The Complexity of Error-Correcting Codes

Daniel A. Spielman[1]

Massachusetts Institute of Technology

Abstract. By concatenating linear-time codes with small, good codes, it is possible to construct in polynomial time a family of asymptotically good codes that approach the Shannon bound that can be encoded and decoded in linear time. Moreover, their probability of decoder error is exponentially small in the block length of the codes. In this survey, we will explain exactly what this statement means, how it is derived, and what problems in the complexity of error-correcting codes remain open. Along the way, we will survey some key developments in the complexity of error-correcting codes.

1 Introduction

Error-correcting codes are the means by which we compensate for the corruption that occurs in communication over imperfect channels. In a coding system, a message first passes through an *encoder*, which transforms it into a *codeword*; this *codeword* is then transmitted over the channel. The channel modifies the codeword by adding *noise*, so that the *received word* received by the receiver may differ from the codeword that was transmitted. The received word is processed by a *decoder*, which uses the received word to guess which codeword was transmitted, and outputs its guess. Much of the research on error-correcting codes is devoted to improving the trade-off between the probability that the decoder's guess is correct and the complexity of the encoders and decoders.

In this survey, we examine the software complexity[1] of this problem from an asymptotic perspective. We will restrict our attention to communication over the binary symmetric channel (BSC), as it seems natural to most computer scientists.[2] A channel is binary if it allows the transmission of only two symbols, which we take to be 0 and 1. The *binary symmetric channel* with error-probability p (BSCp) is the channel that

[1] We should point out that, at the present time, almost every implementation of error-correcting codes uses special-purpose hardware. Moreover, coding schemes that are efficient in software can be inefficient in hardware, and *vice versa*.

[2] Results similar to those in this survey may be obtained for any reasonable memoryless channel.

transmits one bit at a time and flips the value of that bit with probability p. If it does not flip a bit, then it transmits the bit correctly. For each bit transmitted by the channel, the probability that it is flipped is independent of the others.

In the rest of this introduction, we will review the definitions needed to describe the types of error-correcting codes we will use and then state the complexity and coding problems we will consider. We will conclude the introduction with an overview of the rest of the paper.

The framework presented in this survey builds on those developed in [4] and [29,30]. Most of the material in Sections 2 and 3 can be found in those papers. The material in Section 2 is standard, and we refer the reader to a reference such as [21], [40], or [3] for a more thourough treatment. We also recommend the recent survey by Vardy [41]. For references on error-correcting codes that emphasize an engineer's perspective, we recommend [26], [42] and [6].

1.1 Coding Definitions

A *code* is an injective mapping from strings to strings. We will consider families of block codes over the alphabet $\{0,1\}$. A binary *block code* is one whose messages are strings over $\{0,1\}^m$ and whose codewords are strings over $\{0,1\}^n$, for some $n \geq m$; the *length* of the code is n, and its *rate* is n/m. The strings of $\{0,1\}^m$ are the possible messages, and we assume that each is equally likely.[3] Each image of a string from $\{0,1\}^m$ is a *codeword*, and the word "code" is sometimes used to refer just to the set of codewords. An *encoder* maps a message to its codeword, and a *decoder* maps a word in $\{0,1\}^n$ to either a codeword or a message indicating that it cannot decide on a codeword.

A *family* of codes can be defined as an infinite sequence of codes, each of a different length, indexed by their length. A *code constructor* for a family of codes is a device that takes as input a block length, and outputs a description of an encoder and a decoder for the code in the family of that length. We do not insist that this decoder be the best possible decoder for the resulting code, as we will measure the quality of the coding system by the number of errors that the decoder can actually correct. By measuring the complexity of the code constructor, we allow ourselves

[3] The task of adjusting one's message space so that each message is equally likely is that of *compression*. Compression schemes need to take advantage of the special structure of the data to which they are applied. We would like to avoid such concerns. In situations in which error-free communication is desired, it is reasonable to assume that data has been compressed before it is encoded with an error-correcting code.

to consider the non-uniform complexity of the encoder and decoder. We separate the complexity of constructing the encoders and decoders from the complexities of these devices themselves because this is how they are used: the same encoders and decoders are used many times to transmit messages of the same block length. In particular, a long message is usually broken up into a sequence of blocks which are then transmitted separately. It would be reasonable to change the definition of a family of codes so that it allows additional parameters, such as the rate of the code and perhaps a quality parameter; in this case, the complexity of the constructor should also be given in terms of these parameters.

Systematic codes are a particularly convenient class of codes. In a *systematic code*, the message appears as a substring of the codeword. Thus, we can divide the bits of a codeword into *message bits* and *check bits*, where the message bits contain the message and the encoder need only compute the check bits from the message bits. All the codes that we consider will be *linear codes*, and we will see that all such codes can easily be made systematic. Linear codes are codes in which the alphabet is a field, such as the field $GF(2)$, and the encoding function is a linear map. Thus, the codewords form a vector space.

