-Muller code $\mathcal{R}(1,m)$ and $2^{m-1} - 1$ cosets of $\mathcal{R}(1,m)$ in the second-order Reed-Muller code $\mathcal{R}(2,m)$ [10].

We have the following guarantee on distance, generalizing the results for linear and non-linear codes:

Theorem 1.11 *Virtually all $[n, R_1, R_2]$ ULS codes over $GF(q)$ satisfy*

$$H_q(d/n) \geq 1 - R - R_1 + o(1),$$

where $R = R_1 + R_2$.

Proof: Take the order in which the linear subcodes are arranged as important. Then the (ordered) ULS code has Kolmogorov complexity approximately equal to $q^{nR_1} n^2 R_2(1 - R_2)$ in most cases. Given two codewords at distance d, we can save $n(1 - R_2)$ symbols in the specification of one of the subcodes by specifying which subcode the two codewords lie in (taking nR_1 symbols each) and giving the difference

[4]Union of linear subcodes.

between them (taking $nH_q(d/n) + o(n)$ symbols). Then in most cases, $H_q(d/n) \geq 1 - R_2 - 2R_1 + o(1) = 1 - R - R_1 + o(1)$. ▶

Thus with this notation, there is a linear transition between the properties of random linear and non-linear codes.

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

$$\begin{aligned} &\leq \left(1 - q^{-n\left(1-H_q(d/n)\right)+o(n)}\right)^{q^{nR}} \\ &= \left[\left(1 - q^{-n\left(1-H_q(d/n)\right)+o(n)}\right)^{q^{n\left(1-H_q(d/n)\right)+o(n)}}\right]^{q^{n\sigma+o(n)}} \\ &\to e^{-q^{\sigma n}} \end{aligned}$$

as n becomes large.

1.6 Burst-Error-Correcting Codes

Although our discussion so far has concentrated on the classic case where errors occur randomly and independently throughout the codeword, it should be clear that more general results should be possible using the same techniques. In this section, we derive the performance of linear and non-linear codes in burst-error-correcting and mixed random- and burst-error-correcting schemes; we show that the results generalize in a natural way.

Taking the pure burst-error-correction scheme first, we define a *burst* of length l to be a vector whose only non-zero components are among l successive components, the first and last of which are non-zero[5]. We shall regard the first position as being the successor of the last position; the results are not significantly different if the alternative convention is adopted. An old result, analogous to the Singleton bound for random-error-correcting codes, gives an upper bound on the rate of a linear block code in terms of its burst-error-correcting capability:

Theorem 1.12 (Reiger 1960) *A linear (n, k) code that corrects all error patterns of length b or less must have $2b \leq n - k$.*

Proof: The code cannot contain any codeword of length $2b$ or less; otherwise we could bisect the codeword to get two bursts, each of length at most b, that would be in the same coset. But any two patterns that have all their non-zero components in the first $2b$ symbols must then be in different cosets, because their sum is a burst of length at most $2b$. There are q^{2b} words that are zero except in their first $2b$ symbols, and hence at least $2b$ parity check symbols. ▶

Many researchers have generalized the Gilbert construction to the case of burst-error-correction with various constraints [59, 60, 62]. We derive most of these properties, and many more, using Kolmogorov complexity. First we show that the Reiger bound is tight for most linear systematic codes, and that, as for the random-error-correction case, any sufficiently random burst-error-correcting code is good.

Theorem 1.13 *Any linear systematic (n, nR) code over $GF(q)$ that does not correct at least one burst of weight b has a Kolmogorov complexity upper bounded by*

$$n^2R(1-R) + 2b - n(1-R) + 2\log n + C_0 \log\log n + C_1$$

symbols (where logarithms are base q). Thus a fraction of at least $1 - q^{-n\alpha + o(n)}$ of all linear systematic (n, nR) codes over $GF(q)$ have a burst-correction power b satisfying $2b > n(1 - R - \alpha) + o(n)$.

[5]This is the usual definition — see [57], for example; an alternative definition given by Chien & Tang [71] is that a burst of length b is a set of b consecutive symbols the first of which is non-zero (the last can be zero). Bridwell & Wolf [62] extend this to multiple burst correction. Our results still apply under this convention.

Proof: A burst of length b that is not corrected must be in the same coset as a burst of length $\leq b$. The difference between the two bursts is a codeword. Thus we can specify the code by giving all rows but one of the parity matrix, the burst that is not corrected, and the leader of the coset in which the uncorrectable burst lies. To specify a burst of length b, we need to give the location of the beginning of the burst, the value of b, and the burst itself. The burst itself takes b symbols to specify, we need $\lceil \log_q n \rceil$ symbols to specify the value of n and, as $2b$ is usually $n(1-R) - O(\log_q n)$,[6] we need $O(\text{loglog } n)$ symbols to specify b. All rows but one of the parity matrix can be specified in $(k-1)(n-k)$ symbols, so the program length is as given in the theorem. The other claims are obvious consequences. ▶

This claim is not as sharp as that obtained by Peterson [57]: he shows that there exists an (n, nR) linear code correcting bursts of length b or less with

$$2b \geq n(1-R) - \log_q[(q-1)(n-2b-1)+1],$$

whereas our version is that for $\delta > 0$ and fixed rate R, the fraction of (n, nR) linear systematic codes with

$$2b < n(1-R) - 2\log_q n - \delta \log_q n$$

is less than $n^{-\delta(1+o(1))}$.

Theorem 1.10 has the following corollary, analogous to the random-error-correcting case:

Corollary 1.14 *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 $GF(q)$ which has Kolmogorov complexity (or polynomial-time-bounded Kolmogorov complexity) no less than $n^2R(1-R) - C_0$ symbols must have burst correction power b satisfying $2b \geq n(1-R) + o(1)$,* i.e., *must be arbitrarily close to the Reiger bound.*

We shall discuss this result in Section 1.7.3.

[6]We would get this result even by taking $\lceil \log_q n \rceil$ symbols to specify b, and could then run through the argument again; thus we are not begging the question.

1.6.1 Combined Random- and Burst-Error Correction

The permutations possible are almost endless. We summarize some of the main results. The results in this section are, to the best of our knowledge, original.

Theorem 1.15 *Virtually all linear (n, nR) codes correct all error patterns which have weight up to $n\tau$ or burst length up to $n\beta$ if β and τ satisfy*

$$\max\left[2\beta, \beta + (1-\beta)H_q\left(\frac{\tau}{1-\beta}\right), H_q(2\tau)\right] \le 1 - R + o(1).$$

The fraction of codes for which this bound is tight approaches 1 asymptotically.

Proof: If the stated correction is not achieved, there must be an error pattern meeting the requirements that is miscorrected. The cases where a burst error is miscorrected to a burst error and where a random error is miscorrected to a random error have been dealt with before. If a burst error pattern is miscorrected to a random error pattern or *vice versa*, we can specify a non-zero codeword with $n\beta + n(1-\beta)H_q(\tau/(1-\beta)) + o(n)$ symbols: we need $n\beta + o(n)$ symbols for the burst of length $n\beta$, and then the remaining $n(1-\beta)$ symbols of the codeword contain at most $n\tau + o(n)$ non-zero symbols. Thus by the familiar argument, if $\beta + (1-\beta)H_q(\tau/(1-\beta)) \le 1 - R + o(1)$ and the code miscorrects a burst pattern to a random pattern or *vice versa*, then the code has significantly lower complexity than a random one.

It is not clear at first glance that the centre term on the left hand side above can ever be the maximum. To show that it can be, note that we can choose β and τ so that β is less by an arbitrarily small amount than $(1-\beta)H_q(\tau/(1-\beta))$, so that the centre term is greater than the first term. The difference between the central term and the last term is then

$$\begin{aligned}(1-\beta)H_q(\tau/(1-\beta)) + \beta - H_q(2\tau) &= 2(1-\beta)H_q(\frac{\tau}{1-\beta}) - \epsilon - H_q(2\tau) \\ &= 2\Big[(1-\beta)H_q(\frac{\tau}{1-\beta}) - \frac{1}{2}H_q(2\tau)\Big] - \epsilon \\ &> 0\end{aligned}$$

because by the convexity of the entropy function, $\alpha H_q(\gamma) < H_q(\alpha\gamma)$ for $\alpha < 1$.

The fact that the bound is tight if the first term is the maximum follows from the Reiger bound, and if the last term is the maximum, the result follows from the tightness of the Gilbert-Varshamov bound for most linear codes — see Theorem A-2 in Appendix A. In the case where the central term is the maximum, we use the following argument. (Let $(1-\beta)H_q(\tau/(1-\beta))+\beta-(1-R)=\epsilon>0$.) Suppose we have already selected the first $k-1$ rows of the generator matrix. We determine the number of choices for the last row that will result in a code having in the same coset a burst of length $n\beta$ or less and a random pattern of weight $n\tau$ or less confined to the positions outside the burst, with the difference between the two patterns involving the last row of the generator. We let the random variable $\mathbf{X}$ denote the number of such pairs in a code. We have

$$\begin{aligned}\mathrm{E}\mathbf{X} &= \binom{n(1-\beta)}{n\tau}(q-1)^{n\tau}q^{n\beta}q^{-n(1-R)}\\ &= q^{n[(1-\beta)H_q((\tau/(1-\beta))+\beta-(1-R)]+o(n)}\\ &= q^{n\epsilon}.\end{aligned}$$

Thus if $(1-\beta)H_q((\tau/(1-\beta))+\beta>1-R$, we expect on average to have many miscorrections per code. To estimate the deviation from the mean, index each potentially miscorrectable pair with index i, $1\le i\le M$ for appropriate M, and let $\mathbf{X}_i$ be 1 if that pair is miscorrected and 0 otherwise. Then $\mathrm{E}\mathbf{X}^2=\mathrm{E}(\sum_i\mathbf{X}_i)+\mathrm{E}(\sum_i\sum_j\mathbf{X}_i\mathbf{X}_j)<n\mathrm{E}\mathbf{X}+\mathrm{E}^2\mathbf{X}$, so $\sigma^2(\mathbf{X})<n\mathrm{E}\mathbf{X}$. We have used the fact that $\mathrm{E}\mathbf{X}_i\mathrm{E}\mathbf{X}_j=\mathrm{E}^2\mathbf{X}_i$ if the ith and jth pair add up to different words. For each pair, the maximum number of other pairs that add up to the same word is $n-1$ (there are $n-1$ other positions in which to start the burst), so $\mathrm{E}\sum\sum_{i\equiv j}\mathbf{X}_i\mathbf{X}_j\le(n-1)\mathrm{E}\sum_i\mathbf{X}_i=(n-1)\mathrm{E}\mathbf{X}$, where the notation $i\equiv j$ means that the ith and jth pairs give the same codeword. Chebyshev's inequality then gives

$$\Pr(\mathbf{X}=0)<q^{-n\epsilon+o(n)},$$

so virtually all codes do have miscorrection if $(1-\beta)H_q((\tau/(1-\beta))+\beta>1-R$.

▶

Theorem 1.16 [7] *Virtually every linear (n, nR) code corrects all bursts of length up to b that have weight up to w if $2b\bar{H}_q(w/b) \leq 1 - R + o(1)$, where $\bar{H}_q(x) = H_q(x)$ if $x < (q-1)/q$, and is 1 otherwise.*

Proof: If we do not achieve the correction, there are two bursts of length at most b and weight at most w in the same coset. Specifying each takes $b\bar{H}_q(w/b) + o(n)$ symbols, and by specifying both we save $n(1-R) + o(n)$ symbols in the specification of the code string. ▶

Theorem 1.17 *Virtually every linear code corrects all patterns of $f(n)$ bursts of errors of length $b_i, 0 \leq i \leq f(n)$ if $f(n) = o(n/\log n)$ and $2\sum_i b_i < n(1-R) + o(n)$.*

Proof: Specifying the b_i's takes $O(f(n)\log n)$ symbols; then specifying the bursts takes $\sum_i b_i$ symbols. Thus we need $2(\sum_i b_i + O(f(n)\log n)) = 2\sum_i b_i + o(n)$ symbols to specify a non-zero codeword. ▶

1.6.2 Non-linear Burst-Error-Correcting Codes

Recall that in Section 1.5, we defined $f(n, M)$ (for arbitrary but fixed q) to be such that the number of $[n, M]$ non-linear codes over $GF(q)$ is $q^{f(n,M)}$. Then, as before, $\lceil f(n, M) \rceil$ q-ary symbols are necessary to be able to specify any code in the ensemble.

[7]The reader might like to compare the expression to that given by Dass [59]:
Theorem (Dass 1975) *Given positive integers w and b such that $w \leq b$, there exists an (n, k) linear code that corrects all bursts of length b or less with weight w or less satisfying the inequality*

$$\begin{aligned} q^{n-k} \leq\ & [1+(q-1)]^{(b-1,w-1)}\Big\{q^{w-1}[(q-1)(n-b-w+2)+1] \\ & +(q-1)^2 \sum_{i=w+1}^{b} (n-b-i+2)[1+(q-1)]^{(i-2,w-2)}\Big\} + \sum_{i=w}^{2w-1} \binom{b-1}{i}(q-1)^i \\ & +\sum_{k=1}^{b-1} \sum_{r_1,r_2,r_3} \binom{b-k-1}{r_1}\binom{k}{r_2}\binom{b-k-1}{r_3}(q-1)^{r_1+r_2+r_3+1}, \\ & \qquad 0 \leq r_1 \leq w-2, 1 \leq r_2 \leq 2w-2, 0 \leq r_3 \leq w-1, \\ & \qquad r_2 + r_3 \geq w, r_1 + r_2 + r_3 \leq 2w-2, \end{aligned}$$

where $[1+x]^{(m,r)}$ denotes the incomplete binomial expansion of $(1+x)^m$ up to the term x^r in ascending powers of x.
Dass uses an adaptation of the construction given in Section 1.3 for random-error correction; we conjecture that, as for the case of random-error correction, an asymptotic version will be no stronger than our result.

Suppose the code corrects bursts of length up to $b-1$, but miscorrects at least one burst of length b. Then there are two codewords $\mathbf{w}_1$ and $\mathbf{w}_2$ such that the word $\mathbf{w}_1 - \mathbf{w}_2$ consists of the sum of two bursts of length at most b. We can specify the code by using the following procedure: specify the $[n, M-1]$ code obtained by deleting $\mathbf{w}_2$ from $\mathcal{C}$, specify $\mathbf{w}_1$ and $\mathbf{w}_1 - \mathbf{w}_2$, and recover $\mathcal{C}$ from that information. The program length cannot be much shorter than $f(n, M)$ symbols in most circumstances. We have

$$2n\beta + nR + o(n) \geq f(n, M) - f(n, M-1) = n(1-R)(1+o(1))$$

and so

$$2\beta \geq 1 - 2R + o(1)$$

for most non-linear codes. We find a loss in error-correcting power analogous to that found in the random-error-correcting case.

Tightness follows from another application of Chebyshev's inequality. Select M codewords over $GF(q)$ at random, and let $\mathbf{X}$ be the number of codewords at distance at most $2n\beta$ apart. We have $\mathrm{E}\mathbf{X} = \binom{M}{2}\mathrm{E}\mathbf{X}_{ij}$, where $\mathbf{X}_{ij}$ is the event that two randomly chosen words are within distance $2n\beta$, so $\mathrm{E}\mathbf{X} = q^{n[2\beta-(1-2R)]+o(n)}$. As in the random-error-correcting case, $\sigma^2(\mathbf{X}) = O(\mathrm{E}^{-1}(\mathbf{X}))$. Thus if $2\beta > 1 - 2R + o(1)$, virtually all $[n, q^{nR}]$ non-linear codes over $GF(q)$ fail to correct all bursts of length up to $2n\beta$.

1.6.3 Discussion

In Section 1.4, we showed that any code that is sufficiently random must meet the Gilbert-Varshamov bound, and that conversely, random selection from those codes that have some pattern results, with probability bounded away from zero, in a code that does not meet the bound. We suggested that this is an explanation for the fact that no construction is known for arbitrary symbol fields that yields codes lying on the bound. In Section 1.6.1, we showed that the same results hold for burst-error-correcting codes. For this case, however, it is trivial to construct codes that meet the best known bound: we can interleave m copies of a given (n, nR) linear code that corrects bursts of length b or less to get an (nm, nmR) linear code that corrects bursts of length mb or less. For any rate R, we can search exhaustively to find a 'seed' code that meets the bound, and then interleave to get arbitrarily long codes of the same rate that also meet the bound [21]. This demonstrates an important point: our argument in the random-error-correcting case should not be construed as a claim that no polynomial time construction can exist for good codes. Rather, it shows that the fact that virtually every code is good has no bearing whatsoever on the problem of determining whether such a construction exists. There is no known way of showing that a construction does not exist, whereas to show that a construction does exist, we need only demonstrate it and prove that it works; the fact that a construction exists for the burst-correction case and for the random-error-correction case over large symbol fields, does not help us in the general random-error-correcting case. An analogy is the decoding problem: for virtually every burst-error-correcting code, error trapping produces the nearest codeword to any received vector in polynomial time, but this does not mean that the complete decoding problem with random-error correction is easy.

1.7 Other Classes of Codes

We feel that the ideas in Kolmogorov complexity provide a useful and intuitively appealing tool for analysing 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 behaviour of the class of codes. We restrict ourselves here to a few examples to illustrate the idea. Alternative proofs of the results on distance given below can be found elsewhere [29, 30].

1.7.1 Shortened Cyclic Codes

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 $\mathbf{c}$ which 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 $(g_{n-k-1}, \ldots, g_0)$. 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 $H_2(d/n) \leq 1 - R - \sigma + o(1)$ 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 1.7 and 1.8 apply here also. Bad codes (with $H_2(d/n) \leq 1-R-\sigma$ for $\sigma > 0$) have Kolmogorov complexity at most $n\Big(1-R-\sigma+o(1)\Big)$ bits; conversely, random selection from codes with Kolmogorov complexity at most $n\Big(1-R-\sigma+o(1)\Big)$ bits results with non-zero probability in a code with $H_2(d/n) \leq 1-R-\sigma$. This does not show, however, that the same holds for polynomial-time-bounded Kolmogorov complexity.

Clearly, the same arguments can be extended to the cases of burst correction and combined burst- and random-error correction. Overall, we have the following result:

Theorem 1.18 *Virtually every shortened cyclic code over $GF(2)$ satisfies all the following conditions:*

— *Meets the Gilbert-Varshamov bound.*

— *Meets the Reiger bound.*

— *Corrects all error patterns of burst length $n\beta$ or less that have weight $n\omega$ or less provided* $2n\beta H_2(\omega/\beta) \lesssim 1 - R + o(1)$.

— *Corrects all error patterns that are either a burst of length less than $n\beta$ or a pattern of weight less than $n\tau$ provided that* $\max\,[2\beta, \beta + H_2(\tau/(1-\beta)), H_q(2\tau)] < 1 - R + o(1)$.

By using shortened cyclic codes, we demonstrate the existence of good codes that have polynomial-time-bounded Kolmogorov complexity at most $n(1-R)$ bits. This is considerably less than the complexity of random general linear codes, and is the lowest polynomial-time-bounded Kolmogorov complexity known to us to be sufficient to give good codes. We offer the following challenge: show that for arbitrarily large blocklengths, there are codes with polynomial-time-bounded Kolmogorov complexity no greater than $n(1-R-\sigma)$, for positive σ, that meet the Gilbert-Varshamov bound. We are tempted to conjecture that there are no such codes; it seems impossible to prove this, however, while it might be disproved, and so we merely suggest it as a possibility.

