1 Introduction

In the everyday situation where one party wishes to communicate a message to another distant party, more often than not, the intervening communication channel is "noisy" and distorts the message during transmission. The problem of reliable communication of information over such a noisy channel is a fundamental and challenging one. *Error-correcting codes* (or simply, *codes*) are objects designed to cope with this problem. They are now ubiquitous and found in all walks of life, ranging from basic home and office appliances like compact disc players and computer hard disk drives to deep space communication.

The theory of error-correcting codes, which dates back to the seminal works of Shannon [160] and Hamming [93], is a rich, beautiful and to-date flourishing subject that benefits from techniques developed in a wide variety of disciplines such as combinatorics, probability, algebra, geometry, number theory, engineering, and computer science, and in turn has diverse applications in a variety of areas.

In this work, we study the performance of error-correcting codes in the presence of very large amounts of noise, much more than they were "traditionally designed" to tolerate. This situation poses significant challenges not addressed by the classical decoding procedures. We address these challenges with a focus on the algorithmic issues that arise therein. Specifically, we establish limits on what can be achieved in such a high-noise situation, and present algorithms for classically studied codes that decode significantly more errors than all previously known methods. We also present several novel code constructions designed to tolerate extremely large amounts of noise, together with efficient error-correcting procedures. The key technical notion underlying our work is *"List Decoding"*. This notion will be defined, and our contributions will be explained in further detail, later on in this chapter.

1.1 Basics of Error-Correcting Codes

Informally, error-correcting codes provide a systematic way of adding redundancy to a message before transmitting it, so that even upon receipt of a somewhat corrupted message, the redundancy in the message enables the receiver to figure out the original message that the sender intended to transmit.

V. Guruswami: List Decoding of Error-Correcting Codes, LNCS 3282, pp. 1-14, 2004.

The principle of redundant encoding is in fact a familiar one from everyday language. The set of all words in English is a small subset of all possible strings, and a huge amount of redundancy is built into the valid English words. Consequently, a misspelling in a word usually changes it into some incorrect word (i.e., some letter sequence that is not a valid word in English), thus enabling *detection* of the spelling error. Moreover, the resulting misspelled word quite often resembles the correct word more than it resembles any other word, thereby permitting *correction* of the spelling error. The "ispell" program used to spell-check this book could not have worked but for this built-in redundancy of the English language! This simple principle of "built-in redundancy" is the essence of the theory of error-correcting codes.

In order to be able to discuss the context and contributions of our work, we need to define some basic notions concerning error-correcting codes.[1] These are discussed below.

- Encoding. An *encoding function* with parameters k, n is a function $E : \Sigma^k \to \Sigma^n$ that maps a *message* m consisting of k symbols over some alphabet Σ (for example, the binary alphabet $\Sigma = \{0, 1\}$) into a longer, redundant string $E(m)$ of length n over Σ. The encoded string $E(m)$ is referred to as a *codeword*.
- Error-Correcting code. The *error-correcting code* itself is defined to be the image of the encoding function. In other words, it is the set of all codewords which are used to encode the various messages.
- Rate. The ratio of the number of information symbols to the length of the encoding — the quantity k/n in the above definition — is called the *rate* of the code. It is an important parameter of a code, as it is a measure of the amount of redundancy added by the encoding.
- Decoding. Before transmitting a message, the sender of the message first encodes it using the error-correcting code and then transmits the resulting codeword along the channel. The receiver gets a possibly distorted copy of the transmitted codeword, and needs to figure out the original message which the sender intended to communicate. This is done via a *decoding function*, $D : \Sigma^n \to \Sigma^k$, that maps strings of length n (i.e., noisy received words) to strings of length k (i.e., what the decoder thinks was the transmitted message).
- Distance. The *minimum distance* (or simply, *distance*) of a code quantifies how "far apart" from each other different codewords are. Define the *distance* between words as the number of coordinates at which they differ. The (minimum) distance of a code is then defined to be the smallest distance between two distinct codewords.

Historical Interlude: We now briefly discuss the history behind the definition of these concepts. The notions of encoding, decoding, and rate appeared

[1] Here we only define the most basic notions. Further definitions appear in Chapter 2.

in the work of Shannon [160]. The notions of an error-correcting code itself, and that of the distance of a code, originated in the work of Hamming [93].