1.2 Coding Problems

Intuitively, if an error-correcting code is going to provide strong protection against errors, then most codewords should be far from every other codeword. Our notion of distance between two words is the *Hamming distance*, the number of bits in which the words differ. The decoding algorithm for transmission over the BSC that achieves the minimal probability of error will map a received word to the codeword to which it is closest. This approach is known as *maximum likelihood decoding*, as it selects the codeword that maximizes the probability that the channel will output the received word. If there are codewords $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ that differ in only k places, and if $\boldsymbol{w}_1$ is transmitted and more than $k/2$ of those bits are flipped, then the optimal decoder will return $\boldsymbol{w}_2$. On a BSC with error probability p, the probability of this happening is at least

$$\binom{k}{(k+1)/2} p^{(k+1)/2} (1-p)^{(k-1)/2},$$

for k odd.

This observation led to the definition of the *minimum distance* of a code: the minimum distance achieved by a pair of codewords in the code.

If $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, and $\boldsymbol{v}$ are codewords in a linear code, then

$$d(\boldsymbol{w}_1, \boldsymbol{w}_2) = d((\boldsymbol{w}_1 - \boldsymbol{v}), (\boldsymbol{w}_2 - \boldsymbol{v})) = d(\mathbf{0}, (\boldsymbol{w}_2 - \boldsymbol{w}_1)),$$

and $\boldsymbol{w}_2 - \boldsymbol{w}_1$ is a codeword; so, the minimum distance of the code is equal to the minimum weight of a codeword, where the *weight* of a codeword is equal to its distance from the all-0 word. It is possible to construct families of codes in which the rate remains constant while the minimum distance grows linearly with the block length. Such codes are called *asymptotically good*, and we measure their *minimum relative distance*—their minimum distance divided by their block length.

Sufficiently long codes from an asymptotically good family can provide excellent error-protection: the maximum likelihood decoder can correct any number of errors up to half the minimum distance. Moreover, if one uses an asymptotically good code to transmit over a channel whose error probability is less than the relative minimum distance of the code, then the probability of decoding error decreases exponentially in the block length of the code. Since the rate of the code remains constant, the communication overhead of the code does not change–one just divides the communication into longer blocks. Of course, the computation time of most decoders will increase with the block length. These observations lead to a natural complexity of coding problem: to design asymptotically good codes that have fast encoding, decoding, and constructing algorithms, where the complexity is measured in terms of the number of message bits in the codes. In this case, we consider the number of errors that the proposed decoder can correct rather than the number of errors that the maximum likelihood decoder can correct, as this provides a more accurate measure of the quality of the resulting coding system. Of course, one should consider the tradeoff between the rate and minimum distance of the codes as well, although this affects the length of the transmission more than the complexity of the algorithms. Bassalygo, Zyablov, and Pinsker [4] approached versions of this problem using random linear codes, Reed-Solomon codes, and some graph-theoretic codes (from [12]).

A more natural problem is to measure the average-case performance of a coding system. Rather than measure the maximum number of errors that the decoder can correct, we will measure the number of errors that the decoder can usually correct when the errors are chosen at random. In his paper, "A Mathematical Theory of Communication", Shannon [32] proved that every channel has a fixed *capacity*, which is a rate beyond which it is not possible to communicate reliably. Moreover, he demonstrated that for every rate less than the capacity of the channel, it is

possible to transmit information with arbitrarily small error if one uses codes of sufficiently long block length. Thus, we are presented with a natural complexity problem, first posed by Savage [31]: given a binary symmetric channel with error probability p, and an error-tolerance ϵ, find the coding system[4] of least complexity that enables one to communicate over the channel so that the probability that the output of the decoder is incorrect is at most ϵ.

Neither of the previous problem statements have compared the rate of the code produced with the optimal rate possible for the channel. The rate of the code is important as it dictates how much redundancy occurs in the transmission. As the time required to transmit the message across the channel may dominate the time of the computation, such factors cannot be ignored. Moreover, part of the channel may be a network, in which case an unnecessarily long transmission may slow the transmissions of others using the network. Thus, our analyses should include a comparison of the rate of our codes with the best rate possible. The existence of the Shannon bound makes this reasonable in the analysis of the average-case performance of a coding system. Shannon's bound states that the capacity of the BSCp is $1-H(p)$, where $H(x) = -x\log_2 x-(1-x)\log_2(1-x)$ is the binary entropy function. It is trickier to examine the tradeoff between the rate and minimum relative distance of a code because the best tradeoff achieved by known codes, the Gilbert-Varshamov bound, does not meet the best known upper bound, the Linear Programming Bound [24]. Most linear codes meet the Gilbert-Varshamov bound, which means that their rate r and relative minimum distance δ satisfy $r \geq 1 - H(\delta)$. As we are unaware of any recent advances in the construction of binary codes that approach the Gilbert-Varshamov bound, we will not have to worry about this problem in this survey.

All the complexity statements in this paper are designed to hold for the unit-cost model of a RAM (see [1]). We have verified that the statements of Section 5 also hold for Pointer Machines (see [16]) and, with minor modifications of the coding system such as those outlined in [34], for the log-cost model of a RAM [1] as well.