It is easy to verify that Theorem 1.7 applies to binary shortened cyclic codes; because the Kolmogorov complexity of these codes is linear in n, the result can be restated as follows: for sufficiently high budgets, random selection of a binary shortened cyclic code which can be produced by a Turing Machine M with $|\rho(M)|$ acting on input of length at most $\gamma(1-R)$ with $\gamma < 1$ results with probability at least $2^{-(B+1)}$ in a code with $H_2(d/n) \leq \gamma(1-R) + o(1)$. Thus the binary entropy of the distance to length ratio is equal to the complexity in a significant fraction of cases.

1.7.2 Quasi-Cyclic Codes

A *quasi-cyclic* code is one having the property that a cyclic shift of m places applied to any codeword results in a codeword. We show that there are quasi-cyclic linear codes that meet the Gilbert-Varshamov bound. Consider a code of length $2r$ and rate $1/2$

that is invariant under a cyclic shift of 2 positions. This is equivalent to (and hence has the same minimum distance as) a circulant code, *i.e.,* one with generator matrix $G = (I|P)$, where P is a circulant matrix. Any non-zero codeword is of the form $i(x)|f(x)$, where $f(x) = i(x)p(x) \bmod x^r - 1$. Two codes contain the same codeword only if there is some non-zero $i(x)$ for which $i(x)p_1(x) = i(x)p_2(x) \bmod x^r - 1$, *i.e.,* if $p_1(x) - p_2(x)|x^r - 1$. Now if q is a primitive element modulo r, then $(x^r - 1)/(x - 1)$ is irreducible over $GF(q)$, and a non-zero codeword can occur in only q codes. Thus given a non-zero codeword plus one extra symbol, we can recover the code. Assume that for a given q, there are infinitely many primes for which q is a primitive element. Defining the Kolmogorov complexity of the code to be the Kolmogorov complexity of the polynomial $p(x)$, we have a complexity of close to r symbols for most codes, and so $H_q(d/n) \geq 1/2 + o(1)$ asymptotically, so the class of codes meets the Gilbert-Varshamov bound. Assuming the Artin Conjecture or the Generalized Riemann Hypothesis, there are infinitely many primes for which 2 is a primitive element, so in this case the bound holds for binary codes; even without assuming either hypothesis, it is known that for at least one of 2, 3 or 5, the conjecture holds, so there is an infinite class of codes meeting the Gilbert-Varshamov bound over one of these fields.

1.7.3 Generalized Reed-Solomon Codes

Now consider the class of linear concatenated codes with Reed-Solomon outer codes and varying *non-systematic* 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 $(c_0, \ldots, c_{N-1})$ where the c_j's are elements from $GF(2^n)$, and then apply a 'template' $(r_0, \ldots, r_{N-1})$ where the r_j's are also from $GF(2^n)$ to get the resulting codeword $(c_0r_0, \ldots, c_{N-1}r_{N-1})$. 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 $(c_0r_0, \ldots, c_{N-1}r_{N-1})$ and $(r_0, \ldots, r_{NR-1})$, from which we calculate $(c_0, \ldots, c_{NR-1})$, then $(c_{NR} \ldots, c_{N-1})$ and finally $(r_{NR}, \ldots, r_{N-1})$. 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 versions of Theorems 1.5 and 1.8 apply, as do the results in Section 1.7.1.

1.8 Conclusions

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 by Wozencraft & Jacobs is no mere accident, but a fundamental principle of coding theory. We have also demonstrated that a converse holds: codes which are not effectively random have a certain non-zero probability of lying below the Gilbert-Varshamov bound. Furthermore, in a certain sense, the less random is the code, the further away from the bound it is likely to be.

The most interesting convention is to regard a code as random unless it can be recovered from a significantly compressed specification in polynomial time; even with this interpretation, the above results hold.

The behaviour of non-linear codes contrasts sharply with that of the linear codes, and the statement of Wozencraft & Reiffen cannot be said to apply to them. In both cases, the behaviour 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.

The results are shown to carry over to burst-error-correcting codes and combined random- and burst-error-correcting codes, and to other classes of codes, such as shortened cyclic and generalized Reed-Solomon codes.

In addition to shedding light on a celebrated paradox of information and coding theory, the techniques used provide a novel and intuitively appealing way of determining the properties of many classes of codes under different error-correction strategies.

Chapter 2

Complexity of Decoding General Linear Codes

2.1 Introduction

Shannon's original method of proof of the channel coding theorem has not one fundamental drawback, but two: one a problem for the transmitter of the information, and one a problem for the receiver. In Chapter 1, we discussed the difficulty in encoding the data for transmission across a noisy channel. The corresponding difficulty at the receiving end is that of *decoding* the code; the coding theorem states merely that it is possible. The most general decoding method is to compare a received sequence with every possible transmitted sequence and to choose the sequence that is likeliest to have been transmitted. This method has prohibitive complexity for large blocklengths; as a solution to the problem, it is analogous to the solution to the problem of finding a good code of searching through all codes and selecting the best. Ideally, we would like a polynomial time decoding algorithm that works for general linear codes.[1] No one has succeeded in developing such an algorithm, however, and results from the theory of NP-completeness (see Section 2.2) suggest that no such algorithm exists.

[1]Of course, the original coding theorem specified non-linear codes. It is clear that most of these require decoding complexity that is exponential in the blocklength. The case of linear codes is more interesting, so we concentrate on that.

Despite these difficulties, we seek to develop efficient general algorithms for decoding all or virtually all linear codes. The algorithms will deal with codes as combinatorial structures rather than as structures for which an algebraic representation is required, and will be *deterministic* rather than *probabilistic.* These points rule out many approaches to the decoding problem [52,91,92].

Before discussing some of the non-algebraic methods, we should outline some drawbacks to algebraic decoding. First, algebraic decoding is available only for codes with a very specialized structure. Second, the algebraic decoding algorithms decode no further than (and often not as far as) bounded distance and use hard decision only.

As a practical matter, the penalty in using hard decision decoding of binary data transmitted over a channel with additive white Gaussian noise is high — 2–3dB coding gain at operating points of interest. Even when communicating over the binary symmetric channel, there is a penalty in bounded distance decoding: we cannot transmit at such a high rate that the *expected* number of errors is greater than the bounded distance, or we will (by the strong law of large numbers) almost never be able to decode correctly for large blocklengths. The capacity of the binary symmetric channel with crossover probability p is $1 - H_2(p)$; communicating at this rate, the expected number of errors is $nH_2^{-1}(1 - R)$, which is asymptotically the same as the Goblick bound on covering radius (see Appendix B). Thus to achieve capacity using the average linear code, we need to decode out to the covering radius of the code. As the bounded error-correction capability of the code cannot be this high asymptotically [10], bounded distance decoding can never achieve capacity. In fact, for most linear codes, we cannot communicate reliably at a higher rate than $R = 1 - H_2(2p)$ for $p \leq 1/4$, or at any positive rate with $1/4 \leq p \leq 1/2$, with bounded distance decoding.

Thus we require non-algebraic methods in general to communicate at the rate guaranteed by the channel coding theorem for the average linear code. As we have noted, however, the decoding complexity of the full search procedure is prohibitive, and there is good reason to believe that any general algorithm must have exponential complexity. It has become customary to refer to problems for which a polynomial time algorithm exists as 'tractable,' while those for which there is no such algorithm are termed 'intractable.' The reasons for this are usually given in texts on NP-completeness [75]. There is, therefore, a need to justify our investigation of exponential algorithms for decoding. A simple justification that is often used for this situation [75] is that in practice we are interested in solving relatively short instances of the problem — *i.e.,* in decoding, we are interested in decoding codes of short to medium block lengths. Although the complexity will rise prohibitively for large blocklengths, decoding of medium blocklength codes may still be practicable. An example is given by the case of Viterbi decoding of convolutional codes. The algorithm is certainly 'intractable' for large enough constraint lengths, but the coding gain obtainable from short convolutional codes is large enough to make implementation worthwhile.

Although this explanation gives an adequate indication of our goal in examining these algorithms — that of deriving algorithms for medium length block codes — it does not give an adequate explanation of why this is sufficient for our purposes. The decoding problem is fundamentally different from many other NP-complete problems in that the *utility* of the problem does not vary polynomially with the instance size. In the Travelling Salesman problem [75], we should expect the value of a tour to be linearly proportional to the number of cities on the tour. Given this assumption, an exponential increase in complexity with number of cities is unacceptable. In the decoding problem, however, we are concerned with decoder error probability, and this decays *exponentially* with the block length for a wide class of channels of interest. Without examining the exact decoding complexity, we cannot say whether or not decoding is impractical: it may be that when the block length is so high that the complexity of decoding is beyond our means, the decoder error probability is beyond our requirements. A similar argument holds if we define utility in terms of coding gain: the rate of change of coding gain declines exponentially with increase in block length; thus in decoding, we quickly reach the point of diminishing returns, and so a block code of medium length is usually adequate.

The question of whether to use block or convolutional codes for a given application is one for which a general objective answer is probably impossible to give.

A rough comparison is given by McEliece [21] to show that medium length block codes give results comparable to medium constraint length convolutional codes; in this chapter, we shall show that decoding complexity is also comparable or better for block codes in many cases. Block codes have an additional advantage for schemes involving transmission of relatively short packets of data, for which they can be used without penalty, while convolutional codes involve the extra overhead of flushing out the encoder, with consequent lowering in rate and loss of performance.

We analyse various decoding schemes in terms of both full minimum distance hard decision decoding and bounded soft decision decoding. In practice, we are more interested in bounded soft decision decoding, but complete hard decision decoding is both interesting in theory and easier to analyse. For the schemes we discuss, the complexity for each strategy is the same, though for different reasons for different algorithms; we discuss the point later.

Our results will also be applicable to the McEliece public key cryptosystem [76]. For parameters n, k and t, the cryptosystem has as private key a $k \times n$ generator matrix G' for a t-error-correcting Goppa code, an $n \times n$ permutation matrix P, and a $k \times k$ non-singular matrix S. The public key is the $k \times n$ matrix $G = PG'S$. The messages are k-dimensional vectors over $GF(2)$. To encrypt a message m, we form $c = mG + e$, where e is a randomly chosen n-dimensional vector over $GF(2)$ with weight at most t. To decode, we form $c' = cP^{-1}$, apply the algebraic decoding algorithm for the Goppa code to find m' such that $d(m'G, c') \leq t$, and then we have $m = m'S^{-1}$. To crack the system, we apparently have to use a procedure capable of bounded distance decoding for any linear code, and so more efficient ways of doing this are of interest. Note, however, that in cryptanalysis the demands are less stringent: we need only show a probabilistic algorithm that works a significant fraction of the time to say that the cryptosystem is broken, and this fact makes matters much easier. Indeed, such versions of the information set decoding result given in Section 2.5 are already known for the McEliece cryptosystem [77].

In Section 2.2, we discuss and summarize some results from the application of the theory of NP-completeness to coding. In Section 2.3, we discuss non-algebraic decoding methods; we use a simple unified structure to compare and contrast the ideas behind the various methods. We analyse the asymptotic performance of some of these algorithms and suggest a way of combining some of the advantages of two different types of algorithms. Finally, application of these methods to convolutional codes is discussed.

Summary of Original Contributions

This chapter contains much discussion of previous approaches to the general decoding problem. Where we are aware of prior work by other researchers, we have given the appropriate citations. Anything not so cited should be taken as original.

In particular, the derivation of the complexity of the information set decoding algorithm is original and central. The derivations of the complexity of the other algorithms is original in most cases; the bounds for the zero neighbours algorithm and the systematic coset search in the case of binary hard decision decoding existed as estimates in the original papers. The results on complexity for bounded soft decision decoding and for decoding over higher symbols fields are original. So also are two results used in the derivation of these bounds that are of independent interest: the result in Appendix A on the weight distribution of most linear codes, and the result in Theorem 2.10 that virtually no linear (n,k) codes contain *any* k-tuple with column rank significantly less than k.

The section on the Continued Division approach is original in its entirety.

In addition, we have provided a framework in which the various algorithms can be combined and in which the relative merits of each can be compared, and we have contributed some elementary results in NP-completeness.

2.2 NP-Completeness in Coding

2.2.1 Introduction

The result of Berlekamp, McEliece and van Tilborg [31] that the complete decoding problem for linear codes is NP-complete has been influential in shaping attitudes to the problem. Many have interpreted the result to mean that the only general algorithm is exhaustive search: Berlekamp *et al.* state that "the discovery of an algorithm that runs significantly faster than this would be an important achievement;" Bassalygo *et al.* give the same interpretation [22]. In our discussion later, we will see that a great reduction in complexity is possible, though no *polynomial time* algorithm is known; thus it all depends on how we interpret the word "significantly" in this context.

In this section, we discuss and interpret related results from the theory of NP-completeness. Most of the section is a summary of known (though perhaps not widely known) results and discussion of how the results fit into the current state of knowledge about complexity classes, but there are some new observations here. One is that the optimization problem for general linear codes is no harder than the decision problem: a solution to the decision problem can be converted into a solution to the optimization problem in polynomial time. The transformation is in fact just the *step-by-step* decoding algorithm first suggested by Prange around 1960 [57]. Another result is that bounded distance decoding — hard or soft decision — is almost certainly *not* NP-complete provided we already know the guaranteed error-correcting power of the code. Also, we show that there is almost certainly no algorithm to verify that a word is a coset leader, and that similarly, there is almost certainly no general procedure whose correctness is verifiable in polynomial time that allows decoding in polynomial time.

2.2.2 Background and Terminology

A precise development of the theory of NP-completeness is given by Garey & Johnson [75]. Here we give a basic heuristic interpretation of the main points. The class P is the set of computational decision problems which can be solved by an algorithm whose running time is upper bounded by a polynomial in the length of the input. The class NP consists of those decision problems which can be solved by a *non-deterministic* algorithm in polynomial time. A non-deterministic algorithm is said to solve a decision problem if, when the answer is "yes," there is a guess which when appended to the input causes the algorithm to halt and report "yes," while if the answer is "no," there is no guess that will cause the algorithm to do this. For the class co-NP, we take the same definition as for NP and reverse the roles of "yes" and "no."

Many problems in NP can be shown [75] to have the property that if a polynomial time algorithm exists for them, a polynomial time algorithm exists for *every* problem in NP. These problems are thus as 'hard' as any problem in NP, and are called the NP-complete problems. As a large amount of effort has been expended in trying to find polynomial time solutions to many problems in NP without success, showing that a problem is NP-complete is taken as strong evidence that there is no polynomial time algorithm for it.

This last statement is equivalent to the statement that $\mathrm{P} \neq \mathrm{NP}$, a famous conjecture that, although almost universally believed, seems unlikely to be proved in the near future. Another such conjecture is that $\mathrm{NP} \neq \mathrm{co}-\mathrm{NP}$. It is known that $\mathrm{P} \subseteq \mathrm{NP} \cap \mathrm{co}-\mathrm{NP}$, but the question of whether the inclusion is proper is another difficult open question.

2.2.3 The Complete Decoding Problem

The original paper by Berlekamp *et al.* showed that the following decision problem is NP-complete:

Input: A binary matrix A, a binary vector y, and a non-negative integer w.

Property: There exists a vector x of Hamming weight $\leq w$ such that $xA = y$.

This corresponds to the problem of determining whether the coset leader has weight $\leq w$ given the syndrome and the parity check matrix. The corresponding optimization problem is: given A and y, find the vector x of minimum weight such that $xA = y$. Clearly a polynomial time solution to this problem implies a polynomial time solution to the decision problem: we can find x, compute its weight, and compare to w. Thus the optimization problem cannot be easier than the decision problem. It is perhaps not obvious that it cannot be harder either. However, an algorithm to solve the decision problem can be converted in polynomial time into an algorithm solving the optimization problem via a technique called *step-by-step decoding* developed by Prange around 1960 [57].

Step-by-step decoding works as follows. (We take the binary case; the generalization to $GF(q)$ is straightforward.) We order the n-tuples lexicographically: $(a_1 \ldots a_n)$ precedes $(b_1, \ldots b_n)$ if $a_j = 1$ and $b_j = 0$ and the two words agree up to location $j - 1$. We are assumed to have a table that, given any n-tuple, can provide the weight of the coset leader of the coset in which the n-tuple lies. Given the received word, we find the weight of the coset leader. We change the first component and find the weight of the new coset leader. If we obtain a lower weight, we accept the change. We apply the same procedure to the second, third, and all components until a word results that is in the code. Step-by-step decoding always results in a codeword, and the corresponding error pattern has minimum weight in its coset and precedes all other minimum weight words in its coset. The reason for this is simple: if the coset leader has its first non-zero element in position j, we cannot accept any change before position j (or else we could find a minimum weight error pattern preceding the coset leader) and we do accept a change at j; the proof then follows by induction. The table is not required as we can by the previous assumption solve the decision problem in polynomial time.

The next best thing to a polynomial time algorithm for solving the complete decoding problem would be an algorithm that could be derived for a given code with a certain (possibly large) amount of preprocessing, which would then allow us to decode any syndrome with that code in polynomial time. In our later discussion, we implicitly assume that this type of algorithm is being used. We have the following result:

Theorem 2.1 *Linear codes do not in general have a* polynomial-time verifiable *decoding witness that allows complete decoding unless* NP = co − NP.

Proof: Suppose there is such a witness. Then given the general decoding algorithm, we could guess the decoding witness, verify it in polynomial time, and answer the decision question whatever the answer. This would mean that the general decoding problem would be in co-NP; however, it is well known [75] that if any NP-complete problem is in co-NP, then NP = co − NP. ▶

A much stronger result has been obtained recently by Bruck [106]: complete decoding with preprocessing cannot be done in polynomial time in general unless the polynomial hierarchy PH collapses to $\sum_2^P = NP^{NP}$ [75], which is thought to be very unlikely. (However, note that the proviso is different to that of Theorem 2.1.)

Along the same lines as Theorem 2.1, we have the following elementary result:

Theorem 2.2 *Given a binary matrix G and a binary vector w, there is no polynomial time algorithm to verify that w is a coset leader in the code generated by G unless* NP = co − NP.

Proof: Given such an algorithm we can answer the complete decoding problem in polynomial time using a non-deterministic algorithm, regardless of the answer. The appropriate guess is the coset leader; we verify in polynomial time that it is the coset leader, take its weight and compare with w, reporting "yes" or "no" as appropriate. Then the complete decoding problem would be NP-complete and in co-NP, which would imply that NP = co − NP. ▶

Decoding strategies can be divided into four main categories, formed from combinations of full or bounded distance decoding with hard or soft decision. We know from [31] that full hard decision decoding is NP-complete. This is a special case of full soft decision decoding, so it follows that that problem is also NP-complete.[2] However,

[2]A more complete discussion of this point is given by Fang *et al.* [72].

bounded distance decoding, hard or soft, is different. In this case, we assume that we know the guaranteed error-correcting power of the code and will decode only to this limit. We find that the bounded distance decoding problems are in $\mathrm{NP} \cap \mathrm{co}-\mathrm{NP}$.

Theorem 2.3 *The following problems are in* $\mathrm{NP} \cap \mathrm{co}-\mathrm{NP}$*: Input: Integers* t *and* w *with* $w \leq t$*, a binary matrix* G *such that the code* $\mathcal{C}$ *generated by* G *has minimum distance at least* $2t+1$*, a binary* n*-tuple* y *such that* y *is hard (resp. soft) distance at most* t *from some codeword.*
Property: There exists an n*-tuple* x *of weight at most* w *such that* $x - y \in \mathcal{C}$*.*

Proof: An appropriate guess for the non-deterministic algorithm is the least weight word x in the same coset as y. It is easy to check that $x - y \in \mathcal{C}$ in polynomial time. We are guaranteed that x has weight at most t; we are also guaranteed that $\mathcal{C}$ has minimum distance at least $2t+1$, so x is guaranteed to be a coset leader. We then take the weight of x, compare to w, and report "yes" or "no" as appropriate. ▶
The fact that a decision problem is in $\mathrm{NP} \cap \mathrm{co}-\mathrm{NP}$ is taken as strong evidence that it is not NP-complete. It does not follow that it is in P: in fact, it is known [75] that if P≠NP, then there are problems in NP that are neither NP-complete nor in P. We have also mentioned that $\mathrm{P} \subseteq \mathrm{NP} \cap \mathrm{co}-\mathrm{NP}$, and the question of whether the inclusion is proper is a famous open problem.