Shannon proposed a *stochastic model* of the communication channel, in which distortions are described by the conditional probabilities of the transformation of one symbol into another. For every such channel, Shannon proved that there exists a precise real number, which he called the channel's *capacity*, such that in order to achieve reliable communication over the channel, one has to use a encoding function with rate less than its capacity. He also proved the converse result — namely, for every rate below capacity, there *exist* encoding and decoding schemes which can used to achieve reliable communication, with a probability of miscommunication as small as one desires. This remarkable result, which precisely characterized the amount of redundancy needed to cope with a noisy channel, marked the birth of information theory and coding theory.

However, Shannon only proved the *existence* of good coding schemes at any rate below capacity, and it was not at all clear how to perform the required encoding or decoding efficiently. Moreover, the stochastic description of the channel did not highlight any simple criterion of when a certain code is good.

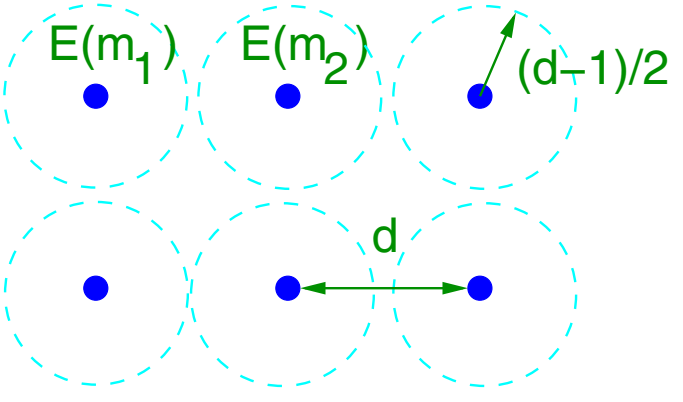

Fig. 1.1. A code of minimum distance d. Spheres of radius $(d-1)/2$ around the codewords are all disjoint.

Intuitively, a good code should be designed so that the encoding of one message will not be confused with that of another, even if it is somewhat distorted by the channel. Now, if the various codewords are all *far apart* from one another, then even if the channel distorts a codeword by a small amount, the resulting string will still resemble the original codeword much more than any other codeword, and can therefore be "decoded" to the cor-

rect codeword. In his seminal work, Hamming [93] realized the importance of quantifying how far apart various codewords are, and defined the above notion of *distance* between words, which is now appropriately referred to as *Hamming distance*. He also formally defined the notion of an error-correcting code as a collection of strings no two of which are close to each other, and defined the (minimum) distance of a code as the smallest distance between two distinct codewords. This notion soon crystallized as a fundamental parameter of an error-correcting code. Figure 1.1 depicts an error-correcting code with minimum distance d, which, as the figure illustrates, implies that Hamming balls of radius $(d-1)/2$ around each codeword are all disjoint. In this model, an optimal code is one with the largest minimum distance among all codes that have a certain number of codewords. As Figure 1.1 indicates, finding a good code in this model is a particular kind of "sphere-packing" problem. Unlike Shannon's statistical viewpoint, this combinatorial formulation permitted a variety of techniques from combinatorics, algebra, geometry, and number theory to be applied in attempts to solve the problem. In turn, this led to the burgeoning of coding theory as a discipline.

1.2 The Decoding Problem for Error-Correcting Codes

The two main algorithmic tasks associated with the use of an error-correcting code are implementing the encoding function E and the decoding function D.

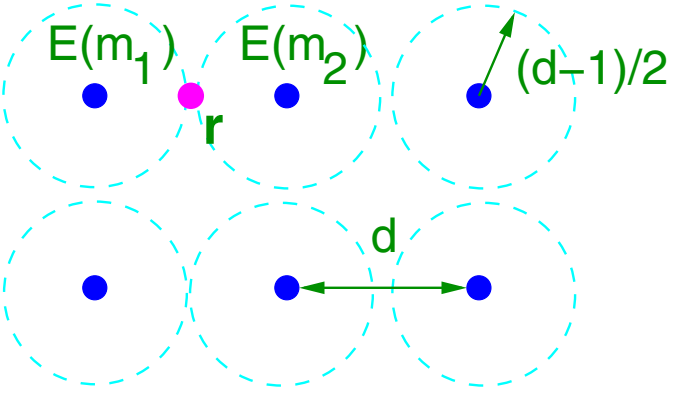

Fig. 1.2. A code of distance d cannot correct $d/2$ errors. The figure shows a received word $\mathbf{r}$ at a distance of $d/2$ from two codewords corresponding to the encodings of m_1 and m_2. In such a case, $\mathbf{r}$ could have resulted from $d/2$ errors affecting either $E(m_1)$ or $E(m_2)$.