1.3 Overview

In Section 2, we examine the random linear codes introduced by Elias [7] and point out that they meet both the Gilbert-Varshamov and Shannon bounds. We then consider the Wozencraft ensemble, a smaller family of

[4] Actually, Savage just considered the complexity of the decoder.

linear codes with similar performance, and show that its complexity is polynomially related with its probability of decoder error. In Section 3, we introduce Forney's [9] method of concatenating codes to obtain low-complexity codes that approach the Shannon bound and have a probability of decoder error that is almost exponential in their complexity. Section 4 provides a very brief overview of constructions of codes based on sparse bipartite graphs. These have been used to obtain coding systems in which the probability of decoder error is exponential in the complexity of encoding and decoding. In Section 5, we concatenate the codes of Section 2 with the codes of Section 4 to obtain codes that do meet the Shannon bound and have a probability of decoder error that is exponential in the complexity of encoding and decoding. We conclude by pointing out that we have not analyzed the complexities of these codes in terms of how much computation is required to obtain a given probability of decoder error at a given distance from the Shannon bound. We argue that this is an important measure of the performance of a coding system, and that more work on this problem is needed.

2 Random Linear Codes

A *linear code* is a subspace of $GF(2)^n$. One can show that, with high probability, a linear code chosen uniformly at random probably meets the Gilbert-Varshamov bound. Similarly, for any rate $r < 1 - H(p)$, most linear codes of rate r enable the maximum likelihood decoder to achieve an error probability that decreases exponentially in the block length when communicating over the BSCp.

A linear code is usually described by its *generator matrix* or its *check matrix*. The generator matrix of a linear code of length n and rate r is a matrix over $GF(2)$ with rn rows and n columns such that the words of the code are precisely the linear combinations of the rows of the matrix. The check matrix of such a code is a matrix of rank $(1-r)n$ with $(1-r)n$ rows and n columns such that the codewords are exactly those words that have inner product zero with each row of this matrix. The check matrix is sometimes called the *parity check matrix* since each row indicates a subset of the message bits whose parity must be zero in order for a word to be a codeword. Note that elementary row operations do not change the code defined by a generator or check matrix. Since the choices for these matrices are not unique, one should presume that some choices will lead to simpler representations of the code than others.

Encoding: By performing row operations on the check matrix of a linear code, one can obtain a check matrix that has the $(1-r)n$-by-$(1-r)n$ identity matrix as a submatrix. Using this check matrix, one can see that a linear code can be made systematic: the bits corresponding to the columns in the identity matrix can be labeled the check bits, and the other rn bits can be used as the message bits. For any setting of the rn message bits, there is exactly one setting of the check bits that produces a word that has inner product 0 with each row of this check matrix; moreover, these check bits can be computed by multiplying the vector of rn message bits by the $(1-r)n$-by-rn submatrix of the check matrix corresponding to the columns of the message bits. Thus, the code constructor could construct[5] a matrix from which the encoder could encode a message in quadratic time.

Code Construction: If we choose a linear code at random, it is very unlikely that its performance will be much worse than that of most linear codes. However, we know of no efficient algorithm for certifying that a particular linear code will perform well, even assuming that we are going to use a maximum likelihood decoder. In general, computing the minimum distance of a binary code is NP-hard (see [41]), and we see no reason that it should be easier for a randomly chosen code. In fact, we know of no algorithm for approximating the minimum distance of a linear code that is substantially faster than enumerating all low-weight words and then checking whether they are codewords. However, we can find explicit constructions of linear codes that meet the Gilbert-Varshamov bound in less time than that taken by the naive algorithm that enumerates all linear codes. An algorithm for doing this was presented in [4]. We will present a different approach in Section 2.1.

Decoding: The best known decoding algorithms for arbitrary linear codes require time exponential in the lengths of the codewords. One algorithm is to enumerate all codewords, compute the distance of each from the received word, and output a word closest to the received word.

If one is willing to make a decoder that uses a lot of space, then one can speed up this algorithm by constructing a giant look-up table indexed by every possible received word. The entry corresponding to a received word would be the codeword closest to it. This results in a table with 2^n entries. To obtain a smaller but almost as useful table, one can use *syndrome decoding*. The *syndrome* of a received word $\boldsymbol{w}$ is the vector obtained by

[5] A naive algorithm will produce this matrix in time $O(n^3)$. An asymptotic advantage can be obtained by observing that the time required is asymptotically the same as the time required for matrix multiplication (see [15] or [5, Section 16.4]).

computing the inner product of $\boldsymbol{w}$ with each row of the check matrix. Note that an error pattern produces the same syndrome regardless of the codeword to which it is added. Thus, if one associates to each syndrome a minimal weight error-pattern that produces this syndrome, then one can perform maximum-likelihood decoding of a linear code by mapping each received word to the codeword obtained by adding the error pattern associated with the received word's syndrome. Such a list of error patterns associated with syndromes has $2^{(1-r)n}$ entries of length at most n, and can be produced in time at most $2^n * n^2$ by enumerating all possible error-patterns. If one just desires an exponentially decreasing probability decoding error when using a BSC with error-probability p, then one can speed up this algorithm by only enumerating over error-patterns of weight at most $p + \epsilon$ for some small $\epsilon > 0$.