Note, however, that if the bounded distance decoding problem is in P, then the modified problem in which we are not guaranteed that y is at distance at most t from a codeword is also in P. It is not sufficient to use the original algorithm: if y is at distance greater than t from the code, the answer should always be "no," but it is possible for the algorithm to give a false positive or to go into a continuous loop. Let $p(n)$ be the polynomial that bounds the running time of our original algorithm. We write a new algorithm that simulates the original algorithm for $p(n)$ steps. If y is at distance more than t from the nearest codeword, then after $p(n)$ steps, the original algorithm has either halted giving the wrong answer, or is still running. If it is still running after $p(n)$ steps, we conclude that y must be at distance greater than t (hence greater than w) from the code. If it has halted in less than $p(n)$ steps, we can check the answer given using the step-by-step decoding algorithm; if it is incorrect, we can again assume that y is at distance greater than t from the code.

2.2.4 Random Algorithms

Many results have been developed relating to *randomized* algorithms. We discuss the implications for the existence of randomized algorithms of a proof that a problem is NP-complete. Our discussion follows that of Johnson [80].

The randomized algorithms use the notion of a *Probabilistic Turing Machine*, defined to be a non-deterministic Turing Machine in which a fork in the tree of possible computation paths is interpreted as a random choice between two equiprobable alternatives. If the PTM is constrained so that for some fixed $\epsilon > 0$, the proportion of leaves with the correct answer is always greater than $1/2 + \epsilon$, we define the class BPP (for Bounded Probabilistic Polynomial). Within this class, we can achieve arbitrarily low error probability with a bounded number of iterations. It is thought that BPP and NP are incomparable in the sense that neither class contains the other. If the computation trees are required to have no "yes" leaves when the answer is "no," we obtain the class R (for Random polynomial time). It does not appear that R=co-R, and so R$\cap$co-R is probably a proper subset of R. This intersection class corresponds precisely to the notion of the class of problems solvable by polynomial time Monte Carlo algorithms. It can also be shown [80] to correspond to the set of problems solvable with zero probability of error in *expected* polynomial time for each instance, a class known as ZPP. In summary, we have

$$\mathrm{P} \subseteq \mathrm{ZPP} = \mathrm{R} \cap \text{co-R} \subseteq \mathrm{R} \subseteq \left\{ \begin{matrix} \mathrm{NP} \\ \mathrm{BPP} \end{matrix} \right\}$$

and it is conjectured that all the inclusions are proper.

Now, it is well known that an NP-complete problem cannot be in P unless P = NP, and cannot be in co-NP unless NP = co – NP; it is also true [80] that an NP-complete problem cannot be in R unless NP = R and cannot be in ZPP unless NP = ZPP. None of these consequences is currently considered likely; in particular, if NP = R, there are two surprising results. One is that the polynomial time hierarchy PH collapses into $\sum_2^p = NP^{NP}$ [80]. The other [80, 94] is that if the factoring and discrete logarithm problems are indeed hard, as is widely believed, then every problem in R is solvable in time $o(2^{n^\epsilon})$ for every $\epsilon > 0$, a property that is not expected to hold for NP.

Our conclusion for coding is that the complete decoding problem almost certainly cannot be solved with zero probability of error in expected polynomial time, or solvable in random polynomial time; it may, however, be in the bounded probabilistic polynomial time class BPP.

2.2.5 Other NP-Completeness Results

Finally, we state without discussion some other NP-completeness results that have been derived by various researchers.

The following problems are NP-complete (we are given the generator matrix G of a linear code $\mathcal{C}$ in each case):

1. [31] Given an integer w, is there a codeword of weight w in $\mathcal{C}$?

2. [82] Given integers k and w, with $k \geq 2$, is there a non-zero codeword of weight not more than w and not a multiple of k? (For $k = 2$, this is the problem of finding the minimum odd-weight codeword.)

3. [82] Given a positive integer w, is there a codeword of weight $\geq w$?

4. [82] Given positive integers w_1 and w_2 with $w_1 < w_2$, is there a codeword of $\mathcal{C}$ with weight in the range w_1 to w_2 inclusive?

5. [72] Given integers w and l, is there a non-zero codeword with weight not greater than w and a non-zero component in location l?

6. [72] Given integers w, p, and j, is there a non-zero codeword with weight not greater than w and a fraction $p/(p+1)$ of its non-zero components in the first l locations?

7. [72] Given a vector y and an integer w, is there a word of weight $\leq w$ in the same coset as y with non-zero components in the first $\lfloor wp/(p+1) \rfloor$ locations?

The last problem implies that the complete decoding problem remains NP-complete even if a fraction of $p/(p+1)$ of the errors could be guessed.

2.3 General Decoding Methods

2.3.1 Model of General Decoding Algorithm

Before discussing specific algorithms, we suggest a framework into which most of the algorithms will fit. This provides a way of comparing the various approaches and suggests a direction for deriving improved algorithms. The algorithm is chosen to be the simplest one possible while still retaining the power of the algorithms it describes. The operations involved are all very simple to implement.

In the decoding methods we discuss, we take the received word and store it in an 'operand' register. The contents of this register are then modified by adding codewords according to the rules of the particular algorithm. Two rules are allowed in deciding whether to add a codeword: we can add a codeword to the word in the operand register if the weight of the new contents is lower than before, and we can decide whether to add a codeword by examining the symbol in a certain location and branching on the value.

Most algorithms have one of two basic strategies, each based on a different heuristic. One strategy, used by the majority of currently known algorithms, is to map directly from the received word to the coset leader. The other attempts to reduce the weight of the word in the operand register progressively.

Two basic heuristics are used in the algorithms. One type of algorithm, which we label *progressive*, works by progressively reducing the weight of the operand word until it is relatively low. Then the codeword equal to the difference between the operand word and the coset leader must also have low weight. It is then sufficient to search systematically through all codewords of low weight. For typical codes, the codewords have a binomial distribution (see Appendix A) and a search through the low weight codewords corresponds to a search through the tail of the distribution with, consequently, a much lower complexity than a full search through all codewords.

The second type of algorithm works by exploiting the redundancy of the code. Suppose we know all the errors in a set of k independent symbols — an *information set.* Then we can construct the transmitted codeword, and by comparing with the received word, we find the coset leader. This holds no matter what pattern of errors lies outside the information set. Thus knowing the errors in one information set is sufficient to decode a large number of error patterns, and this is where the procedure derives its efficiency.

2.4 Progressive Algorithms

2.4.1 Projecting Set Decoding

This algorithm, suggested by Hwang [63], depends in complexity on the number of codewords in the *projecting set* of the code. This is defined to be the minimum set of non-zero codewords $\mathcal{C}_p$ such that every non-zero codeword outside $\mathcal{C}_p$ can be expressed as the sum of *disjoint* codewords from $\mathcal{C}_p$ (*i.e.,* codewords with disjoint supports). The reason why this set of codewords is sufficient is simple: if the operand word is not the coset leader, then we can add a codeword to get a weight reduction. The weight change obtained by adding a codeword that is not in the projecting set is equal to the sum of the weight changes obtained by adding the constituent codewords from the projecting set. It is thus impossible to achieve a weight reduction by adding a codeword that is not in the projecting set unless at least one codeword in the projecting set also causes a weight reduction. Our algorithm is to subtract all the codewords in the projecting set from the operand word, accepting the operation if it results in a weight reduction. The algorithm terminates in a coset leader when none of the words in the projecting set provide a weight reduction.

The following theorem, again due to Hwang [63], shows that the projecting set is a set of low weight codewords.

Theorem 2.4 (Hwang) *Let C be a binary linear (n, k, d) code. The projecting set* *of C contains all codewords of weight $2d-1$ or less, and no codeword of weight greater than $n-k+1$.*

Proof: The first property holds because a codeword that is not in the projecting set is the sum of at least two disjoint codewords, each of weight at least d, and hence must have weight at least $2d$. For the second property, we have the following argument. If a codeword has weight w, the corresponding w columns of the parity check matrix must sum to zero. If any subset of these columns sums to zero, then the columns outside the subset also sum to zero, and the codeword can be expressed as the sum of two disjoint codewords. Thus if a codeword of weight w is in the projecting set, the corresponding w columns of the parity check matrix must be such that any subset of $w-1$ or fewer columns must be linearly independent. If $w \geq n-k+2$, this means that a codeword of weight w that is in the projecting set must give sets of at least $w-1 \geq n-k+1$ columns that are linearly independent. The column rank of the matrix would then be greater than $n-k$, the row rank, which is impossible. ▶

An interesting corollary is that a binary linear code in which $2d-1 \geq n-k+2$ contains no codewords of weight w for $n-k+2 \leq w \leq 2d-1$.

More generally, the first part of this theorem holds for non-binary codes also, though the second does not. The algorithm also extends directly to perform *full* (maximum likelihood) soft decision decoding with the same number of codewords. It remains to be shown that the number of iterations required is bounded by a polynomial in n.

Table 1 [63] gives the number of projecting set codewords for various codes. Only in one case does it happen that the number codewords is less than the number of syndromes (it is always less than the total number of codewords); however, the fact that we achieve *full* soft decision decoding makes it interesting. (We shall see in Section 2.10 that full soft decision decoding can always be achieved with complexity equal to the number of syndromes).

Using our bound on the number of codewords of a given weight in the average linear code (see Appendix A), we shall now derive the asymptotic bounds on performance:

(n,k,d)	$\lvert C_s\rvert$	2^k	2^{n-k}
(15,10,4)	385	1024	32
(17,9,5)	340	512	256
(23,12,7)	3335	4096	2048
(23,11,8)	1794	2048	4096
(31,25,4)	23653	33554432	64

Table 1: Size of the Projecting Set for Some Codes

Theorem 2.5 *The number of codewords in the projective set, $|C_s|$ is bounded by*

$$2^{n[2H_2(H_2^{-1}(1-R))-(1-R)]+o(n)} \le |C_s| \le 2^{n[\min(R,H_2(1-R)-(1-R)]+o(n)}.$$

The lower bound uses Equations A-1 and A-2 of Appendix A to find the number of codewords of weight less than $2d$. For the upper bound, we take the number of codewords of weight $n-k+1$ or less. If $R < 1/2$, Theorem A-1 of Appendix A shows that only an asymptotically insignificant proportion of the codewords have weight greater than $n(1-R)+o(n)$, so the projecting set consists of almost all the codewords. If $R > 1/2$, we have $\sum_{i\le n-k+1} A(i) = 2^{n[H_2(1-R)-(1-R)]+o(n)}$ for most codes.

These bounds are plotted in Fig. 2.1. Although the size of the projecting set is significantly smaller than the number of codewords for $R > 1/2$, we will see in Section 2.10 that it is possible to achieve full minimum distance soft decision decoding of any linear block code with complexity $q^{-n(1-R)}$ by using a trellis. Thus the upper bound in Fig. 2.1 is unsatisfactory, and the algorithm's efficiency is suspect.

Hwang suggests a second algorithm in which a subset of low weight codewords of the projecting set which contains a basis of the code is used. He gives the results of simulations of the performance of this suboptimum algorithm, and conjectures that with the proper choice of the subset, we can achieve a great reduction in complexity with only a small loss in performance.

In the context of the model in Section 2.3.1, we are using the 'progressive' feature extensively, while not making any use of the symbol-decision option.

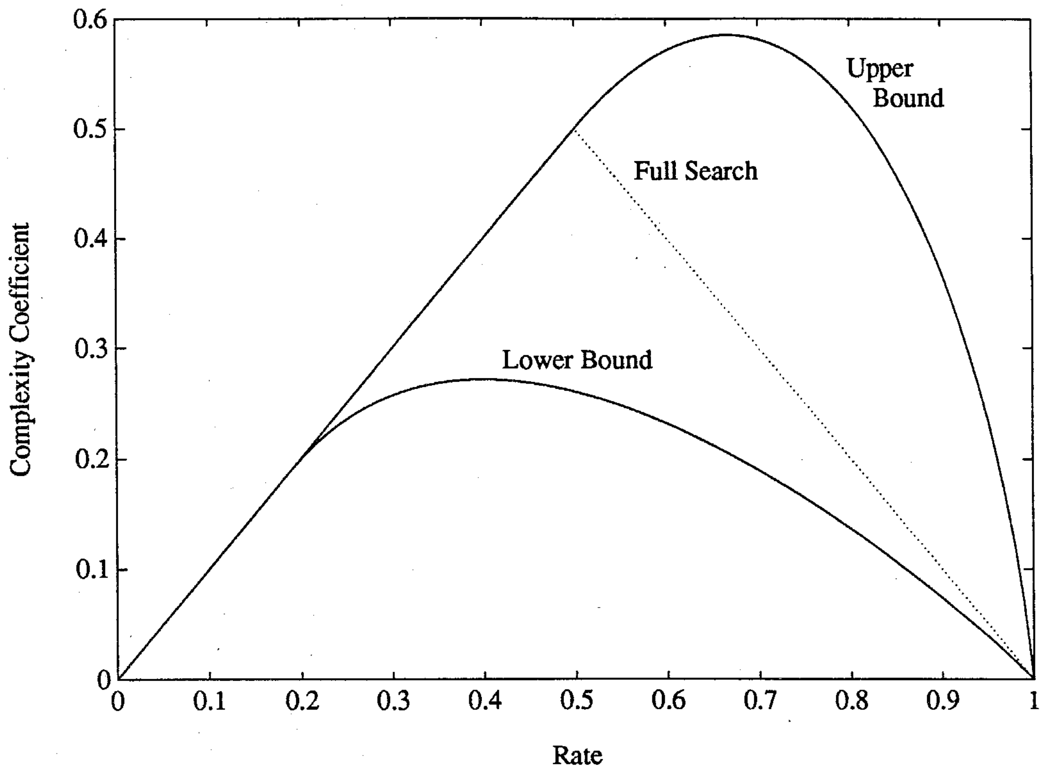


Figure 2.1: Asymptotic Bounds for Projecting Set Decoding

2.4.2 Zero Neighbours Algorithm

The Zero Neighbours algorithm, proposed relatively recently by Levitin and Hartmann [32], is a more advanced version of the projecting set algorithm, and is optimal for the pure progressive strategy, in which we form a set of codewords with the property that for *any* n-tuple that is not a coset leader, subtraction of one of the codewords in the set will produce a word of lower weight.

We need the following notation. For a linear code $\mathcal{C}$, the *domain* of a codeword $\mathbf{c}$, denoted $\mathcal{D}(\mathbf{c})$, consists of those n-tuples that are at least as close to $\mathbf{c}$ as to any other codeword of $\mathcal{C}$. (We will call any word in the domain of the zero codeword a coset leader, so some of the cosets will have multiple 'joint' coset leaders.) The *domain frame* of $\mathcal{C}$, denoted $\mathcal{G}(\mathbf{c})$, consists of all those n-tuples that, while not contained in the domain of $\mathcal{C}$, are at distance 1 from at least one n-tuple in the domain. The set of *zero neighbours*, denoted $\mathcal{N}(\mathcal{C})$, is the smallest cardinality set of codewords so that every n-tuple in the domain frame of the zero codeword is in the domain of at least one zero neighbour. Thus the zero neighbours form a minimum covering of the 'border' around the coset leaders.

The crucial result is the following [32]: if $\mathbf{w}$ is not a coset leader, then for at least one $\mathbf{c} \in \mathcal{N}(\mathcal{C})$, the word $\mathbf{w} - \mathbf{c}$ has lower weight than $\mathbf{w}$. Thus to decode, we need only the set of zero neighbours, and the procedure is the same as that in the case of projecting set decoding. As in that case, the number of iterations cannot be greater than n; the number of codewords required is in general much smaller.

To show that the main property holds, let $\mathbf{w}$ be an n-tuple that is not a coset leader. Consider a chain of descendants of $\mathbf{w}$:

$$\mathbf{w}_0 = \mathbf{0},\ \mathbf{w}_1, \ldots, \mathbf{w}_{\mathrm{wt}(\mathbf{w})-1},\ \mathbf{w}_{\mathrm{wt}(\mathbf{w})} = \mathbf{w},$$

such that $\mathbf{w}_{i-1}$ is an immediate descendant of $\mathbf{w}_i$. Now by assumption $\mathbf{w}_{\mathrm{wt}(\mathbf{w})}$ is not in $\mathcal{D}(\mathbf{0})$, while $\mathbf{w}_0$ *is* in $\mathcal{D}(\mathbf{0})$. There must therefore be two consecutive descendants $\mathbf{w}_i$ and $\mathbf{w}_{i+1}$ such that $\mathbf{w}_i \in \mathcal{D}(\mathbf{0})$ and $\mathbf{w}_{i+1} \notin \mathcal{D}(\mathbf{0})$. Then $\mathbf{w}_{i+1}$ is a word that is at distance 1 from a word in $\mathcal{D}(\mathbf{0})$, while not itself in $\mathcal{D}(\mathbf{0})$; thus it is in the domain frame $\mathcal{G}(\mathbf{0})$. It must be in the domain of at least one zero neighbour $\mathbf{c}_{\mathcal{N}}$, and

$d(\mathbf{0}, \mathbf{w}_{i+1}) > d(\mathbf{c}_{\mathcal{N}}, \mathbf{w}_{i+1})$. We then have

$$\begin{aligned} d(\mathbf{w}, \mathbf{c}_{\mathcal{N}}) &\leq d(\mathbf{w}, \mathbf{x}_{i+1}) + d(\mathbf{x}_{i+1}, \mathbf{c}_{\mathcal{N}}) && (2.1) \\ &< d((\mathbf{w}, \mathbf{x}_{i+1}) + d(\mathbf{x}_{i+1}, \mathbf{0}) && (2.2) \\ &= d(\mathbf{w}, \mathbf{0}). && (2.3) \end{aligned}$$

So $\mathrm{wt}(\mathbf{w} - \mathbf{c}_{\mathcal{N}}) < \mathrm{wt}(\mathbf{w})$ for at least one $\mathbf{c}_{\mathcal{N}}$ as claimed.

The concept of zero neighbours is an exact formulation of the heuristic notion of 'low weight codewords' we discussed in Section 2.3. Some properties of the set are discussed in [32]; the main two to illustrate the low weight nature of the set are (i) every minimum non-zero weight codeword is in the set of zero neighbours, and (ii) no codeword of weight greater than $2r_{\mathcal{C}} + 1$ is a zero neighbour, where $r_{\mathcal{C}}$ is the covering radius of the code. Property (ii) holds because a word in the domain frame of the zero codeword cannot have weight greater than $r_{\mathcal{C}} + 1$, and, if in the domain of $\mathbf{c}_{\mathcal{N}}$, cannot be at distance more than $r_{\mathcal{C}}$ from $\mathbf{c}_{\mathcal{N}}$. For property (i), let $\mathbf{x}$ be a descendant of $\mathbf{c}$ of weight $\lceil (\mathrm{wt}(\mathbf{c}) + 1)/2 \rceil$ (*i.e.*, $\mathbf{x}$ is in the domain frame of the zero codeword). Now for any other non-zero codeword $\mathbf{c}'$, we have $\mathrm{wt}(\mathbf{c}) \leq d(\mathbf{c}, \mathbf{c}') \leq d(\mathbf{c}, \mathbf{x}) + d(\mathbf{x}, \mathbf{c}')$. So $d(\mathbf{x}, \mathbf{c}') \geq \mathrm{wt}(\mathbf{c}) - d(\mathbf{c}, \mathbf{x}) = \mathrm{wt}(\mathbf{x}) > d(\mathbf{c}, \mathbf{x})$. So $\mathbf{x}$ is in the domain frame of the zero codeword and is closer to $\mathbf{c}$ than to any other codeword. Thus $\mathbf{c}$ is a zero neighbour.

Levitin and Hartmann derive an asymptotic version of the upper bound given by property (ii), although as they did not have any of the results in the Appendix, their derivation really provides only an estimate. Their result, which we obtain directly from the results in the Appendices, is

$$|\mathcal{N}(\mathcal{C})| \leq 2^{n[H_2(2H_2^{-1}(1-R))-(1-R)]+o(n)}$$

for virtually all binary linear codes for all R such that $2H_2^{-1}(1-R) < 1/2$ (otherwise virtually all codewords have weight less than twice the covering radius). The upper bound to the function $\lim_{n\to\infty}(1/n)\log|\mathcal{N}(\mathcal{C})|$ is plotted in Fig. 2.2. As long as the rate is higher than 0.1887 (the solution to $H_2^{-1}(1-R) = 1/4$), we achieve a substantial reduction in the decoding complexity of full minimum distance hard decision decoding. Note that the upper bound on decoding complexity here has the same asymptotic

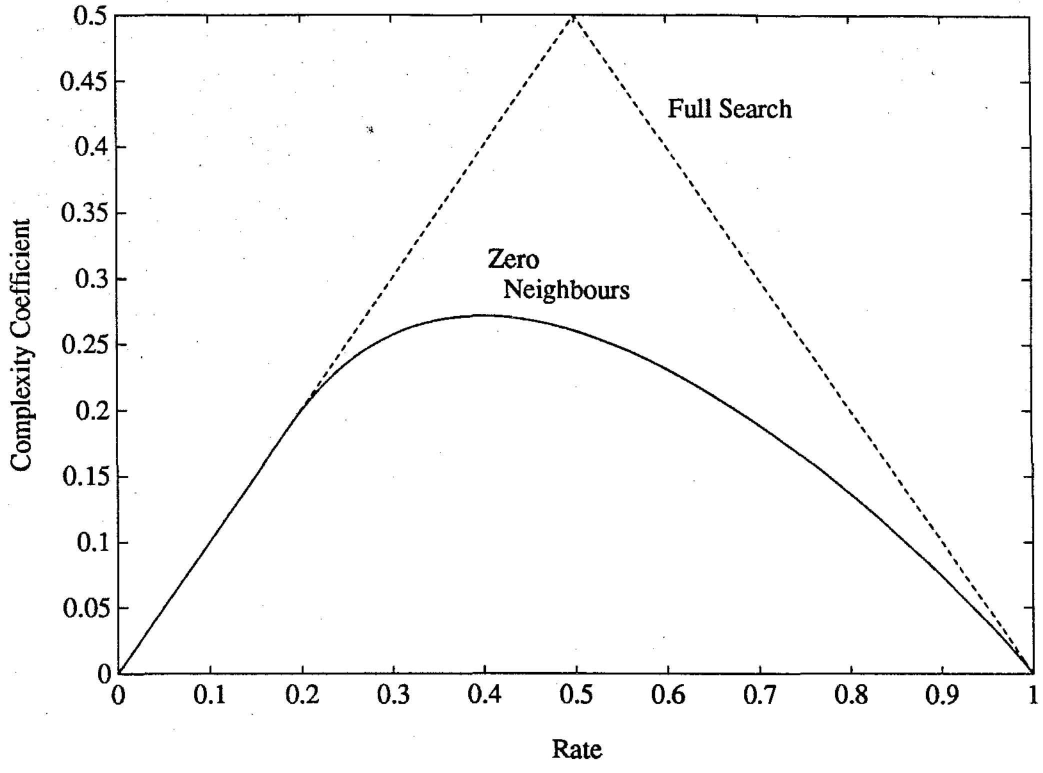


Figure 2.2: Complexity of the Zero Neighbours Algorithm

form as the lower bound for projecting set decoding, because for the average linear code, the minimum distance is asymptotically about equal to the covering radius.

The result can be extended in two further ways not considered by Levitin & Hartmann. We have:

Theorem 2.6 *The zero neighbours algorithm performs bounded soft decision decoding for virtually all binary linear codes with the same complexity as for the full hard decision case,* i.e., $F(R) = H_2(2H_2^{-1}(1-R)) - (1-R)$ *(see the plot in Fig. 2.2). The algorithm is also applicable to decoding over non-binary symbol fields, but the complexity rises to that of the trivial search algorithms as q becomes large.*

Proof: For the soft decision case, it is well known [52] that an error pattern can have at most $d-1$ hard errors if it is to be within the guaranteed soft error correcting power of the code. This is because a hard error can occur if the received bit is further from the transmitted bit than from the complement of the transmitted bit. Thus we have a contribution of more than 1/2 in the soft metric for every hard error. Overall, the bounded soft distance is $t = \lfloor (d-1)/2 \rfloor$, and we can have no more than $2t$ hard errors. Our strategy is to perform full minimum distance hard decision decoding using the regular zero neighbours algorithm. For virtually all codes, the coset leader can have weight at most $r_{\mathcal{C}}$ and the best bounded soft decision word has at most $d-1$ hard errors. The two words are at distance at most $d-1+r_{\mathcal{C}}$ apart, so we add all codewords of weight at most $d-1+r_{\mathcal{C}}$, taking as the soft decision error pattern the sum with the lowest soft weight. From our estimates of the covering radius and minimum distance, we are adding all codewords of weight at most $2nH_2^{-1}(1-R)+o(n)$. However, this is just the set we take as the set of zero neighbours, and so modifying the result from full hard decision decoding to bounded soft decision decoding takes a subexponential amount of complexity.

For the case of non-binary symbol fields, we have $\mathcal{N}(\mathcal{C}) \leq q^{n[H_q(2H_q^{-1}(1-R))-(1-R)]+o(n)}$. For the behaviour of the function $H_q^{-1}(x)$ as q becomes large, we have

$$\begin{aligned} H_q(x) &= x\,\log_q(q-1) + x\,\log_q x + (1-x)\,\log_q(1-x) \\ &= x\Big(1+\log_q(1-1/q)\Big) + H_2(x)/\log_2 q \\ &= x\Big(1-\big(1/q-1/2q^2+\cdots\big)/\ln q\Big) + H_2(x)/\log_2\text{q} \\ &= x\,(1+O(1/\log_2 q)) \\ &\to x \qquad \text{as } q\to\infty. \end{aligned}$$

Then by continuity of $H_q(x)$, we must have

$$\lim_{q\to\infty} H_q^{-1}(x) = x.$$

Assuming that $2H_q^{-1}(1-R) < (q-1)/q$, we have $\log_q(1/n)\mathcal{N}(\mathcal{C}) \leq H_q(2(1-R)+o(1)) - (1-R) = 2(1-R)-(1-R)+o(1) = 1-R+o(1)$, so the complexity of the zero neighbours algorithm approaches that of a full search. ▶

As in the case of projecting set decoding, we are employing the progressive part of our general scheme extensively (and have an optimal solution for this case). The burden of having to be able to provide progressive weight reduction from *any* word in any coset to the coset leader is quite high, however; in fact, it is so high that we will see later that the zero neighbours algorithm is far from being optimal. An algorithm exploiting the full power of the general model would only have to provide progressive weight reduction from some word or words of the coset to the coset leader.

2.5 Information Set Algorithms

Information set decoding was first suggested by Prange [33] for decoding cyclic codes and has been extensively examined and modified by many other researchers [34–47]. As we discussed in Section 2.3.1, the basic idea is that knowing the errors in an information set is sufficient to find all the errors.

An interesting result is given by Mandelbaum [46].

Theorem 2.7 (Mandelbaum) *For any linear code C and any coset leader or joint coset leader w of C, there is at least one information set in C that is disjoint from w. Thus a pure information set algorithm is always sufficient to achieve full minimum distance decoding.*

Proof: Suppose the complement of the support of w contains less than k independent symbols. We set all these independent symbols to zero; this must mean that all symbols not in the support of w are zero. Now we must be able to find at least one symbol in the support of w that is independent of all symbols in the complement of w by assumption, and we set this symbol to zero. We now have a word in the same coset as w but with lower weight, contradicting the assumption that w is a coset leader. ▶

Many embellishments of this basic idea exist. In permutation decoding, [34], sets of k independent positions are obtained by using the automorphism group of the code. If the code is in systematic form, the first k symbols form an information set, and so do all valid permutations thereof. This deals with the problem of how the information sets are to be generated, but an exact analysis of complexity, even for correction of a very small number of errors, is very cumbersome [35, 36]. Nevertheless, this method has been the focus of much attention [34–36, 44].

Another modification of the basic idea is to drop the condition that the information set is error-free. We then search systematically through precomputed patterns of information set errors. These patterns are called "covering polynomials" by Kasami [37], who applies the method to cyclic codes, using n information sets and concentrating the computational effort in the use of covering polynomials. The obvious idea of using general information sets with covering polynomials has been suggested many times, for example by Dmitriev [38] and Evseev [39].

The approach we refer to as generalized information set decoding uses k-tuples which are not necessarily information sets. If the k-tuple has fewer than k independent symbols, we augment the set by adding more symbols till there are k independent symbols in the set. All possible patterns in the augmented symbols are then searched. This approach is equivalent to both "decoding with multipliers" [40] and combined information set and covering polynomial methods.

Despite the great amount of interest in algorithms based on the information set idea, no precise estimates of the decoding complexity have been produced. Clark & Cain [41] discuss some reasons why the problem is difficult. First, it is related to a long-standing unsolved problem in combinatorics, the (n, k, t) covering problem [42]. Given a set of n objects, we seek the minimum number of subsets of cardinality k, such that any subset of cardinality t is contained in at least one of the subsets of cardinality k. To avoid confusing the value of k in the (n, k, t) covering problem with the dimension of the code, we refer to the problem as the (n, l, t) covering problem. We refer to the minimum number of subsets required as the (n, l, t) covering coefficient, denoted by $b(n, l, t)$. In our problem, the subsets of cardinality t are the error patterns, and the subsets of cardinality l are the sets of parity positions, so that in our notation $l = n - k$. A t-tuple which is covered by an $(n - k)$-tuple is said to be "trapped" by the corresponding k-tuple. Our problem is thus to find an approximation for the $(n, n - k, t)$ covering coefficient. However, the problem is more difficult for two reasons: the $k-$tuples selected must represent an information set, and (for complete decoding) we must decode all patterns which are coset leaders, not just all patterns of a fixed weight or less. Despite these difficulties, we present a solution which is logarithmically accurate for virtually all linear codes.

First we derive a logarithmically accurate expression for $b(n, l, t)$, the (n, l, t) covering coefficient. We have the following result:[3]

Theorem 2.8 *Let R and ρ be constants such that $0 < \rho < 1 - R < 1$. Then*

$$\lim_{n\to\infty} \frac{1}{n} \log b(n, \lfloor n(1-R) \rfloor, \lfloor n\rho \rfloor) = H_2(\rho) - (1-R) H_2\big(\rho/(1-R)\big)$$

where $H_2(x) = -x \log_2 x - (1-x) \log_2(1-x)$ is the binary entropy function.

Proof: A lower bound for $b(n, l, t)$ is easy to obtain. We must "trap" all t-tuples. Each selected k-tuple can trap $\binom{n-k}{t}$ t—tuples. Even in the most optimistic scenario where each t-tuple is trapped by exactly one k-tuple, we still need

$$\binom{n}{t} \Big/ \binom{n-k}{t}$$

k-tuples. Using the relation [10]

$$2^{nH_2(\lambda) - O(\log n)} \le \binom{n}{\lambda n} \le 2^{nH_2(\lambda)}$$

for $0 < \lambda < 1$, we have $\binom{n}{\rho n} \ge 2^{nH_2(\rho) - o(n)}$ and $\binom{n(1-R)}{n\rho} \le 2^{n(1-R)H_2(\rho/(1-R))}$ and thus

$$b(n, \lfloor n(1-R) \rfloor, \lfloor n\rho \rfloor) \ge \binom{n}{n\rho} \Big/ \binom{n(1-R)}{n\rho} \ge 2^{n\left[H_2(1-R) - (1-R)H_2\left(\rho/(1-R)\right)\right] + o(n)}.$$

For the upper bound, we adopt the following argument. We select a large number $f(n, k, t)$ of k-tuples independently and at random. The probability that a given t-tuple is *not* trapped is

$$\left[1 - \binom{n-t}{k} \Big/ \binom{n}{k}\right]^{f(n,k,t)}$$

because for each choice of k-tuple, there are $\binom{n-t}{k}$ "good" k-tuples, out of a total of $\binom{n}{k}$ k-tuples. The expected number of t-tuples not trapped is

$$\binom{n}{t} \left[1 - \binom{n-t}{k} \Big/ \binom{n}{k}\right]^{f(n,k,t)}.$$

[3]We have recently found a similar, though not identical, result for the non-asymptotic case in Erdös & Spencer [101]; our results have been derived independently.

Now let

$$f(n,k,t) = \frac{\binom{n}{k}}{\binom{n-t}{k}} g(n,k,t)$$

for some function $g(n,k,t)$. Then the expected number of t–tuples not trapped is

$$\binom{n}{t}\left[1 - \binom{n-t}{k}\Big/\binom{n}{k}\right]^{\frac{\binom{n}{k}}{\binom{n-t}{k}} g(n,k,t)}.$$

Using the relation

$$\lim_{x\to\infty}(1 - 1/x)^x = e^{-1},$$

we see that the above expression tends towards

$$\binom{n}{t} e^{-g(n,k,t)} = 2^{nH_2(t/n)+o(n)-g(n,k,t)\log\, e}.$$

Setting $g(n,k,t) > \frac{1}{\log\, e}[nH_2(t/n) + o(n)]$ now gives an expected number of t-tuples not trapped less than one. This is possible only if there is at least one set of $f(n,k,t)$ k-tuples which traps all t–tuples. Thus we need

$$f(n,k,t) = \frac{\binom{n}{k}}{\binom{n-t}{k}} \cdot cn$$

for any constant c greater than $H_2(t/n)/\log_2 e$. Using the identity $\binom{n}{t}\binom{n-t}{k} = \binom{n}{k}\binom{n-k}{t}$, we have

$$\binom{n}{k}\Big/\binom{n-t}{k} = \binom{n}{t}\Big/\binom{n-k}{t}.$$

Thus the upper bound is such that

$$b(n, n-k, t) \le f(n,k,t) = cn\binom{n}{t}\Big/\binom{n-k}{t} = 2^{n\left[H_2(\rho)-(1-R)H_2\left(\rho/(1-R)\right)\right]+o(n)},$$

which has the same form as the lower bound. ▶

To analyse the complete decoding problem for linear codes, it is necessary to have some knowledge of the covering radius of these codes. The covering radius is the weight of the highest weight coset leader of the code. The Goblick bound [48] states that

$$r \ge nH_q^{-1}(1-R) + o(n)$$

for all codes, but no general upper bound has been known until recently, when Blinovskii [17] and Levitin [102] showed independently that the Goblick bound is tight for virtually all linear codes, *i.e.*, that

$$r = nH_q^{-1}(1-R) + o(n)$$

for all but a fraction of codes that tends to zero as $n \to \infty$. We give Blinovskii's proof in Appendix B.

We seek the number of k-tuples to be selected such that any *coset leader* of the code is disjoint from some k–tuple. This number is given by the following theorem.

Theorem 2.9 *For virtually all linear (n,k) codes over $GF(q)$, the minimum number $M(C)$ of k–tuples required to ensure that each coset leader is disjoint from at least one k-tuple satisfies*

$$\frac{1}{n}\log_2 M(C) = H_2\Big(H_q^{-1}(1-R)\Big) - (1-R)H_2\left(\frac{H_q^{-1}(1-R)}{1-R}\right) + o(1).$$

Proof: An upper bound is obtained by considering the number of k–tuples necessary to trap *all* patterns of up to r errors, whether the patterns are coset leaders or not. By the definition of covering radius, this set includes all coset leaders. From Theorem 2.8, we have the upper bound

$$2^{n\left[H_2(r/n)-(1-R)H_2\left(r/n(1-R)\right)\right]+o(n)} = 2^{n\left[H_2\left(H_q^{-1}(1-R)\right)-(1-R)H_2\left(\frac{H_q^{-1}(1-R)}{1-R}\right)\right]+o(n)}.$$

For the lower bound, we note that we must trap q^{n-k} coset leaders, and that virtually all coset leaders have weight greater than $nH_q^{-1}(1-R) - o(n)$. Each k-tuple can trap no more than

$$\sum_{\rho-o(1)\le i\le\rho+o(1)} \binom{n-k}{i}(q-1)^i$$

such coset leaders. Now $q^{n-k} = \binom{n}{n\rho}(q-1)^{n\rho}\cdot q^{o(n)}$ where $\rho = H_q^{-1}(1-R)$, so the lower bound has the form

$$\binom{n}{n\rho}\Big/\binom{n(1-R)}{n\rho} = 2^{n\left[H_2\left(H_q^{-1}(1-R)\right)-(1-R)H_2\left(\frac{H_q^{-1}(1-R)}{1-R}\right)\right]+o(n)}$$

and the theorem follows. ▶

The only remaining problem is how we deal with the case when a selected k-tuple does not contain k linearly independent symbols. If a k-tuple has only $k - l$ independent symbols, we say that the k-tuple is *l-defective.* We can remedy this condition by finding l symbols from the remaining $n-k$ in such a way that the $(k+l)$-tuple has k linearly independent symbols, and then exhaustively searching through all possible error patterns in that l-tuple — these are just the covering polynomials mentioned earlier. This will cause an increase in complexity of q^l for that k-tuple. We need to show that this increase in complexity is subexponential. Given fixed R and α, with $0 < R, \alpha < 1$, we say that an $\lfloor nR \rfloor$-tuple is *seriously α-defective* if the $\lfloor nR \rfloor$-tuple contains less than $\lfloor nR(1-\alpha) \rfloor$ independent symbols. We will show that for any fixed $\alpha > 0$ and sufficiently large n, there are virtually *no* linear (n, k) codes that contain *any* k-tuple that is seriously α-defective. To do this, we employ Kolmogorov complexity. We have

Theorem 2.10 *For any fixed R and α satisfying $0 < \alpha, R < 1$, and for all sufficiently large values of n, virtually all linear $(n, \lfloor nR \rfloor)$ codes over any symbol field contain no $\lfloor nR \rfloor$-tuple with fewer than $\lfloor nR(1-\alpha) \rfloor$ independent symbols.*

Proof: Let G be the $k \times n$ generator matrix of a linear code $\mathcal{C}$. With this generator matrix we associate the string $\mathbf{s}(G)$ of length nk obtained by writing G out row by row. Each generator matrix corresponds to exactly one such string and *vice versa* (note that we do not insist that each *code* be represented by exactly one string; nor do we insist that $\dim \mathcal{C} = k$). From the key lemma on Kolmogorov complexity, the fraction of these strings with Kolmogorov complexity less than $nk - c$ is less than q^{-c}. Suppose a code contains a k-tuple that is seriously α-deficient. We can specify the generator matrix (and hence the full code string) as follows:

— specify the deficient k-tuple (taking $\log_q \binom{n}{k}$ symbols);

— write out the other $n - k$ columns in full (taking $k(n-k)$ symbols);

— write out the $k(1-\alpha)$ independent columns in the defective k-tuple (taking $k^2(1-\alpha)$ symbols);

— specify each of the remaining columns in the defective k-tuple by specifying the linear combination of the independent columns which yields it (taking $k\alpha.k(1-\alpha)$ symbols).