Using syndrome decoding, one can check whether a given code meets the Shannon bound in time $O(2^{(1-r+\epsilon)n})$, for any $\epsilon > 0$. First, construct the syndrome table for all error-patterns of weight at most $(p + \epsilon')n$ in time $2^{(1-r+\epsilon)n}n^2$. Then, for each error pattern of weight at most $(p + \epsilon')n$, compute its syndrome and check whether the syndrome decoding algorithm would decode it to the the $\mathbf{0}$ word. Using this information, one can estimate the probability of decoding error of the syndrome decoding algorithm.

2.1 The Wozencraft Ensemble

The Wozencraft ensemble[6] is a set of codes much smaller than the set of linear codes, but from which a randomly chosen code has an error tolerance similar to that of a linear code chosen uniformly at random. The property of the Wozencroft ensemble that gives its codes their power is that a every word appears as a codeword in an equal number of codes in the ensemble. The proof that this condition suffices is almost identical to the proofs that a random linear code meets the Gilbert-Varshamov and Shannon bounds.

Theorem 1. *Let S be a set of codes of length n and rate r such that $|\{C \in S : w \in C\}|$ does not depend on w. Then,*

1. a code chosen at random from S probably meets the Gilbert-Varshamov bound, and

[6] The Wozencraft ensemble was presented by Massey [23] and attributed to Wozencraft.

2. a code chosen at random from S probably approaches the Shannon bound.

Proof. [**sketch**] The probability that a randomly chosen code from S contains a word of weight at most k is $\binom{n}{k}2^{-(1-r)n}$. Thus, the probability that a randomly chosen code from S contains no non-zero codeword word of weight less than δn is at least

$$2^{-(1-r)n}\sum_{i=1}^{\delta n}\binom{n}{i},$$

which becomes less than any constant as n grows large, provided that $r \leq 1 - H(\delta)$.

To prove that a code chosen at random from S probably meets the Shannon bound, we assume, without loss of generality, that the **0** word is transmitted over the BSCp. The expected number of errors produced by the channel is pn, and Chebyshev's inequality implies that the probability that it produces more than $pn+\sqrt{cpn}$ errors is less than $1/c$, for any $c \geq 1$. Assuming that at most $pn + \sqrt{cpn}$ errors are produced by the channel, the probability that there is another codeword closer than the **0** word to the received word is at most

$$2^{(1-r)n}\sum_{i=1}^{pn+\sqrt{cpn}}\binom{n}{i},$$

which becomes less than any constant as n grows large, provided that $r < 1 - H(p)$. Thus, the bound is obtained by letting c grow large and then letting n grow much more quickly.

The 1/2-rate codes of length n from the Wozencraft ensemble can be indexed by elements of the field $GF(2^{n/2})$. For each element $\alpha \in GF(2^{n/2})$, the code W_α will consist of the binary representation of all pairs $(x, \alpha x)$, where $x \in GF(2^{n/2})$, and by αx we mean α multiplied by x. Thus, x represents the message bits, and αx its check bits. It is clear that each word appears as a codeword exactly once; in fact, the word (x, y) appears in the code $W_{y/x}$. To produce codes of shorter length, we can *puncture* the code by ignoring some check bits. If we ignore c check bits, then each word will appear as a codeword 2^c times. To produce longer codes,[7] we can append more check bits by constructions such as

[7] These shorter and longer codes are different from those described by Wozencraft. We have presented them because they are easier to describe in this framework. If we use Wozencraft's original codes, we should be able to replace the r' with r in what follows.

setting $W_{\alpha,\beta}$ to be the set of words of the form $(x, \alpha x, \beta x)$, where α and β are chosen from $GF(2^{n/2})$. A simple analysis shows that these codes also meet the requirements of Theorem 1.

Encoding: The codes can be encoded using Discrete Fourier Transform based algorithms for multiplication over finite fields (see [27] and [1]). These algorithms take time $O(n \log n)$.

Decoding: We would decode these codes with the syndrome decoding algorithm presented in Section 2. This algorithm takes time $O(n^2)$ and space $O(2^{(1-r+\epsilon)n})$, provided that the code constructor spends time $O(2^{(1-r+\epsilon)n} n^2)$, for $\epsilon > 0$, to build the syndrome table.

Construction: One can find a code in these ensembles that meets the Gilbert-Varshamov bound in time $O(2^{(1-r')n} n^2)$, where r' is the greatest reciprocal of an integer that is less than r. To do this, enumerate all words of weight at most δn, where $H(\delta) = 1 - r$. For each of these words, determine the codes in which it appears. Then, choose a code in which none of these words appeared.