The total length of this program is

$$C + \log_q \binom{n}{k} \quad + k(n-k) + k^2(1-\alpha) + k\alpha k(1-\alpha)$$
$$= n^2R - n^2R^2\alpha^2 + o(n^2).$$

The fraction of such strings is thus less than $q^{-n^2R^2\alpha^2+o(n^2)}$, as required. ▶

Putting together the results of Theorems 2.8, 2.9 and 2.10, we have the following result:

Theorem 2.11 *For virtually all linear* (n,k) *codes* $\mathcal{C}$ *over* $GF(q)$*, the complexity* $M(\mathcal{C})$ *of complete minimum distance decoding using the generalized information set decoding algorithm satisfies*

$$\frac{1}{n}\log_2 M(\mathcal{C}) = H_2\Big(H_q^{-1}(1-R)\Big) - (1-R)H_2\Big(\frac{H_q^{-1}(1-R)}{1-R}\Big) + o(1).$$

By the convexity of the entropy function, we have $H_2(xy) > xH_2(y)$ for $0 < x, y < 1$, and so the function is always greater than zero, as we would expect.

The behaviour of this function versus R for the case $q = 2$ can be seen in Fig. 2.3. Clearly, it represents a huge improvement over exhaustive search procedures for any fixed rate. For $R = 1/2$, generalized information set decoding requires less than the fourth root of the number of computations required by a search through all codewords.

For bounded distance hard decision decoding, we need to decode all error patterns of weight up to t, where $t = \lfloor (d-1)/2 \rfloor$. From Equation 1.1 and Corollary A-2, the Gilbert-Varshamov bound is tight for virtually all linear codes over any symbol field, *i.e.*, $t = nH_q^{-1}(1-R)/2 + o(n)$ for most codes.

Theorem 2.12 *Bounded distance decoding using generalized information set decoding has, for virtually all linear codes, a complexity* $M(\mathcal{C})$ *satisfying*

$$\frac{1}{n}\log_2 M(\mathcal{C}) = H_2\Big(H_q^{-1}(1-R)/2\Big) - (1-R)H_2\Big(\frac{H_q^{-1}(1-R)}{2(1-R)}\Big) + o(1).$$

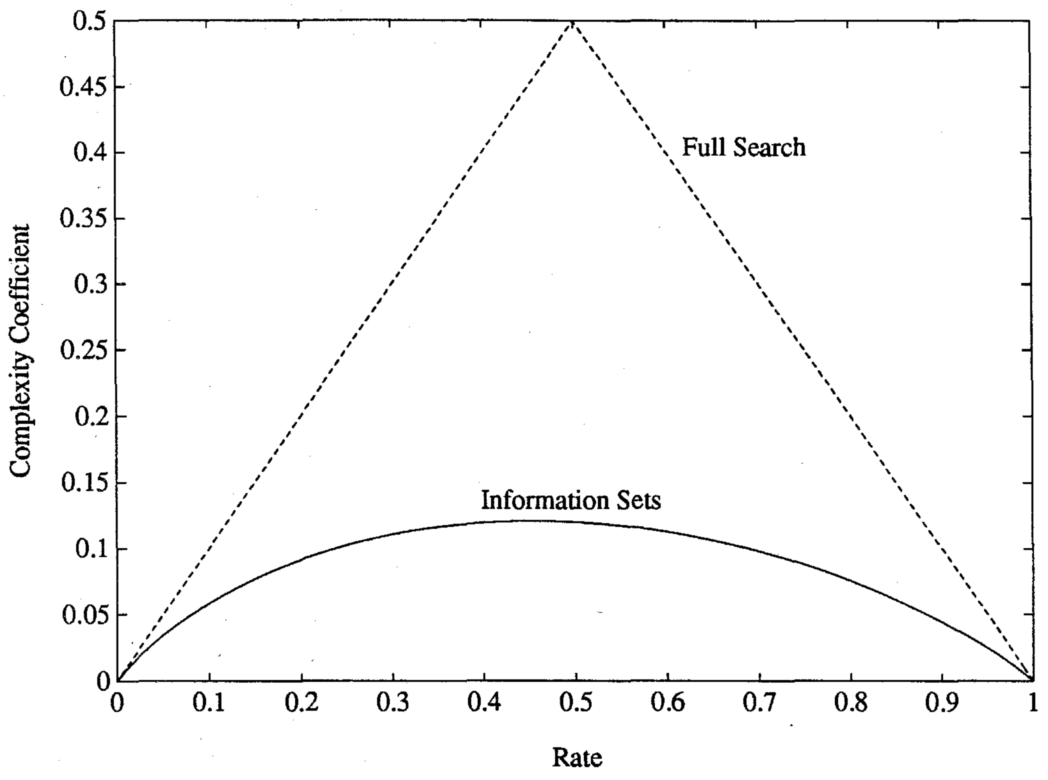


Figure 2.3: Decoding Complexity for Information Set Decoding

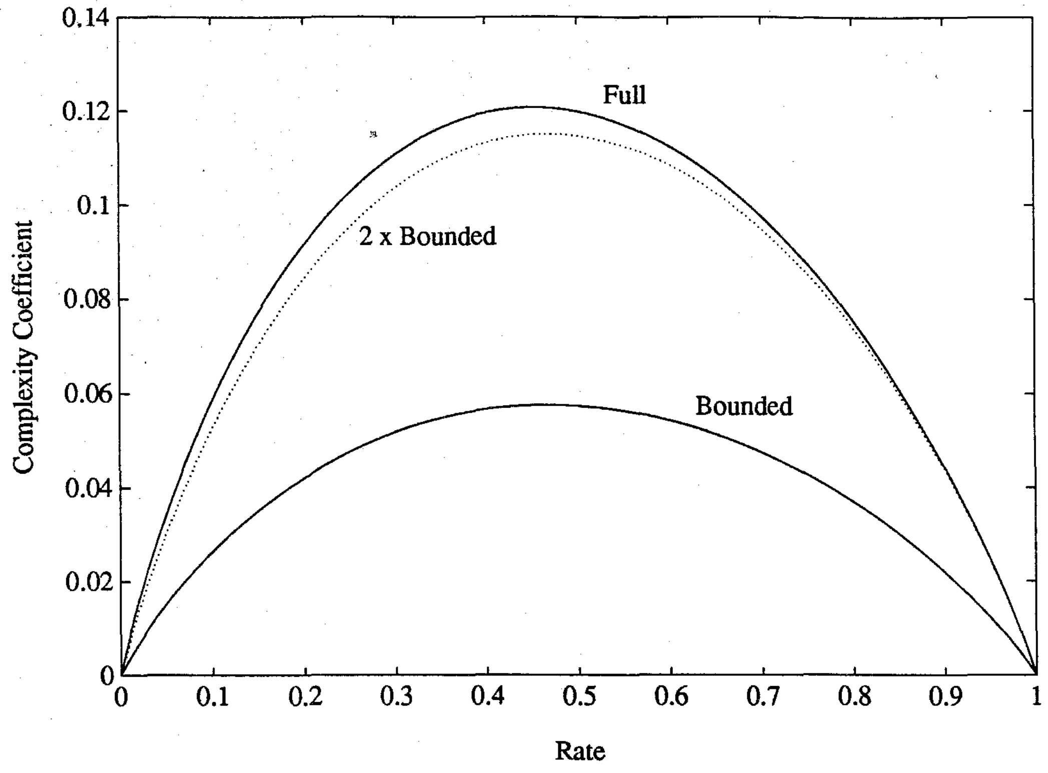


Figure 2.4: Bounded and Full Decoding using Information Sets

Proof: Follows from Theorems 2.8–2.10 with $\rho = nH_q^{-1}(1-R)/2 + o(n)$. ▶

This function is plotted versus rate for the binary case ($q = 2$) in Fig. 2.4. The number of computations is far less than for exhaustive search, and is also much less than for complete minimum distance decoding, requiring slightly less than the square root of the number of computations required for complete decoding at rate one-half. This represents a complexity of slightly more than the ninth root of the number of codewords. We would expect that decoding half as many errors should require a complexity coefficient that is about half as large; Fig. 2.4 shows that this is a reasonable approximation.

In Section 2.4, we mentioned that an error pattern can have at most $d-1$ hard errors if it is to be within the guaranteed soft error correcting power of the code. Conversely, given any set of $d-1$ locations, we can construct an error pattern within the guaranteed soft-error-correcting power of the code with hard errors in all those $d-1$ positions. Thus to derive *a priori* an algorithm which achieves soft decision decoding up to the guaranteed soft-error-correcting power of the code, it is necessary and sufficient that we should be able to correct all patterns of up to $d-1$ hard errors. For most codes, we have $d-1 = nH_q^{-1}(1-R) + o(n) \approx \rho$, so the computational requirement is as given in Theorem 2.13.

Theorem 2.13 *Bounded soft decision decoding using generalized information set decoding has, for virtually all binary linear codes, a complexity $M(C)$ satisfying*

$$\begin{aligned}\frac{1}{n}\log_2 M(C) &= H_2\Big(H_2^{-1}(1-R)\Big) - (1-R)H_2\Big(\tfrac{H_2^{-1}(1-R)}{1-R}\Big) + o(1)\\ &= (1-R)\Big[1 - H_2\Big(\tfrac{H_2^{-1}(1-R)}{1-R}\Big)\Big] + o(1).\end{aligned}$$

Again, the complexity is plotted as a function of R in Fig. 2.3 , where the exhaustive search procedures involve searching through all codewords (for $R \leq 1/2$) or decoding with a trellis [86] (for $R > 1/2$). In practical applications, bounded soft decision decoding asymptotically (in SNR) doubles the error-correcting power. Fig. 2.4 shows that for virtually all codes, it also about doubles the exponent in the number of computations, assuming generalized information set decoding is used. In some applications, we may not wish to decode out to double the guaranteed hard distance, but rather to three halves the hard distance, or some other multiple η. In general, this requires a complexity coefficient of

$$H_2\Big(\frac{\eta}{2}H_2^{-1}(1-R)\Big) - (1-R)H_2\Big(\frac{\eta H_2^{-1}(1-R)}{2(1-R)}\Big) + o(1).$$

Finally, suppose $t = n\tau$ is quite small. Then we have $\left(\frac{1}{1-R}\right)^t = 2^{nF'(R)+o(n)}$ from our results, and so

$$F'(R) = \tau \log\Big(\frac{1}{1-R}\Big).$$

This corresponds exactly to the complexity derived empirically by Omura [103], and also shows that the complexity coefficient is approximately linear in the number of errors corrected if that number is low.

Another parameter of interest is the behaviour of the algorithm when q becomes very large. We have the following result.

Theorem 2.14 *For large q, we have $F(q,R) \rightarrow H_2(1-R)/\log_2 q$. Thus*

$$\lim_{q\rightarrow\infty} F(q,R) = 0.$$

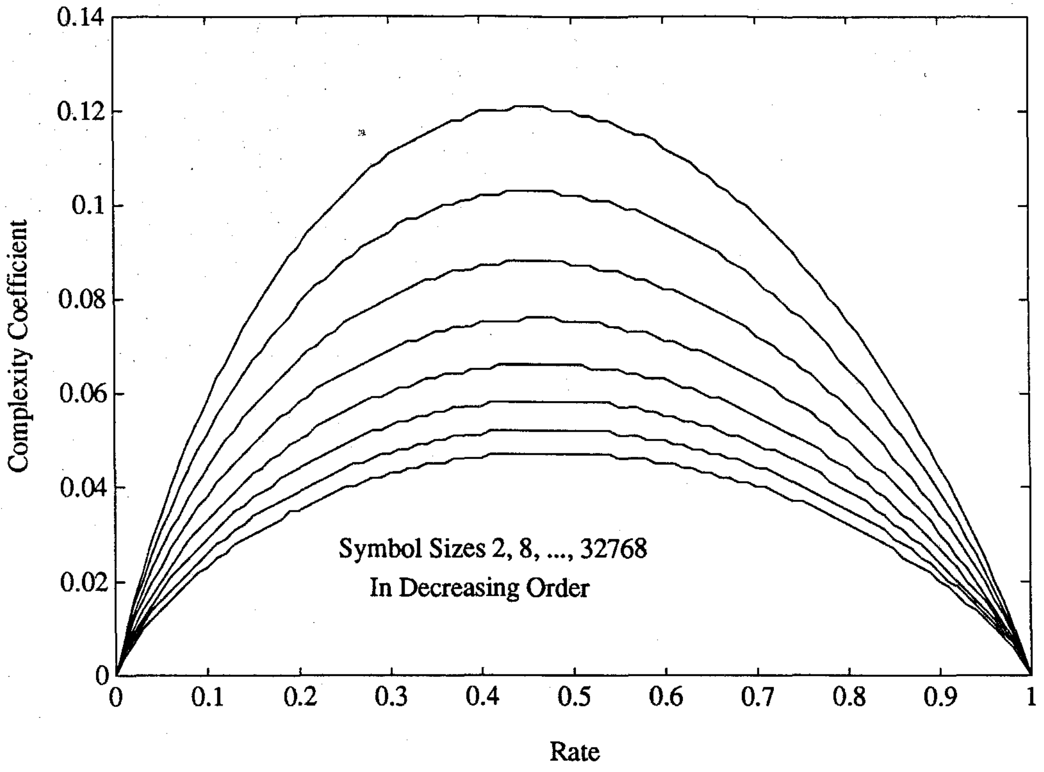
Proof: This follows from the behaviour of the function $H_q^{-1}(x)$ as q becomes large. From the proof of the complexity of the zero neighbours algorithm for non-binary fields , we know that $H_q(x) = x(1+O(1/\log_2 q))$ and that $\lim_{q\rightarrow\infty} H_q^{-1}(x) = x$. From Theorem 2.11, we have

$$\begin{aligned}
\frac{1}{n}\log_2 M(\mathcal{C}) &= H_2\Big(H_q^{-1}(1-R)\Big) - (1-R)H_2\left(\frac{H_q^{-1}(1-R)}{1-R}\right) + o(1) \\
&= H_2(1-R-o(1)) - (1-R)H_2\left(\frac{1-R-o(1)}{1-R}\right) + o(1) \\
&= H_2(1-R) + o(1) - (1-R)o(1) \\
&= H_2(1-R) + o(1)
\end{aligned}$$

Thus the computational effort $M(\mathcal{C})$ has the form $2^{nH_2(1-R)+o(n)}$ for large n, independent of q. This is equivalent to $q^{(nH_2(1-R)+o(n))/\log_2 q}$, so the complexity coefficient is $H_2(1-R)/\log_2 q$ as claimed. ▶

Fig. 2.5 shows the complexity coefficient for complete minimum distance decoding for many values of q. The fact that the complexity coefficient tends to zero with increasing q may seem surprising. It corresponds to the fact that it is always possible to decode by trying every set of k symbols as an information set.

We have mentioned some approaches that have been taken to the problem of constructing information sets. Our result suffers from the disadvantage that it is non-constructive. However, as in the case of the random coding proof of the channel coding theorem, we have something more than an existence proof. Suppose we accept a complexity of $F(R)+\epsilon$ for small positive ϵ for 'insurance.' The probability that we do not get a satisfactory selection is then, by our results above, at most $e^{-2^{n\epsilon+o(n)}}$. The double exponential suggests that we can put our trust in a random number generator to derive the information sets for us.



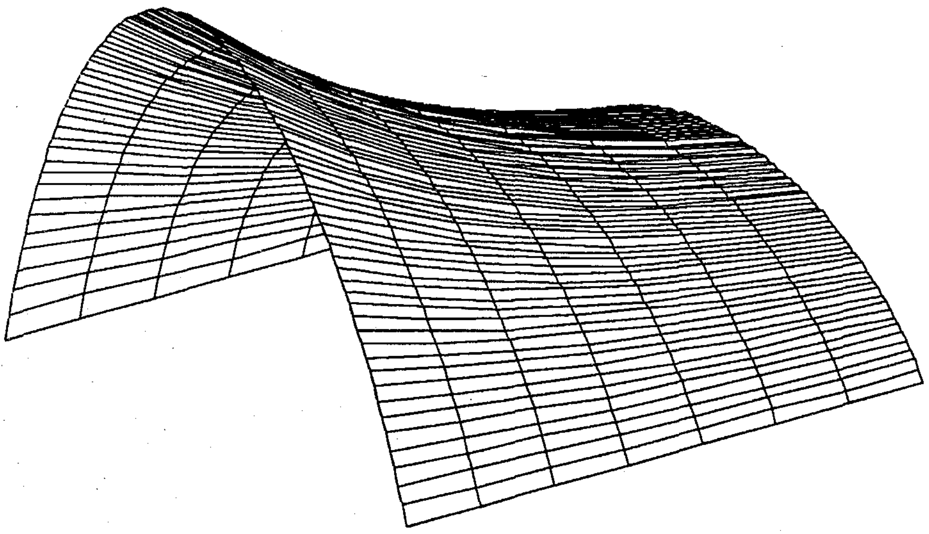


Figure 2.5: Complexity for Various Symbol Fields

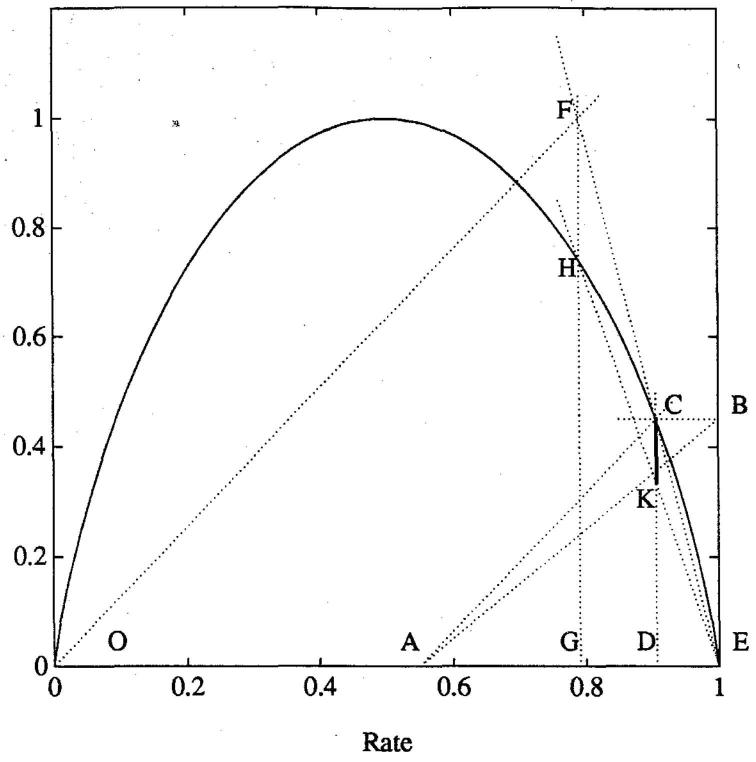


Figure 2.6: Geometric Construction of $F(R)$

2.5.1 A Geometric Construction

It is enlightening to consider a geometric construction for the complexity coefficient of generalized information set decoding. (In the light of the discussion in Section 2.1, it might be interesting to compare this construction to Shannon's geometric construction for $E(R)$ [3].) We show how to construct the complexity function given the entropy function and the standard draughtsman's equipment. For simplicity, we demonstrate the construction for the binary case; the generalization to $GF(q)$ should be clear.