Finding a code in the these ensembles that meets the Shannon bound does not seem to be as easy. One approach would be to enumerate over the codes, and then choose one that meets the Shannon bound. If we test the quality of a code using the syndrome decoding approach outlined at the end of Section 2, then this algorithm takes time $2^{(2-r'-r)n}$, where r' is the greatest reciprocal of an integer that is less than r.

3 Concatention of Codes

Forney [9] introduced the method of concatenating codes to obtain low-complexity codes that approach the Shannon bound. Concatenation is a means of combining two codes, referred to as the *inner code* and an *outer code*. We'll assume that the inner code is a binary code with m_1 message bits, rate r_1, and length n_1. The outer code will be a code over an alphabet with at most 2^{m_1} symbols. If the outer code has length n_2 and rate r_2, then the concatenation of the inner code with the outer code will be a binary code of rate $r_1 r_2$ with $r_2 n_2 m_1$ message bits and length $n_1 n_2$.

To encode the concatenated code, one collects the message bits into $r_2 n_2$ symbols in the alphabet of the outer code. These symbols are treated as message symbols and encoded using the encoding algorithm for the outer code, resulting in n_2 symbols. Each of these n_2 symbols of m_1 bits each is then treated as a set of message bits and encoded using the inner code. The code is then decoded in the reverse order. Thus, the time

required to encode (decode) the concatenated code is the sum of the time required to encode (decode) the outer code and n_2 times the time required to encode (decode) the inner code. The idea behind the construction is that the length of the inner code should be small so that its complexity is small relative to n_2. On the other hand, the outer code should be a code for which correction of errors in its symbols is very simple.

As an outer code, Forney suggested the using a Reed-Solomon code. *Reed-Solomon* codes can be built with any field as their alphabet, and have length one less than the number of elements in the field. Using algorithms based on the Discrete Fourier Transform, Reed-Solomon codes can be encoded in time $O(n \log n \log \log n)$ and decoded in time $O(n \log^2 n \log \log n)$.[8] Using a new result of Pan [25], one can decode Reed-Solomon codes over prime fields in time $O(n \log n \log \log n)$. The decoding algorithm for a Reed-Solomon code of length n and rate r can correct up to $n - 2rn - 1$ errors, which is the best one could hope for.

We now sketch how one can use concatenated codes to approach the Shannon bound with low complexity. For any p, find a binary code C_1 of rate $r_1 < 1 - H(p)$ and length n_1 such that the probability of decoding error using the maximum likelihood decoder is at most ϵ. Concatenate this binary code with a with a Reed Solmon code C_2 of rate $1 - 3\epsilon$ and length n_2 over the field $GF(2^{r_1 n_1})$. If fewer than $3\epsilon/2$ of the inner code decoding operations fail, then the decoder for the Reed-Solomon code will correctly decode the entire code. One can use a Chernoff bound to show that the probability that at least $3\epsilon/2$ of the inner code decoding operations fail is exponentially small in n_2. As the rate of the concatenated code is $(1 - 3\epsilon)r_1$, we can decrease ϵ to obtain codes that approach the Shannon bound. A more careful analysis reveals that, for any rate $r < 1 - H(p)$ and for sufficiently long block lengths,[9] the probability that an inner code decoding operation fails will be $c_1^{-n_1}$ for some $c_1 > 1$. Thus, the probability that the concatenated decoder will fail will be $c_2^{-n_1 n_2}$ for some $c_2 > 1$.

Encoding: If we use a code from the Wozencroft ensemble as the inner code, then the total time of the inner code encoding operations will be $O(n_2 n_1 \log n_1)$. As $r_1 n_1 = \lceil \log n_2 \rceil$, the encoding time will be dominated by the encoding time of the outer Reed-Solomon code.

---

[8] See [14], [28], [1], [38] and [5, Chapter 3, Sections 1 and 2].

[9] Note that we must have $1 - H(p) - r < \sqrt{pn_1}$ before this analysis become applicable. Without this restriction, there is a constant probability that the number of errors will be more than the inner decoder can handle

Decoding: Assuming that the code constructor has created look-up tables to aid the decoding of the inner code, the decoding time of the concatenated code will be dominated by the time required to encode and decode the outer Reed-Solomon code. For sufficiently large n_1, we can find primes close to and less than 2^{n_1}, so we can actually use a Reed-Solomon code over a prime field as the outer code and make use of Pan's [25] improved decoding algorithm.

Construction: As the total block length is $n = n_1 n_2$ and $r_1 n_1 = \lceil \log n_2 \rceil$, the discussion in Section 2.1 yields an algorithm for choosing the inner code that takes time $O(n^{2-r-r'})$, where r' is the greatest reciprocal of an integer such that $r' < r$. This factor will dominate as it is larger than the time required to build the decoder and the time required to describe the Reed-Solomon codes.

While concatenation allows us to construct codes with good average-case performance, it does not seem to help much with worst-case performance. It would be difficult to construct new codes that beat the Gilbert-Varshamov bound by concatenation: if the inner code has minimum relative distance δ_1 and the outer code has minimum relative distance δ_2, then their concatenation could have minimum relative distance $\delta_1 \delta_2$.

3.1 Justesen Codes

Justesen [13] was the first to find an explicit construction of an asymptotically good family of error-correcting codes. Exactly what is meant by *explicit construction* is not completely clear, but it certainly excludes any construction that involves a search over small codes. Ideally, it should mean that there is a "closed form" description of the code. One might like to formalize the notion of an explicit construction by defining it in terms of the complexity of the code constructor. However, no such definition will avoid the possibility of performing a constant-size search.

Justesen's main idea was to concatenate an outer Reed-Solomon code with many different inner codes. This is, instead of using just one inner code, he would use a different inner code for each outer code symbol. In particular, he used a Reed-Solomon code constructed over a field of the form $GF(2^n)$ and the 1/2-rate Wozencraft ensemble indexed by elements of $GF(2^n)$ as the inner codes. The previous analysis proving that concatenated codes can be asymptotically good can easily be modified to show that the concatenated code will be asymptotically good provided that almost all of the inner codes are asymptotically good. While it was not Justesen's main objective, one can also show that his codes approach the Shannon bound for rate 1/2 by using as an outer code a Reed-Solomon

code of rate $1-\epsilon$, and letting ϵ slowly go to zero as the block length grows large.

4 Graph Theoretic Codes

A major advance in the development of low-complexity codes was introduced in Gallager's [11] thesis, "Low Density Parity-Check Codes". This led to the development of codes that we will refer to as *graph-theoretic codes*. We do not have the time or space to provide a good explanation of the development of graph-theoretic coding; so, we will just mention a few of the points that are relevant to this work.

Gallager [11] defined a family of linear codes by setting their check matrix to be the the bipartite adjacency matrix of a high-girth sparse bipartite graph. If this graph is chosen at random and the codes are decoded by a maximum likelihood decoder, then they will approach the Shannon bound. Gallager proved that there was a linear-time algorithm for decoding these codes that would correct some constant fraction of randomly chosen errors with probability $2^{-O(\sqrt{n})}$. As we are now beginning to use codes of sufficiently long block lengths to make Gallager's codes useful, they have become the subject of many recent papers (see, for example [10,8,20,43,44]). Tanner [39] presented an important generalization of Gallager's construction, along with generalizations of his decoding algorithms.

Zyablov and Pinsker [45] showed that, with high probability over the choice of graph, Gallager's decoder would alway correct some constant fraction of errors, regardless of where they appeared. Sipser and Spielman [34] analyzed Tanner's codes in terms of the expansion of the underlying graph and, using recent explicit constructions of expander graphs [19,22], obtained polynomial-time constructions of low-density parity check codes for which a constant fraction of errors could always be decoded in linear time. For all of the low-density parity check codes listed so far, no encoding algorithm was known that took time less than $\Omega(n^2)$, until Lafferty and Rockmore [18] discovered a $O(n^{4/3})$ time algorithm for encoding the codes of Sipser and Spielman.

In a few other cases, low-density generator matrix codes have been constructed. In these, the sparse bipartite graph is used to define a generator matrix rather than a check matrix. These are usually not used alone, but are combined in a recursive fashion with some other code. Gelfand, Dobrushin, and Pinsker [12] used such techniques to present a randomized construction of linear-time encodable codes that meet the Gilbert-

Varshamov bound. They proposed no decoding algorithm for these codes. Alon et al. [2] used low-density generator matrix codes defined by expander graphs to to obtain *uniformly constructive* low-rate codes that lie above the Zyablov bound for large alphabet sizes, and binary codes that perform better than other known constructions at very low rates. In our terms, their codes are polynomial-time constructible, where the polynomial is independent of their rate. Spielman [37] observed that, by cascading low-density generator matrix codes, any construction of low-density parity check codes can be transformed into a construction of linear-time encodable codes with a similar rate and average decoding performance as their check-matrix cousins. In [37], Spielman techniques related to those in [34] and [12] to obtain a polynomial-time construction of asymptotically good codes that can be encoded and decoded in linear time.

5 Linear-Time error-Correcting Codes

In [37,35], we presented a polynomial-time algorithm that constructs encoders and decoders for asymptotically good codes for which the encoder and decoder both run in time linear in the block length, and the number of errors that the decoder can always correct grows linearly with the block length. We will call an encoder and decoder that run in linear time a *linear-time coding system* and the code it uses a *linear-time code*. In this section, we will concatenate these codes with codes from the Wozencraft ensemble to obtain a polynomial-time constructor for a linear-time coding system that approaches the Shannon bound.