The construction is given in Fig. 2.6. Given R (= 0.55 in the diagram), project a line right at 45° above the horizontal; this intersects the line $y = 1$ at point B. A horizontal line through B intercepts the entropy function at $C = (1 - H_2^{-1}(1 - R), 1 - R)$, *i.e.*, D is $\rho = H_2^{-1}(1 - R)$ from the line $y = 1$. We draw a line through the origin O parallel to AC; this intersects CE at F. From similar triangles, we have $|AE| = (1-R)|OE|$, so $|CE| = (1-R)|EF|$ and $|DE| = (1-R)|GE|$. Thus $|GE| = \rho/(1-R)$. We need $H_2(\rho/(1-R))$, which is the length of GH. Connecting H to E, and labelling the point at which this line intersects CD with the letter K, we find that $|KE| = (1-R)|HE|$, and so $|KD| = (1-R)|HG| = (1-R)H_2(\rho/(1-R))$. As $|CD| = 1-R$, the complexity coefficient is given by the length of the line segment KC (marked in bold in the diagram). For comparison purposes, with rate greater than 1/2 the trivial algorithm has complexity $1-R$: this is the length of the line segment CD in the diagram.

2.6 Comparisons

Fig. 2.7 shows the complexity of four of the decoding methods we have discussed. The complexity of the zero neighbours algorithm ($F_{ZNA}(2,R) \approx H_2\big(2H_2^{-1}(1-R)\big)-(1-R)$ for $R > 0.1887$ and for $q = 2$) is much higher than that for generalized information set decoding — for example, in the case $R = 1/2$, generalized information set decoding requires less than the *square root* of the complexity required by the zero neighbours algorithm. This is even greater than the gain made by the ZNA over exhaustive search (the ZNA requires marginally more than the square root of the number of codewords at $R = 1/2$). In addition to this favourable comparison, generalized information set decoding has two further major advantages. First, it can be modified easily to perform bounded hard-decision decoding, with a significant reduction in complexity. For the ZNA, on the other hand, bounded hard-decision decoding cannot be achieved with lower complexity. Second, the complexity characteristic for large q is much less favourable for the ZNA, approaching the complexity required by exhaustive search, rather than zero.

Previous analysis of algorithms based on the error trapping idea have usually given the lower bound for bounded distance hard decision decoding [37, 41]. Evseev [39] discusses an algorithm — "Q–decoding" — which is basically the same as information set decoding. He shows that with soft decision decoding, the probability of error for the algorithm is no more than double that for maximum likelihood decoding, with

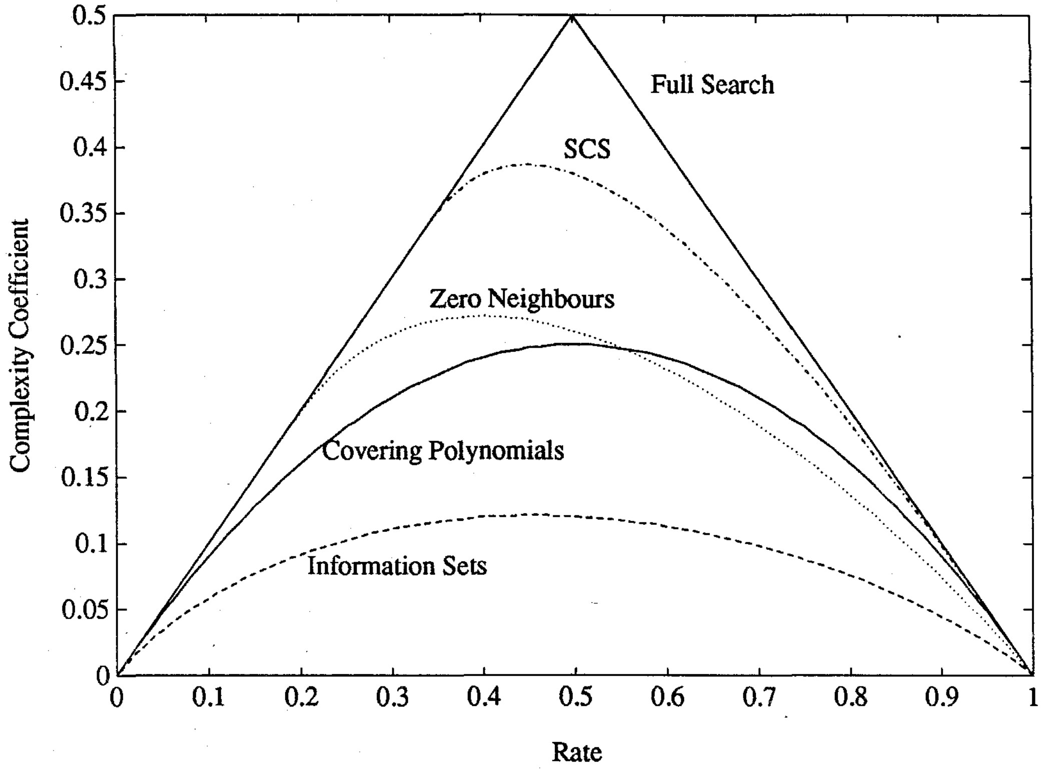


Figure 2.7: Comparison of Complexity of Various Schemes

complexity coefficient $F(R) \leq R(1-R)$. Clearly, it lies far above our (exact) solution; it is the same as the complexity coefficient for covering polynomials.

2.7 Other Members of the Information Set Decoding Family

2.7.1 Systematic Coset Search

This algorithm has been suggested by Montgomery *et al.* [85] and by Levitin [78]. It involves taking a single information set and searching through all possible error patterns in that set. We would expect the complexity of this procedure to be very high; we have the following result:

Theorem 2.15 *For virtually all linear (n,k) codes over $GF(q)$, we have $F(R) = RH_q(H_q^{-1}(1-R)/R)$ if $R > 1 - H_q(R(q-1)/q)$, and $F(R) = R$ otherwise. As q becomes large, we have $F(R) \to \min\,[R, 1-R]$.*

Proof: The complexity is $\sum_{i \leq r_C} \binom{nR}{i}(q-1)^i$, and r_C is about equal to the Goblick bound for most codes. If $H_q^{-1}(1-R) < R(q-1)/q$, the dominant term is the last one (otherwise $F(R) = R$) and $q^{nF(R)} = \binom{nR}{nH_q^{-1}(1-R)}(q-1)^{nH_q^{-1}(1-R)}$, or $F(R) = RH_q(H_q^{-1}(1-R)/R)$ as required. ▶

The result for the binary case is compared with other results in Fig. 2.7. Clearly, it is not an effective algorithm.

2.7.2 Covering polynomials

Although properly speaking we use covering polynomials in the generalized information set decoding algorithm, the method examined here is in essence the same as that originally suggested by Kasami [37]. We take a relatively small number of information sets and try sufficiently many error patterns in each so that in at least one of the sets, the actual error pattern will have been tried. Take the n sets of k consecutive (viewed cyclically) symbols; from Theorem 2.10 the rank of each will be sufficiently close to

k for most codes. The total number of errors will not exceed $nH_q^{-1}(1-R)+o(n)$ for most codes, and the expected number of errors in a given set of k symbols in this case is $nRH_q^{-1}(1-R)+o(n)$. The number of errors cannot exceed the average in *all* the sets, so it is sufficient to try all error patterns of weight up to $nRH_q^{-1}(1-R)+o(n)$ for each of the sets. This gives a complexity of

$$n\sum_{i=0}^{nRH_q^{-1}(1-R)}\binom{nR}{i}(q-1)^i = q^{nRH_q(RH_q^{-1}(1-R)/R)}+o(n) = q^{nR(1-R)+o(n)}.$$

So the complexity coefficient $F(R)$ is $R(1-R)$ for *every* symbol field. This represents an intermediate case between the zero neighbours and systematic coset search algorithms, which tend to the complexity of the trivial algorithms as q becomes large, and the generalized information set decoding algorithm, which tends to zero in complexity as q becomes large. The function is plotted in Fig. 2.7. It is more efficient than the zero neighbours algorithm in the binary case for $R < 0.55$.

2.8 Other Algorithms

2.8.1 Threshold and Majority Logic Decoding

Majority logic and threshold decoding have a long history and have been examined as extensively as any of the algorithms we have considered [93, 95, 96, 10]. They involve selection of codewords from the *dual* code. Each row of the parity check matrix H defines a parity check that each codeword must satisfy. Linear combinations of these rows also provide parity checks. Thus any of the $q^{n-k}-1$ non-zero codewords of the dual code specify a parity check on a codeword. The basic idea is that a set of dual codewords containing one location far more than all others will have corresponding parity checks dominated by that location if the overall number of errors is small.

The details of the procedure are well documented, and we will not discuss them here. Some main results are, from [10]:

— For any linear code over $GF(q)$, the number of errors which can be corrected by one-step majority logic decoding is at most $(n-1)/2(d^{\perp}-1)$ where $d^{\perp}$ is the minimum distance of the dual code.

— For any linear code over $GF(q)$, the number of errors which can be corrected by L-step majority decoding is at most $n/d^{\perp} - 1/2$.

Note that for most codes, $n/d^{\perp}$ tends to a constant for a given rate. Thus asymptotically very few errors can be corrected by these procedures.

On the other hand, it is possible to generalize the decision function and decode any binary linear code — a result due to Rudolph [96]. His proof expresses the error pattern as a threshold function of all 2^{n-k} dual codewords; it is not clear how to improve this, or how to select the dual codewords.

We now show that a special form of majority logic decoding is exactly equivalent to information set decoding; it follows that *all* linear codes over *any* symbol field can be decoded by this method, and that for *most* codes, we achieve a reduction in complexity using the method. The proof is quite simple. Every linear code is equivalent to a systematic code with generator matrix of the form $G = (I_k|P)$. The corresponding parity check matrix (generator of the dual code) is of the form $(-P^T|I_{n-k})$ [21]. Each of the rows of this matrix is a codeword in the dual code. We perform decoding with these $n-k$ dual codewords. For the first k symbols, we take the symbol as being correct. For each of the last $n-k$ symbols, we use one dual codeword — the word with a one in the appropriate location — and threshold on the result of the corresponding parity check. This procedure succeeds if and only if the first k (independent) symbols are error free; in fact, we have merely reinterpreted the information set decoding algorithm. The results on information set decoding all apply; in particular, note that Mandelbaum's result, which implies that there is an information set disjoint from every coset leader, means that this version of majority logic decoding achieves complete decoding for any linear code over any symbol field. The number of different sets of n dual codewords is given by the results in Section 2.5.

The codewords of the dual code have apparently been chosen in a degenerate way. This suggests that other choices might improve on the information set algorithm, although we have been unable to find such an improvement so far.

Finally, we note that Bossert & Hergert [89] have proposed an intriguing algorithm in which the codewords of minimum weight in the dual code are used as parity checks, and a majority vote taken among them. Although there seems to be only a heuristic reason for this algorithm, it would be interesting to investigate an analogue of the zero neighbours algorithm, *i.e.,* to find a set of low weight codewords from the dual code that is sufficient for complete decoding.

2.8.2 Boolean Linear Programming

Omura has examined an algorithm that incorporates some of the features of both the progressive and the redundancy-type algorithms [88]. The idea is to use a method analogous to the simplex algorithm for linear programming with the real variables replaced by Boolean ones.

An account of the mechanics of the simplex algorithm can be found in [90]. Very roughly, the algorithm involves starting with a matrix and a subset non-singular *basis* matrix; we progressively update this basis by taking one column out of the basis and replacing it with a non-basis column. If we do not have an optimal solution (and assuming certain non-degeneracy conditions) it is always possible to decrease the cost by exchanging one pair of columns. Although the simplex algorithm has in the worst case an exponential running time, its attraction is that in practice it is extremely efficient.

Omura's approach is to take an information set as the basis, and to clear the bits in the set to zero. The 'cost' is the weight of the resulting syndrome. We then try to replace a bit in the information set in such a way that the cost goes down. If this is impossible, we try to replace a pair of bits in the information set by a pair of bits outside it to achieve a weight reduction, and so on.

Despite the heuristic reasons behind this procedure, it does not seem that it is better than selecting the information sets at random. Indeed, Clark & Cain [41] draw this conclusion from simulations. One possible reason is that, unlike the real simplex algorithm, Boolean LP does not necessarily achieve a reduction in cost by introducing *one* new column. We may need to try patterns of large numbers of columns, and this will dominate the complexity. There is in fact no reason to believe that the algorithm is effective; however, no proof either way is available, and we include it as an idea that contains both the progressive and the redundancy approaches.

2.9 Continued Division Algorithms

In this section, we investigate a family of algorithms that employs some features of both the zero neighbours algorithm and of information set and covering polynomial methods. The method was first suggested by Farrell [43, 54, 56] on empirical grounds; the analysis here is original. Although the methods can easily be extended to general linear codes, we shall concentrate on cyclic and shortened cyclic codes; this clarifies the ideas and simplifies the analysis.

The basic procedure is as follows. We have a received word $r(x)$, and a dividing codeword $c(x)$. Division by $c(x)$ will produce a remainder $r(x) \bmod c(x)$. By *continued division*, we mean the process of producing $x^i r(x) \bmod c(x)$ for $0 \leq i \leq M$ for some large M. This corresponds to performing longhand division with a large number of zeros appended to the right of $r(x)$. The remainders in this process are all cyclic shifts of words in the original coset; it is easy to recover the correct cyclic shift from any of these. Alternatively, we can view the process as involving continued subtractions of shifts of $c(x)$ that may "wrap around" the end of the word. Thus we perform longhand division in the usual way to get $r(x) \bmod c(x)$. Then we set a 'pointer' to point to the highest order non-zero symbol in $r(x) \bmod c(x)$. Let this be the location indexed by x^i. Then we subtract $x^{i-\deg c(x)}c(x) \bmod x^n - 1$ (or the appropriate non-zero multiple thereof in a non-binary code) to set the bit (or symbol) indexed by the pointer to zero. The pointer location is then multiplied by $x^{-1} \bmod x^n - 1$ (i.e, shifted right cyclically) until it indexes another non-zero symbol, and then the procedure is repeated.

As an example of the effect of this procedure, consider the binary $(23, 12)$ Golay code. We take the syndrome and divide cyclically by a dividing codeword. We take

Dividing Codeword	Undecoded Syndromes
—	2047
$g(x)$	759
$(x^2+x+1)g(x)$	461
$(x^3+x+1)g(x)$	195
$(x^4+x+1)g(x)$	41
$(x^5+x+1)g(x)$	23
$(x^5+x^4+x^3+x^2+1)g(x)$	0

Table 2: Continued Division of the Golay Code

the lowest weight result and divide cyclically by the next dividing codeword, and so on. The table shows the number of undecoded syndromes after each step (the number of cycles in this example was chosen arbitrarily to be twenty). So by using five (minimum weight) codewords plus the generator, we achieve complete decoding. This is not the most efficient method for decoding the Golay code — see, for example, [84]. In fact, no particular attempt has been made to achieve minimization, and we offer it as an illustrative example only.

Our motivation in examining this procedure is derived from a number of sources. First, the procedure is extremely simple to implement, requiring only the most basic application of the operations allowed in the model discussed in Section 2.3. Instead of subtracting many different codewords, we are subtracting one repeatedly; instead of subtracting at scattered locations, we progress from one location to the adjacent location. Secondly, the procedure allows us to allocate complexity in a more satisfactory way. In the zero neighbours algorithm and in information set decoding, we have a very high space complexity and low time complexity. Ideally, we would like to have the option of trading off space complexity against time complexity; however, the problem of generating the zero neighbours and the information sets is too difficult to be done on-line, and so there is no available flexibility in implementation. In continued division, on the other hand, we are effectively adding large numbers of codewords that are generated on-line. Thirdly, we note that in the other algorithms mentioned above, most of the precomputed codewords and information sets are not useful in decoding any given syndrome. Ideally, we should have an algorithm that generates words on-line, but with a distribution skewed towards those words that will be useful.

The main questions regarding this procedure are:

— Under what conditions does the procedure produce the coset leader?

— For how long should the division process continue?

— What is the least number of dividing words necessary?

In addition to answering these questions, we shall suggest a new decoding algorithm based on the continued division process.

2.9.1 Division by the Generator of a Cyclic Subcode

The simplest case is that of continued division by the generator polynomial $g(x)$. This is just error trapping [57], *i.e.,* information set decoding in which the information sets are the n sets of k consecutive bits. The coset leader is found if the error pattern is trappable, and we need only perform n operations after producing the syndrome, as the $n+1$st result is the syndrome, and we go into a loop; allowing for the operations necessary to produce the syndrome, we get a maximum of $n+k$ basic operations, or less than two cycles.[4]

When dividing by a codeword $d(x) = i(x)g(x)$ other than the generator, we draw a distinction between the case when $d(x)$ divides $x^n - 1$ and the case when it does not. If $d(x)$ divides $x^n - 1$, it is itself the generator of a cyclic code of block length n, a subcode of $\mathcal{C}$. (We label this subcode $\mathcal{C}'$.) Let the received word be of the form $r(x) = C(x) + E(x)$, where $E(x)$ is the coset leader. Suppose that it happens that $C(x) \in \mathcal{C}'$. Then the situation is exactly as it would have been if we had been using the code $\mathcal{C}'$ and obtained the error pattern $E(x)$. Continued division in this case corresponds to error trapping in the subcode. Thus the error pattern is detected if it is trapped in the subcode, and this happens if the burst length is less than the

[4]A cycle is defined in the obvious way to consist of n basic operations.

redundancy. As the subcode has higher redundancy than the main code, however, so far more error patterns can be detected if the transmitted codeword $C(x)$ is in the subcode.

In general, this will not håppen, and the received word will be of the form $r(x) = C(x) + E(x) = C_1(x) + E_1(x)$, where $C_1(x) \in C'$ is the nearest codeword to the received vector in the subcode, and $E_1(x)$ is in the same coset as $E(x)$. $E_1(x)$ is the *subcode* coset leader. Three situations are possible when we begin division of $r(x)$ by $d(x)$:

— $r(x)$ is a coset leader in C'.

— $r(x)$ is not a coset leader in C', but no word of lower weight in the same coset has burst length $\leq n - k + \deg\, i(x)$.

— $r(x)$ is not a coset leader in C', and there are words of lower weight in the same coset with burst length $\leq n - k + \deg\, i(x)$.

In the first case, no weight reduction is possible on division by $d(x)$. In the second case, no weight reduction will be achieved if $\deg\, r(x) < \deg\, d(x)$; we can only achieve weight reduction in the exceptional case of a 'bonus' pattern appearing, *i.e.,* if there is a word $w(x)$ of lower weight in the coset of burst length $\leq n - k + \deg\, i(x) + l$ such that $r(x) = x^l a(x) d(x) + w(x)$. In the third case, we definitely achieve a weight reduction.

This suggests the following algorithm, which employs features of both information set decoding and the zero neighbours algorithm: we take a large number of codewords $c_i(x)$, all of which divide $x^n - 1$. Divide $r(x)$ by each $c_i(x)$ for 2 cycles. Take the lowest weight resulting word, and start the process again, treating this word as $r(x)$. Eventually, no further weight reduction will be obtained from any dividing word; we then apply codewords from an appropriately constructed table to get from the lowest weight word found to that point to the coset leader. Note that from the weight reduction mechanism above, we should expect that the first part of the algorithm halts only when there are very few words of lower weight in the coset, *i.e.,* when the lowest weight word found to that point has low weight. Then, as in the zero neighbours algorithm, only codewords of relatively low weight are required. Note also that, assuming the burst length of the error pattern is not excessively large, we do not need to store any zero neighbour that is contained in one of the cyclic subcodes. Heuristically, this algorithm seems promising, though exact results for given codes are difficult to obtain.

2.9.2 Dividing by a 'Non-Cyclic' Codeword

In this case, $d(x)$ does not divide $x^n - 1$, so the 'wrap around' version of $d(x)$, *i.e.*, $x^{-1}d(x) \bmod x^n - 1$, does not belong to the subcode C'. After one complete cycle, we have added a codeword, non-zero in general, to the original syndrome, so the procedure takes longer to go into a loop than in the first case. If $S_0(x)$ is the syndrome, and $S_1(x)$ is the result after one cycle, we have the relation $S_1(x) = x^n S_0(x) \bmod d(x)$. In general, after i cycles, we have $S_i(x)$ as a result, and have added a codeword $\beta_i(x)$ to the original syndrome, where

$$\begin{aligned} S_i(x) &= x^{in} S_0(x) \bmod d(x) && (2.4)\\ \Rightarrow S_0(x) + \beta_i(x) &= x^{in} S_0(x) \bmod d(x) && (2.5)\\ \Rightarrow \beta_i(x) &= (x^{in} - 1) S_0(x) \bmod d(x). && (2.6) \end{aligned}$$

We get repetition when $\beta_i(x) = 0$, which occurs at the first i for which $d(x) \mid x^{in} - 1)S_0(x)$. Let $\gcd(d(x), x^n - 1) = d'(x)$. Then we must have $d'(x) \mid x^{in} - 1$, which implies that ord $(d'(x)) \mid in$. We need the least i for which this is true, which is given by lcm $(n, \text{ord } (d'(x)))/n$. So in general the process of dividing the syndrome $S_0(x)$ continuously by the dividing word $d(x)$ has period p given by

$$p = \frac{\text{ord } \Big(d(x)/\gcd(d(x), S_0(x))\Big)}{\gcd\Big(\text{ord } \Big(d(x)/\gcd(d(x), S_0(x))\Big), n\Big)}.$$

As an example, consider the binary $(23, 12)$ Golay code. We take the syndrome and divide continuously by the two codewords $(x^{10}+x^3+1)g(x)$ and $(x^{10}+x^7+1)g(x)$. We assert that the coset leader will be found by this process.

The generator polynomial and the two information polynomials above are irreducible, so in at least one of the cases we have $\gcd(d(x), S_0(x)) = 1$. Both the information polynomials are primitive, so their order is $2^{10} - 1 = 1023$ [58]. Now the order of the product of two distinct monic irreducible polynomials is the lowest common multiple of the orders of the polynomials, so ord $d(x) = 23.1023$ in each case. The denominator in the equation is gcd $((23.1023), 23) = 23$, and so the period is 1023 in each case. Thus all codewords of degree less than 22 except one are added. In at least one of the first n basic operations (in fact, in most of them) the error pattern will be 'trapped,' and the appropriate number of cycles later, the correct codeword has been added and the error pattern appears in the clear. This happens unless the error pattern or the starting word is a multiple of $i(x)$; it is easy to verify that the starting word and the error pattern cannot be a multiple of $i(x)$ for *both* $i(x)$'s.

Although this procedure is extremely inefficient as a practical algorithm, it serves as an illustration of the possibility of tradeoff of space versus time complexity. Our question of how many dividing codewords were necessary is seen to be answered by 'very few,' at least for some codes.[5] The important point is that we can calculate exactly when repetition occurs, and we can characterize the codewords that are effectively added during the process. For the most effective results, we should choose the dividing words so that the added codewords are of low weight, though finding the words seems to be a very difficult task.

2.9.3 Continued Division for General Linear Codes

We turn to a generalization of the continued division procedure to the case of general linear codes. Although the result we report in this section is negative, we feel that the greatest prospect for improving on the information set decoding algorithm by an exponential amount asymptotically lies in a variant of the procedure outlined below,

[5]Ideally, we would like to have one dividing word; if we choose $i(x)$ so that it has degree 11 (so that it can never be a multiple of an error pattern), we find that because 2047 is divisible by 23, the period of the process is 89, not 2047, and the argument no longer holds. In general, a quadratic residue code of block length n must always satisfy $n \mid 2^{(n-1)/2} - 1$, as this is a necessary condition for 2 to be a quadratic residue of n [57]. Thus we must always take $i(x)$ to have degree one less than the maximum for quadratic residue codes.

and we suggest some promising lines of attack.

For our generalization of the continued division process, we select an information set of the linear code $\mathcal{C}$ and a subset of the information set. The set of codewords whose non-zero information bits are confined entirely to the subset define a subcode $\mathcal{C}'$ of the code. We clear the bits in the subset to zero; this detects the error pattern if the added codeword is in the subcode and the error pattern is disjoint from the subset. We calculate an expression for the probability of decoding in the case where we have the second lowest weight word in the coset (weight $p = n\rho$), the coset leader has weight only slightly less, and the two words intersect in $n\rho^2$ bits. This is the typical case, and the one that dominates the complexity result, because otherwise we have easily derivable statistical information about the coset leader from the next lowest weight word. Suppose there are w errors in the information set, as well as m correct ones of $r(x)$. We now draw a distinction between two types of errors: ones turned to zeros (type I) and zeros turned to ones (type II). A necessary condition for decoding is that the type I errors are in the parity bits: if a type I error is in the cleared subset of the information set, the error pattern is not trapped, while if it is in the uncleared part of the information set, the added codeword cannot be in the subcode. Thus, given that we have w errors in the information set, a necessary condition is that all these errors are type II errors, and this event has probability $\binom{p\rho}{w}/\binom{p}{w}$. Instead of explicitly choosing a subset of the information set, we choose an information bit at random and clear it, then choose another information bit at random and clear that, and so on. The probability that the errors are trapped is just the probability that the m correct ones are selected before the w incorrect ones, which is $\binom{m+w}{w}^{-1}$. (Note that we are only including an extra weight computation at each step compared to information set decoding; no new codewords are subtracted. Thus we achieve a higher probability of decoding essentially free.) Then we must multiply by the probability that we get w errors and m correct ones in the information set. the probability that there are w errors in the information set is $\binom{n-k}{p-w}\binom{k}{w}/\binom{n}{p}$, and the probability that there are m correct ones given that there are w errors is $\binom{k-w}{m}\binom{n-k-p+w}{p(1-\rho)-m}/\binom{n-p}{p(1-\rho)}$.

Overall, the probability that we get decoding with a single trial is

$$\left[\frac{\binom{p\rho}{w}}{\binom{m+w}{w}\binom{p}{w}}\right]\left[\frac{\binom{k}{w}\binom{n-k}{p-w}}{\binom{n}{p}}\right]\left[\frac{\binom{k-w}{m}\binom{n-k-p+w}{p(1-\rho)-m}}{\binom{n-p}{p(1-\rho)}}\right].$$

Now let $m = n\mu$ and $w = n\omega = n\xi\rho^2$, with $0 \leq \xi \leq 1$. The expected number of decoding operations will be the inverse of the probability above from the argument used for the information set algorithm. We find that the complexity coefficient $F(R)$ is given by

$$\begin{aligned} F(R) \;=\; & (R-\xi\rho^2)H\Big(\frac{\mu}{R-\xi\rho^2}\Big) + (1-R-\rho-\xi\rho^2)H\Big(\frac{\rho(1-\rho)-\mu}{1-R-\rho-\xi\rho^2}\Big) \\ & -\rho H(\xi\rho) + (1-R)H\Big(\frac{\rho-\xi\rho^2}{1-R}\Big) + RH\Big(\frac{\xi\rho^2}{R}\Big) + \rho^2 H(\xi) \\ & -(2-\rho)(1-R) - (\mu+\xi\rho^2)H\Big(\frac{\xi\rho^2}{\mu+\xi\rho^2}\Big) \end{aligned}$$

where $H(x)$ is the binary entropy function. For a given ω, the function is maximized for a unique μ, obtained by setting the ratio of the probabilities of decoding with m and $m+1$ correct ones in the information set to one. We find that

$$\mu_{opt} = \frac{\rho(1-\rho)R-\omega^2}{1+\omega-\rho} - \omega.$$

Substituting this into the equation and maximizing over the choice of the parameter ξ gives us the overall complexity of this algorithm. Unfortunately, numerical simulations indicate that the expression is maximized at $\xi = 0$, which corresponds to information set decoding. We feel, however, that this result does not rule out the possibility of a variant of the scheme achieving an exponential reduction in complexity asymptotically. One possibility is to skew the distribution of the information sets so that ones are more likely to be in the information set than zeros. The rationale is that we must have all type I errors in the parity bits, whereas we can tolerate some type II errors in the information set; the skewed distribution may help us achieve this. An exponential improvement over information set decoding would be of considerable importance, and the matter is the subject of continuing investigations.

2.10 Application to Convolutional Codes

A number of results in recent years have dealt with the link between block and convolutional codes [83, 86, 87, 97]. As far as decoding is concerned, the main outcome is that block codes can be decoded on a trellis [86]. Thus we can achieve *full* soft-decision decoding of block codes with complexity bounded by q^{n-k}. In the light of our results, the reverse application is worth investigating, *i.e.,* that of applying a block decoding algorithm to a convolutional code.

It is well known [21] that we can view a truncated convolutional code as a block code: in fact, the Lth truncation [21] of an (n, k) convolutional code is an $(n(M + L), kL)$ linear block code. Our block decoding algorithm would only be able to decode the truncated convolutional code, but in practice we truncate the code no matter what algorithm is being used. Another way to view the problem is to start with a block code and to transform to a convolutional code. For example, Solomon & van Tilborg have shown that any rate k/n quasi-cyclic code can be represented as an equivalent convolutional code [83]. We take such a block code, find its decoding algorithm and then change to a convolutional code, mapping the decoding codewords over. Assuming that the necessary results on the average properties of the codes remain valid, we should find that information set decoding of a rate 1/2 binary convolutional code of truncated length N has complexity about $2^{N/9}$, by analogy with the block decoding case. If we ignore the difference between bounded soft decision decoding and full soft decision decoding, we should find that information set decoding is competitive with Viterbi decoding as long as the truncated length does not exceed about nine times the memory. Any enhanced algorithm for block codes would be even more competitive; we suggest this as a topic for future work.

2.11 Conclusions

Exact solutions for the complexity coefficient for many different types of decoding algorithm with various decoding strategies (full hard decision decoding, and bounded hard- and soft-decision decoding) have been given. Generalized information set decoding gives results that are significantly better than the best available bounds from other algorithms, and vastly less than the requirements from the trivial exhaustive search algorithms. Indeed, for large symbol fields, the gain over the full search algorithms is essentially unlimited.

Comparison of the complexity requirements for the various decoding strategies yields an insight into the tradeoffs of performance versus complexity that are available. In particular, bounded soft decision decoding gives a performance asymptotically twice as good as that for bounded hard decision decoding for the AWGN channel. Using generalized information set decoding, it requires a complexity coefficient that is about twice as high.

We have formulated a model in which the ideas behind the various algorithms can be combined, and have put forward and analysed a new approach (continued division) that synthesizes the two main heuristics behind most combinatorial decoding algorithms. This and the other algorithms are expected to be applicable to decoding convolutional codes.

Finally, some useful results concerning the weight structure of the average linear code and the absence of sets of k symbols with relatively low rank are given.

APPENDICES

APPENDIX A

Weight Distribution of Average Linear Code

We derive an important bound on the average behaviour of codes. The result shows that for most codes, the weight enumerator behaves as a suitably scaled version of the weight enumerator of the repetition code. In Appendix B, we show that a lower bound of the same form holds for virtually all codes for the weight enumerator of *every coset.*

It is well known and has been noted many times [98, 93, 10] that the expectation of the number of codewords of weight w of a randomly chosen binary linear code is very close to $\binom{n}{w}2^{-(n-k)}$. It is surprising, therefore, that the theorem below is new, and equally surprising that the general version of the corollary was first stated relatively recently [17] (the binary version has been known for much longer [65, 66, 51]).

Theorem A.1 *For any fixed ω and R with $0 < \omega, R < 1$ and for any prime power q, the fraction of linear (n, nR) codes over $GF(q)$ with $A(n\omega) = q^{n[H_q(\omega)-(1-R)]+f(n)}$, where $f(n) = \Omega(\log n)$ is less than $q^{-[|f(n)|+O((\log n)/n)]}$. Thus*

$$|\frac{1}{n}\log_q A(n\omega) - [H_q(\omega) - (1-R)]| < \alpha^2\sqrt{n}$$

for a fraction of more than $1 - q^{-\alpha^2\sqrt{n}+O(\log n)}$ of all linear (n, nR) codes over $GF(q)$.

Proof: Assume that the components of the generator matrix are chosen at random from the uniform distribution. (This may result in a code which has rank less than

k; we deal with this point later.) For a given w, define $\mathbf{X}$ to be a random variable denoting the number of non-zero codewords of weight w. There are q^k-1 combinations of the k rows if at least one row must be taken. Let $\mathbf{X}_i, 1 \le i \le q^k-1$ be a random variable taking the value 1 if the ith combination gives a codeword of weight w, and taking the value 0 otherwise. We have $\mathbf{X} = \sum_i \mathbf{X}_i$, and

$$E\mathbf{X} = E\sum_i \mathbf{X}_i = \sum_i E\mathbf{X}_i = (q^k-1)E\mathbf{X}_i = \frac{(q^k-1)\binom{n}{w}(q-1)^w}{q^n},$$

where we have used linearity of expectation and the fact that $E\mathbf{X}_i$ is independent of i. Thus

$$E\mathbf{X} = q^{n[H_q(w/n)-(1-R)]+o(n)}.$$

For the variance of $\mathbf{X}$, we have

$$\begin{aligned}
\sigma^2(\mathbf{X}) &= E(\mathbf{X}^2) - E^2(\mathbf{X}) \\
&= E\sum_{i,j} \mathbf{X}_i\mathbf{X}_j - E^2(\mathbf{X}) \\
&= \sum_i E\mathbf{X}_i + E\sum_{\substack{i,j \\ i\neq j}} \mathbf{X}_i\mathbf{X}_j - E^2\mathbf{X} \\
&= E\mathbf{X} + (E\sum_i \mathbf{X}_i)(E\sum_{\substack{j \\ j\neq i}} \mathbf{X}_j) - E^2\mathbf{X} \\
&= \big(1 - 1/(q^k-1)\big)E\mathbf{X}.
\end{aligned}$$

Now from Chebyshev's inequality, $\Pr(|\mathbf{X}-\mu| \ge t) < \sigma^2/t^2$, so

$$\Pr(|\mathbf{X}/\mu - 1| \ge \alpha) < \sigma^2/(\mu\alpha)^2 < (\mu\alpha^2)^{-1},$$

which is of the required form. This is sufficient to prove the result for a generator matrix selected at random if we do not insist that the resulting matrix has rank k.

To complete the proof of the theorem, we need to deal with the case where we do insist on this condition. The probability that the matrix will have rank k is lower bounded by the probability that the $k \times k$ matrix formed by taking the first k columns has rank k. The combinatorial arguments involved were first discussed by Landsberg in 1893 [74], and run as follows: the first row must be non-zero, which happens with probability $1-q^{-k}$. The second row must lie outside the one-dimensional subspace containing 0 and the first row; this happens with probability $1-q^{-(k-1)}$. The $(i+1)$st row must lie outside the i-dimensional subspace spanned by the first i rows, and this happens with probability $1-q^{-(k-i)}$. The probability that a randomly chosen $k \times k$ matrix over $GF(q)$ is non-singular is thus $\prod_{i=0}^{k-1}(1-q^{-(k-i)})$. For any given k, this quantity is lowest for $q=2$, and in that case, the product converges quickly to about 0.288 [64]. Thus a loose lower bound on the probability that a randomly chosen $k \times n$ matrix over $GF(q)$ has rank k is 0.288 for all but very low values of n. This means that we can multiply the probability of selecting a non-binomially distributed code by a constant $C < (0.288)^{-1}$; we can therefore get an upper bound on the number of codes not satisfying the relation given in the theorem by multiplying the fraction of codes for which it is valid by this constant. The constant is absorbed into the $o(n)$ term in the exponent, which is sufficient to show the theorem. ▶

We have the following corollary:

Corollary A.2 *The Gilbert-Varshamov bound is tight for virtually all linear codes over any symbol field. More precisely, the fraction of linear (n, nR) codes over $GF(q)$ that have minimum distance d satisfying $H_q(d/n) \geq 1-R+\sigma$ for $\sigma > 0$ is less than $q^{-n\sigma+o(n)}$.*

APPENDIX B

Weight Distribution in Cosets

We examine the problem of showing that the same binomial distribution holds for all cosets in most codes. This is a difficult problem; it has as a consequence the fact that the Goblick bound on the covering radius is tight for virtually all linear codes.[1] This was unproven until relatively recently despite the efforts of many [99, 100]; indeed, Piret and Delsarte, who solve the non-linear case, offer it as a challenge [100]. The proof is due to Blinovskii [17]; because it is important, relatively complicated, and recent, we give a version of the theorem and proof here.

Theorem B.1 (Blinovskii) *For any fixed R in $]0,1[$ and for any ω such that $H_q(\omega) > 1 - R$, the fraction of codes which have any coset in which*

$$A'(n\omega) < \frac{\sum_{i=0}^{n\omega}(q-1)^i}{n^{\log_2 q+\alpha} q^{n(1-R)}}$$

tends to zero faster than $q^{-n^\delta(1+o(1))}$ for any $\delta < 1$. Thus

$$\lim_{n\to\infty} \frac{1}{n} \log_q A'(n\omega) = H_q(\omega) - (1-R)$$

holds in every coset in virtually all codes.

Proof: A straightforward application of Chebyshev's inequality is not sufficient: any given coset has an exponentially low probability of being distributed other than

[1]The covering radius [10] is the weight of the highest-weight coset leader. A simple sphere-packing argument shows that we must have $\rho \geq H_q^{-1}(1-R) + o(1)$; this is known as the Goblick bound [48].

binomially, but the number of cosets is exponentially high. Thus a single application of Chebyshev's inequality will show only that an exponentially low *fraction* of cosets are distributed 'badly,' but this amounts to an exponential number of cosets. The idea of the proof is basically to select a code of dimension less than k, apply Chebyshev's inequality, add an extra basis codeword at random, and then reapply Chebyshev's inequality.

We begin by selecting an $l \times n$ generator matrix at random from the uniform distribution; the optimum choice for l will be given later. Let $\mathbf{X}_{nlt}$ be a random variable denoting the number of codewords (resulting from taking combinations of one or more rows) in a sphere of radius t around a given word. We have

$$\mu \equiv E\mathbf{X}_{nlt} = (q^l - 1)V_{nt}q^{-n},$$

where $V_{nt} = \sum_{0 \leq i \leq t} \binom{n}{i}(q-1)^i$. By Chebyshev's Inequality, we have

$$\Pr\{|\mathbf{X}_{nlt} - \mu| \geq q^{\epsilon+1}\sqrt{\mu}\} < \frac{\sigma^2}{q^{2\epsilon+2}\mu},$$

and as in Theorem A-1, we have $\sigma^2 = \mu(1 - (q^l - 1)^{-1})$, so

$$\Pr\{|\mathbf{X}_{nlt} - \mu| \geq q^{\epsilon+1}\mu^{1/2}\} < q^{-2\epsilon-2}.$$

If this relation holds for an n-tuple $\mathbf{x}$ and a code, we say that the sphere of radius t about $\mathbf{x}$ is 'bad' with respect to ϵ. Note that although we will be adding codewords later, the term 'bad sphere' will always mean the same thing (*i.e.*, we do not replace l by $l+1$ after the first step).