Following Remark 21 of [37], we can find a constant ζ so that for any rate[10] $r \geq 1/4$, we produce a polynomial-time constructor for a linear-time coding system in which the decoder can correct a $O((1-H^{-1}(\zeta r))^2)$ fraction of errors, where H^{-1} denotes the lesser inverse of the binary entropy function. We note that this construction itself must make use of a subcode of length $O((1-H^{-1}(\zeta r))^2)$ and minimum relative distance $O(1-H^{-1}(\zeta r))$ from some asymptotically good family. One could use a search such as those described in Sections 2 and 2.1 to find this code. If the length of the whole code is greater than an exponential in the length of this subcode, then one can use look-up tables to perform operations on this subcode so that the total time of the decoding algorithm is bounded by Cn, for some absolute constant C that does not depend on the rate of the code.

[10] In the following, we will only consider the channels BSCp where $1-H(p) \geq 1/4$, as the lower-rate cases are easier to handle.

The construction of [37] also relies on the existence of certain families of expander graphs. While it is known that for sufficiently long block lengths the required families exist, the dependence of these block lengths on r is not at present clear to this author.

5.1 Linear-Time Codes Approaching The Shannon Bound

To obtain linear-time coding systems at rates approaching the Shannon bound, we concatenate a code from an asymptotically good family of linear-time codes with a code of much smaller block length from a family that approaches the Shannon bound. To obtain a family of linear-time coding systems of rate $1-H(p)-\epsilon$ that enable reliable communication over the BSCp, we use linear-time codes of rate $\sqrt{1-H(p)-\epsilon}$ as outer codes and some fixed inner code of rate $\sqrt{1-H(p)-\epsilon}$ that has a probability of decoding error on the BSCp that is less than the fraction of errors that the outer codes can correct. We know that such a code exists by Shannon's theorem, and we can use the methods of Section 2.1 to find one.

The inner code will be fixed for the entire family. For sufficiently large block lengths, the cost of finding this inner code will become negligible and the polynomial-time cost of finding the outer code will dominate. Similarly, the code constructor will create a look-up table to be used by the encoder and decoder of the inner code, and the size of this table will become negligible when compared with the length of the outer code. Using this look-up table, the encoding and decoding algorithms for the inner code will run in linear time. As the outer code was chosen to have linear-time encoding and decoding algorithms, the encoders and decoders of the concatenated code will also run in linear time. Moreover, for each rate $r < 1$, there is a block length after which the linear time bound on these algorithms does not depend upon the rate of the code. So long as the number of inner code decoding errors is less than the number of errors that the outer code can tolerate, the concatenated decoder will decode correctly. As the probability of an inner code decoding error is less than the fraction of errors that the outer code can correct, we can use a Chernoff bound to show that the probability that the concatenated decoder will fail becomes exponentially small as the length of the outer code grows. Thus, for any rate less than the capacity of the BSCp, we can construct a coding system that has a probability of decoding failure that decreases exponentially with the amount of computation performed.

6 Conclusions

A clear deficiency of the coding systems described in Section 5.1 is that we have no bound on the block length at which they begin to exist. Also, the dependence of the decrease in decoding error probability upon the difference between the rate of the codes and the channel capacity is not clear. While we know that the decoding error probability will eventually become exponential in the block length, we have not examined what the base of the exponential will be. For a given rate r, error probability p of the channel, and decoder error probability E, we have no estimate on the least complexity of a coding system that communicates at rate r over the BSCp and has decoder error-probability E; we merely know that for fixed r and p, the complexity of the coding system eventually becomes logarithmic in the decoder error-probability. We have not even mentioned the problem that the constants hidden by the big-O are enormous because such details are usually ignored in an asymptotic analysis.

In light of these deficiencies, we propose that the average case complexity of error-correcting codes be analyzed by asking:

"Determine, in terms of p, ϵ, and E, the minimum complexity of the constructors, encoders and decoders of coding systems that communicate at rate $1 - H(p) - \epsilon$ and have decoder error probability E over the BSC with error probability p."

Much of the motivation for this problem statement comes from the fact the relation between p and E is the focus of the performance curves presented by engineers when they present error-correcting codes (see, for example [6,42]). After all, what we really want to do is find ways to communicate as quickly and reliably as possible.

References

1. A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithms.* Addison-Wesley series in computer science and information processing. Addison-Wesley, Reading, Massachusetts, 1974.
2. N. Alon, J. Bruck, J. Naor, M. Naor, and R. M. Roth. Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *IEEE Transactions on Information Theory*, 38(2):509–516, March 1992.
3. R. B. Ash. *Information Theory.* Dover Publications, Inc., New York, 1965.
4. L. A. Bassalygo, V. V. Zyablov, and M. S. Pinsker. Problems of complexity in the theory of correcting codes. *Problems of Information Transmission*, 13(3):166–175, 1977.