Let $\mathrm{N}_0(\mathcal{C}, t, \epsilon)$ be a random variable denoting the number of n-tuples $\mathbf{x}$ which are at the centre of bad spheres in the code $\mathcal{C}$. (Henceforth we abbreviate this by N_0, and the parameters are understood.) From Equation A-1 above, we have $E\mathrm{N}_0 < q^{n-2\epsilon}$. Because N_0 is non-negative for any parameters, the fraction of codes for which N_0 is greater than $q^\epsilon E(\mathrm{N}_0)$ is less than $q^{-\epsilon}$. Thus the relation $\mathrm{N}_0 < q^{n-\epsilon}$ holds for a fraction of at least $1 - q^{-\epsilon}$ of all codes.

We now start an iterative procedure in which new codewords are added successively to the generator matrix, and in which the number of bad spheres in the new code is bounded as before. In taking averages in what follows, it should be understood that we are talking about averages over the ensemble for which the equation $\mathbf{N}_0 < q^{n-\epsilon}$ holds.

Step I: Add a randomly chosen n-tuple $\mathbf{w} \notin \mathcal{C}$ to the generator matrix of the code. The new code consists of words of $\mathcal{C} \cup (\mathbf{w} + \mathcal{C})$. Let $\mathbf{N}_1$ denote the number of bad spheres in the new code, and let $\mathbf{X}_{nit}$ be the event that the ith word from F_q^n is at the centre of a bad sphere. We have

$$E\mathbf{N}_0 = E(\sum_{i=1}^{q^n-1} \mathbf{X}_{nit}) = (q^n - 1)E\mathbf{X}_{nit}.$$

A necessary condition for the ith word from F_q^n to be at the centre of a bad sphere is that it is at the centre of a bad sphere for $\mathcal{C}$ and for $\mathbf{w} + \mathcal{C}$. These two events are independent (as $\mathbf{w}$ is chosen randomly) so $E\mathbf{X}_{nit} \leq q^{-\epsilon} \cdot q^{-\epsilon}$, and $E\mathbf{N}_0 < q^{n-2\epsilon}$. As before, a fraction of less than $q^{-\lambda}$ of the codes can have a number of bad spheres that is more than $q^{-\lambda}$ times the average; thus

$$\mathbf{N}_1 < q^{n-2\epsilon+\lambda}$$

for a fraction of at least $(1 - q^{-\epsilon})(1 - q^{-\lambda})$ of all codes.

For Step II, we add another codeword, and the same analysis holds, except that in place of ϵ, we now have $2\epsilon - \lambda$. We find that

$$\mathbf{N}_2 < q^{n-2(2\epsilon-\lambda)+\lambda}$$

for a fraction of at least $(1 - q^{-\epsilon})(1 - q^{-\lambda})^2$ of the codes. In general, after the ith step, we have

$$\mathbf{N}_i < q^{n+\lambda}(\mathbf{N}_{i-1}q^{-n})^2$$

for a fraction of $(1 - q^{\epsilon})(1 - q^{\lambda})^i$ of all codes.

Now note that if a code has a bad sphere centred at $\mathbf{x}$, it also has bad spheres centred at $\mathbf{c}_i + \mathbf{w}$ for all codewords $\mathbf{c}_i$. After step m, we have a code of dimension $l + m$. We will terminate the iteration after step m, choosing l and m so that $l + m = k$. At this iteration, if the code has any bad spheres, it has at least q^k. Choose m so that

$E\mathbf{N}_m < 1$. Then $\Pr\{\mathbf{N}_m > q^k\} < \Pr\{\mathbf{N}_m > q^k E\mathbf{N}_m\} < q^{-k}$. So there are no bad spheres for a fraction of

$$(1 - q^{-\epsilon})(1 - q^{-\lambda})^m(1 - q^{-k})$$

of all codes.

We need to find the value of m. The $\mathbf{N}_m$'s satisfy the recurrence

$$\mathbf{N}_m < q^{n+\lambda}(\mathbf{N}_{m-1}q^{-n})^2$$

with $\mathbf{N}_0 < q^{n-\epsilon}$. We find that

$$\mathbf{N}_m < q^{n-2^m(\epsilon-\lambda)-2\lambda}.$$

Taking $m = \lceil \log_2 n \rceil$ and $\epsilon - \lambda = 1$ gives $E\mathbf{N}_m < 1$ as required. So $l = k - \lceil \log_2 n \rceil$, and $E\mathbf{X}_{nlt} = E\mathbf{X}_{nkt}/n^{\log_2 q}(1 + o(1))$, *i.e.,* is less by only a polynomial factor. Thus a sphere that has exponentially fewer codewords than the average in the (n, k) code will also be 'bad' in the sense of the relation given in the theorem.

All that remains is to choose ϵ. We want $(1 - q^{-\epsilon})(1 - q^{-\lambda})^m(1 - q^{-k})$ to tend to 1 as quickly as possible. Thus we should make λ (and hence ϵ) as large as possible. On the other hand, our definition of a 'bad' sphere is tied to the value of ϵ, so if ϵ is too large, we cannot be sure that the estimate for the number of words in the sphere of radius t is logarithmically accurate. To avoid this, we take ϵ to be $o(n)$, say n^α with $\alpha < 1$. Then $(1 - q^{-\epsilon})(1 - q^{-\lambda})^m(1 - q^{-k}) \rightarrow q^{-n^\alpha(1+o(1))}$, *i.e.,* the proportion of codes for which the result is invalid decays to zero subexponentially. ▶

Bibliography

[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. Peredachi 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 Transactions on Information Theory,* vol. IT-14, pp. 662–664, 1968.

[9] G. J. Chaitin, "On the difficulty of computations," *IEEE Transactions on Information 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," *Information & Control,* vol. 26, pp. 369–380, 1974.

[12] R. T. Chien and D. M. Choy, "Algebraic generalization of BCH-Goppa-Helgert codes," *IEEE Transactions on Information Theory,* vol. IT-21, pp. 70–79, 1975.

[13] V. D. Goppa, "A new class of linear error-correcting codes," *Probl. Peredachi Inform.,* vol. 6, pp. 207–212, 1970.

[14] T. Kasami, "A Gilbert-Varshamov bound for quasi-cyclic codes of rate 1/2," *IEEE Transactions on Information 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 Transactions on Information 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. Peredachi 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 Transactions on Information Theory,* vol. IT-23, pp. 157–166, 1977.

[19] M. A. Tsfasman, S. G. Vlădut, 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. Peredachi 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. Peredachi Inform.,* vol. 13, pp. 5–17, 1977.

[24] V. Yu. Krachkovskii, "Complexity of constructing codes with specified correction properties," *Probl. Peredachi Inform.,* vol. 15, pp. 50–55, 1979.

[25] V. V. Zyablov, "An estimate of the complexity of constructing binary linear cascade codes," *Probl. Peredachi Inform.,* vol. 7, pp.5–13, 1971.

[26] J. T. Coffey and R. M. F. Goodman, "The complexity of information set decoding," submitted to *IEEE Transactions on Information Theory.*

[27] P. Martin-Löf, "The definition of random sequences," *Information & Control,* 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 International Conference on Applied Algebra, Algorithmics and Error-Correcting Codes, Toulouse, France, October 1984.

[30] P. Delsarte, "On subfield subcodes of modified Reed-Solomon codes," *IEEE Transactions on Information Theory,* vol. IT-21, pp. 575–576, 1975.

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

[32] L. B. Levitin and C. R. P. Hartmann, "A new approach to the general minimum distance decoding problem — the zero neighbors algorithm," *IEEE Transactions on Information Theory,* vol. IT-31, pp. 378–384, 1985.

[33] E. Prange, "The use of information sets in decoding cyclic codes," *IRE Trans.*, vol. IT-8, pp. S5–S9, 1962.

[34] F. J. MacWilliams, "Permutation decoding of systematic codes," *Bell Syst. Tech. J.,* vol. 43, pp. 485–505, 1964.

[35] S. G. S. Shiva and K. C. Fung, "Permutation decoding of certain triple-error-correcting binary codes," *IEEE Transactions on Information Theory,* vol. IT-18, pp. 444–446, 1972.

[36] A. Benyamin-Seeyar, S. G. S. Shiva, and V. K. Bhargava, "Capability of error-trapping technique in decoding cyclic codes," *IEEE Transactions on Information Theory,* vol. IT-32, pp. 166–180, 1986.

[37] T. Kasami, "A decoding procedure for multiple-error-correcting cyclic codes," *IEEE Transactions on Information Theory,* vol. IT-10, pp. 134–138, 1964.

[38] O. F. Dmitriev, "An algorithm for the correction of independent errors by cyclic codes," *Probl. Peredachi Inform.,* vol. 3, pp. 102–104, 1967.

[39] G. S. Evseev, "Complexity of decoding for linear codes," *Probl. Peredachi Inform.,* vol. 19, pp. 3–8, 1983.

[40] L. D. Baumert, R. J. McEliece, and G. Solomon, "Decoding with multipliers," JPL Deep Space Network Progress Report, 42–34, pp. 42–46, 1976.

[41] G. C. Clark and J. B. Cain, *Error-Correcting Coding for Digital Communications.* New York, Plenum Press, 1981.

[42] A. H. Chan and R. A. Games, "(n,k,t)-covering systems and error-trapping decoding," *IEEE Transactions on Information Theory,* vol. IT-27, pp. 643–646. 1981.

[43] P. G. Farrell, M. Rice, and F. Taleb, "Minimum weight decoding for cyclic codes," Proc. IMA Conf. on Cryptography and Coding, Cirencester, 1986.

[44] R. M. F. Goodman and A. D. Green, "Microprocessor-controlled permutation decoding of error-correcting codes," Proc. IERE Conf. on Microprocessors in Automation and Communications, Kent, No. 41, pp. 365–376, 1978.

[45] D. M. Mandelbaum, "On vote-taking and complete decoding of certain error-correcting codes," *Information and Control,* vol. 43, pp. 195–197, 1979.

[46] D. M. Mandelbaum, "On complete decoding of linear error-correcting codes," *Information and Control,* vol. 47, pp. 195–200, 1980.

[47] V. K. Wei, "An error-trapping decoder for nonbinary cyclic codes," *IEEE Transactions on Information Theory,* vol. IT-30, pp. 538–541, 1984.

[48] T. J. Goblick, Jr., "Coding for a discrete information source with a distortion measure," Ph.D. dissertation, Dept. of Elec. Eng., M.I.T., Cambridge, Mass., 1962.

[49] J. T. Coffey and R. M. F. Goodman, "Any code of which we cannot think is good," submitted to *IEEE Transactions on Information Theory.*

[50] G. J. Chaitin, "Information-theoretic computational complexity," *IEEE Transactions on Information Theory,* vol. IT-20, pp. 10–15, 1974.

[51] J. N. Pierce, "Limit distribution of the minimum distance of random linear codes," *IEEE Transactions on Information Theory,* vol. IT-13, pp. 595–599, 1967.

[52] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Transactions on Information Theory,* vol. IT-18, pp. 170–182, 1972.

[53] J. K. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis," *IEEE Transactions on Information Theory,* vol. IT-24, pp. 76–80, 1978.

[54] P. G. Farrell, M. Rice, F. Taleb, "Division algorithms for hard and soft decision decoders," Proc. Int'l. Conf. on Digital Signal Processing, Florence, Italy, 1987.

[55] W. Godoy, Jr. and D. S. Arantes, "Sub-optimum soft-decision decoding of block codes using the zero-neighbors algorithm," *IEEE Int'l. Symposium on Information Theory,* Kobe, Japan, 1988.

[56] J. T. Coffey, R. M. F. Goodman, and P. G. Farrell, "Mapping Vector Decoding," Proc. Second Conference on Error-Correcting Codes, IBM Research Center, Almaden, California, 1987.

[57] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes.* Cambridge, MA: MIT Press, 1972.

[58] R. Lidl and H. Niederreiter, *Finite Fields.* Reading, MA: Addison-Wesley, 1983.

[59] B. K. Dass, "A sufficient bound for codes correcting bursts with weight constraints," *J. ACM,* vol. 22, no. 4, pp. 501–503, 1975.

[60] S. N. Gupta, "On low-density-burst correcting linear codes," *Journal of Combinatorics, Information and System Sciences,* vol. 4, no. 2, pp. 219–226, 1979.

[61] A. D. Wyner, "On coding and information theory," *SIAM Review*, vol. 11, no. 3, pp. 317–346, 1969.

[62] J. D. Bridwell and J. K. Wolf, "Burst distance and multiple-burst correction," *Bell Syst. Tech. J.,* vol. 49, no. 5, pp. 889–909, 1970.

[63] T.-Y. Hwang, "Decoding linear block codes for minimizing word error rate," *IEEE Transactions on Information Theory,* vol. IT–25, no. 6, pp. 733-737, 1979.

[64] E. R. Berlekamp, "The technology of error-correcting codes," *Proc. IEEE,* vol. 68, no. 5, pp. 564–593, 1980.

[65] V. N. Koshelev, "On some properties of random group codes of great length," *Probl. Peredachi Inform.,* vol. 1, no. 4, pp. 35–38, 1965.

[66] M. V. Kozlov, "The correcting capacities of linear codes," *Soviet Physics — Doklady,* vol. 14, no. 5, pp. 413–415, 1969.

[67] M. Deza, F. Hofmann, "Some results related to generalized Gilbert-Varshamov bounds," *IEEE Transactions on Information Theory,* vol. IT–23, no. 4, pp. 517–518, 1977.

[68] R. J. McEliece, "On the symmetry of good nonlinear codes," *IEEE Transactions on Information Theory,* vol. IT–16, no. 5, pp. 609–611, 1970.

[69] J. Wolfmann, "A permutation decoding of the (24, 12, 8) Golay code," *IEEE Transactions on Information Theory,* vol. IT–29, no. 5, pp. 748–750, 1983.

[70] K. R. Matis, J. W. Modestino, "Reduced-search soft-decision trellis decoding of linear block codes," *IEEE Transactions on Information Theory,* vol. IT–28, no. 2, pp. 349–355, 1982.

[71] R. T. Chien and D. T. Tang, "On definitions of a burst," *IBM J. Res. Develop.,* vol. 9, no. 4, pp. 292–293, 1965.

[72] J. Fang, G. Cohen, P. Godlewski, and G. Battail, "On the inherent intractability of soft decision decoding of linear codes," Proceedings of the 2nd International Colloquium on Coding Theory and Applications, Cachan-Paris, France, November 1986. Lecture Notes in Computer Science, vol. 311. Berlin: Springer-Verlag, 1987.

[73] D. M. Jones and J. J. Bussgang, "Tree-like structure of block codes," *IEEE Transactions on Information Theory,* vol. IT–8, no. 5, pp. 384–385, 1962.

[74] G. Landsberg, "Über eine Anzahlbestimmung und eine damit zusammenhängende Reihe," *J. Reihe Angew. Math.,* vol. 111, pp. 87–88, 1893.

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

[76] R. J. McEliece, "A public-key cryptosystem based on algebraic coding theory," JPL Deep Space Network Progress Report, 42–44, Jet Propulsion Laboratory, 1978.

[77] E. F. Brickell and A. M. Odlyzko, "Cryptanalysis: a survey of recent results," *Proc. IEEE,* vol. 76, no. 5, pp. 578–593, 1988.

[78] L. B. Levitin, "A new minimum distance decoding algorithm for general linear codes," *IEEE International Symposium on Information Theory,* Kobe, Japan, June 1988.

[79] D. S. Johnson, "The NP-completeness column: an ongoing guide," *J. Algorithms,* vol. 3, pp. 182–195, 1982.

[80] D. S. Johnson, "The NP-completeness column: an ongoing guide," *J. Algorithms,* vol. 5, pp. 433–447, 1984.

[81] D. S. Johnson, "The NP-completeness column: an ongoing guide," *J. Algorithms,* vol. 6, pp. 291–305, 1985.

[82] S. C. Ntafos and S. L. Hakimi, "On the complexity of some coding problems," *IEEE Transactions on Information Theory,* vol. IT–27, no. 6, pp. 794–796, 1981.

[83] G. Solomon and H. C. A. van Tilborg, "A connection between block and convolutional codes," *SIAM J. Appl. Math.,* vol. 37, no. 2, pp. 358–369, 1979.

[84] D. M. Gordon, "Minimal permutation sets for decoding the binary Golay codes," *IEEE Transactions on Information Theory,* vol. IT–8, no. 3, pp. 541–543, 1982.

[85] B. L. Montgomery, H. Diamond and B. V. K. Vijaya Kumar, "A general minimum distance decoding procedure for binary linear block codes," *IEEE International Symposium on Information Theory,* Ann Arbor, Michigan, September 1986.

[86] J. K. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis," *IEEE Transactions on Information Theory,* vol. IT–24, no. 1, pp. 76–80, 1978.

[87] H. H. Ma and J. K. Wolf, "On tail-biting convolutional codes," *IEEE Transactions on Communications,* vol. COM-34, no. 2, pp. 104–111, 1986.

[88] J. K. Omura, "Iterative decoding of linear codes by a modulo-2 linear program," *Discrete Math.,* vol. 3, pp. 193–208, 1972.

[89] M. Bossert and F. Hergert, "Hard- and soft-decision decoding beyond the half minimum distance — an algorithm for linear codes," *IEEE Transactions on Information Theory,* vol. IT–32, no. 5, pp. 709–714, 1986.

[90] J. N. Franklin, *Methods of Mathematical Economics.* New York: Springer-Verlag, 1980.

[91] C. R. P. Hartmann and L. D. Rudolph, "An optimum symbol-by-symbol decoding rule for linear codes," *IEEE Transactions on Information Theory,* vol. IT–22, pp. 514–517, Sept. 1976.

[92] H. Greenberger, "An iterative algorithm for decoding block codes transmitted over a memoryless channel," JPL Deep Space Network Progress Report 42–47, Jet Propulsion Laboratory, 1978.

[93] J. L. Massey, *Threshold Decoding.* Cambridge Massachusetts: MIT Press, 1963.

[94] A. C. Yao, "Theory and applications of trapdoor functions," in "Proceedings, 23rd Annual Symposium on Foundations of Computer Science," pp. 80–91, IEEE Computer Society, Los Angeles, 1982.

[95] L. D. Rudolph, " A class of majority-logic decodable codes, *IEEE Transactions on Information Theory,* vol. IT–13, pp. 305–307, 1967.

[96] L. D. Rudolph, "Threshold decoding of cyclic codes," *IEEE Transactions on Information Theory,* vol. IT–15, pp. 414–418, 1969.

[97] S. M. Reddy and J. P. Robinson, "A construction for convolutional codes using block codes," *Information & Control*, vol. 12, 55–70, 1968.

[98] W. W. Peterson, "On the weight structure and symmetry of BCH codes," Contr. Rep. AFCRL–65–515, July 1968.

[99] G. D. Cohen, "A nonconstructive upper bound on covering radius," *IEEE Transactions on Information Theory,* vol. IT–29, no. 3, pp. 352–353, 1983.

[100] P. Delsarte and P. Piret, "Do most binary linear codes achieve the Goblick bound on the covering radius?" *IEEE Transactions on Information Theory,* vol. IT–32, no. 6, pp. 826–828, 1986.

[101] P. Erdös and J. Spencer, *Probabilistic Methods in Combinatorics,* New York: Academic Press, 1974.

[102] L. B. Levitin, "Covering radius of almost all linear codes is asymptotically equal to the Goblick bound," IEEE International Symposium on Information Theory, Ann Arbor, Michigan, October 1986.

[103] J. K. Omura, "A probabilistic decoding algorithm for binary group codes," IEEE International Symposium on Information Theory, 1969.

[104] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423 and 623–656, 1948.

[105] Y. Abu-Mostafa, *Information and Complexity,* to be published.

[106] J. Bruck and M. Naor, "The hardness of decoding linear codes with preprocessing," preprint, 1988.