5. P. Bürgisser, M. Clausen, and M. Shokrollahi. *Algebraic Complexity Theory.* Springer Verlag, 1996.
6. G. C. Clark and J. B. Cain. *Error-Correction Coding for Digital Communication.* Plenum Press, New York, 1981.
7. P. Elias. Coding for noisy channels. *IRE Conv. Record*, 3:37–46, 1955.
8. B. J. Frey and F. R. Kschischang. Probability propogation and iterative decoding. In *Proc. 34-th Allerton Conference of Comm., Control, and Computing*, pages 482–493, October 1996.
9. J. G. D. Forney. *Concatenated Codes.* The MIT Press, Cambridge, Massachusetts, 1966.
10. J. G. D. Forney. The forward-backward algorithm. In *Proc. 34-th Allerton Conference of Comm., Control, and Computing*, pages 432–446, October 1996.
11. R. G. Gallager. *Low Density Parity-Check Codes.* MIT Press, Cambridge, MA, 1963.
12. S. I. Gelfand, R. L. Dobrushin, and M. S. Pinsker. On the complexity of coding. In *Second International Symposium on Information Theory*, pages 177–184, Akademiai Kiado, Budapest, 1973.
13. J. Justesen. A class of constructive asymptotically good codes. *IEEE Transactions on Information Theory*, 18:652–656, 1972.
14. J. Justesen. On the complexity of decoding Reed-Solomon codes. *IEEE Transactions on Information Theory*, 22(2):237–238, March 1976.
15. W. Keller-Gehrig. Fast algorithms for the characteristic polynomial. *Theoretical Computer Science*, 36(2–3):309–317, Apr. 1985.
16. A. Kolmogorov and V. Uspenskiĭ. On the definition of an algorithm. *Uspehi Math. Nauk*, 13(4):3–28, 1958. English translation in [17].
17. A. Kolmogorov and V. Uspenskiĭ. On the definition of an algorithm. *American Mathematical Society Translations*, 29:217–245, 1963.
18. J. D. Lafferty and D. N. Rockmore. Spectral techniques for expander codes. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 160–167, 1997.
19. A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.
20. D. J. C. MacKay and R. M. Neal. Near shannon limit performance of low-density parity-check codes. *Elect. Lett.*, to appear, 1997.
21. F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes.* North Holland, Amsterdam, 1977.
22. G. A. Margulis. Explicit group theoretical constructions of combinatorial schemes and their application to the design of expanders and concentrators. *Problems of Information Transmission*, 24(1):39–46, July 1988.
23. J. L. Massey. *Threshold Decoding.* The MIT Press, Cambridge, Massachusetts, 1973.
24. R. J. McEliece, E. R. Rodemich, H. C. Rumsey, and L. R. Welch. New upper bounds on the rate of a code via the Delsarte-MacWilliams inequalities. *IEEE Transactions on Information Theory*, 23:157–166, 1973.
25. V. Y. Pan. Faster solution of the key equation for decoding BCH error-correcting codes. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 168–175, 1997.
26. W. W. Peterson and J. E. J. Weldon. *Error-Correcting Codes.* The MIT Press, Cambridge, Massachusetts, 1972.

27. F. Preparata and D. Sarwate. Computational complexity of Fourier transforms over finite fields. *Mathematics of Computation*, 31(139):740–751, July 1977.
28. D. V. Sarwate. On the complexity of decoding Goppa codes. *IEEE Transactions on Information Theory*, 23(4):515–516, July 1977.
29. J. E. Savage. The complexity of decoders–part I: Decoder classes. *IEEE Transactions on Information Theory*, 15:689–695, November 1969.
30. J. E. Savage. The complexity of decoders–part II: Computational work and decoding time. *IEEE Transactions on Information Theory*, 17(1):77–85, January 1971.
31. J. E. Savage. *The Complexity of Computing*. John Wiley & Sons, 1976.
32. C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, July 1948.
33. M. Sipser and D. A. Spielman. Expander codes. In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science*, pages 566–576, 1994.
34. M. Sipser and D. A. Spielman. Expander codes. *IEEE Transactions on Information Theory*, 42(6):1710–1722, 1996. preliminary version appeared as [33].
35. D. A. Spielman. *Computationally efficient error-correcting codes and holographic proofs*. PhD thesis, M.I.T., May 1995. Available at `http://www-math.mit.edu/~spielman`.
36. D. A. Spielman. Linear-time encodable and decodable error-correcting codes. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 388–397, 1995.
37. D. A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6):1723–1731, 1996. preliminary version appeared as [36].
38. V. Strassen. The computational complexity of continued fractions. *SIAM J. Comput.*, 12(1):1–27, February 1983.
39. R. M. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, September 1981.
40. J. H. van Lint. *Introduction to Coding Theory*. Springer-Verlag, 1992.
41. A. Vardy. Algorithmic complexity in coding theory and the minimum distance problem. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 92–109, 1997.
42. A. J. Viterbi and J. K. Omura. *Principles of Digital Communication and Coding*. McGraw-Hill, 1979.
43. N. Wiberg. *Codes and decoding on general graphs*. PhD thesis, University of Linköping, Sweden, 1996.
44. N. Wiberg, H.-A. Loeliger, and R. Kötter. Code and iterative decoding on general graphs. *Euro. Trans. Telecommun.*, 6:513–526, 1995.
45. V. V. Zyablov and M. S. Pinsker. Estimation of the error-correction complexity of Gallager low-density codes. *Problems of Information Transmission*, 11(1):18–28, May 1976.