The former task is usually easy to perform efficiently, since the "construction" of a code often works by giving such an encoding procedure.

For the decoding problem, we would ideally like $D(E(m) + \text{noise}) = m$ for every message m, and every "reasonable" noise pattern that the channel might effect. Now, suppose that the error-correcting code has minimum distance d (assume d is even) and m_1, m_2 are two messages such that the Hamming distance between $E(m_1)$ and $E(m_2)$ is d. Then, assume that $E(m_1)$ is transmitted and the channel effects $d/2$ errors and distorts $E(m_1)$ into a word $\mathbf{r}$ that is right in between $E(m_1)$ and $E(m_2)$ (see Figure 1.2). In this case, upon receiving $\mathbf{r}$, the decoder has no way of figuring out which one of m_1 or m_2 was the intended message, since $\mathbf{r}$ could have been received as a result of $d/2$ errors affecting either $E(m_1)$ or $E(m_2)$.

Therefore, when using a code of minimum distance d, a noise pattern of $d/2$ or more errors cannot always be corrected. On the other hand, for any received word $\mathbf{r}$, there can be only one codeword within a distance of $(d-1)/2$ from $\mathbf{r}$. This follows using the triangle inequality (since Hamming distance between strings defines a metric). Consequently, if the received word $\mathbf{r}$ has at most $(d-1)/2$ errors, then the transmitted codeword is the *unique* codeword within distance $(d-1)/2$ from $\mathbf{r}$ (see Figure 1.3). Hence, by searching for a codeword within distance $(d-1)/2$ of the received word, we can recover the correct transmitted codeword as long the number of errors in the received word is at most $(d-1)/2$.

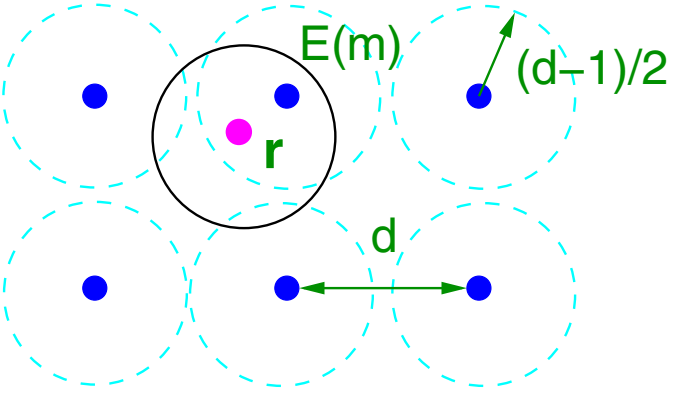

Fig. 1.3. A code of distance d can correct up to $(d-1)/2$ errors. For the received word $\mathbf{r}$, $E(m)$ is the unique codeword within distance $(d-1)/2$ from it, so if fewer than $(d-1)/2$ errors occurred, $\mathbf{r}$ can be correctly decoded to m.

Due to these facts, a well-posed algorithmic question that has been the focus of most of the classical algorithmic work on efficient decoding, is the

problem of decoding a code of minimum distance d up to $(d-1)/2$ errors. We call such a decoding *unique/unambiguous decoding* in the sequel. The reason for this terminology is that the decoding algorithm decodes only up to a number of errors for which it is *guaranteed* to find a unique codeword within such a distance of the received word.

The obvious unique decoding algorithms which search the vicinity of the received word for a codeword are inefficient and require exponential runtime. Nevertheless, a classic body of literature spanning four decades has now given efficient unique decoding algorithms for several interesting families of codes. These are among the central and most important results in algorithmic coding theory, and are discussed in detail in any standard coding theory text (eg., [132, 193]).

We are interested in what happens when the number of errors is greater than $(d-1)/2$. In such a case the unique decoding algorithms could either output the wrong codeword (i.e., a codeword other than the one transmitted), or report a decoding failure and not output any codeword. The former situation occurs if the error pattern takes the received word within distance $(d-1)/2$ of some other codeword. In such a situation, the decoding algorithm, though its answer is wrong, cannot really be faulted. After all, it found some codeword much closer to the received word than any other codeword, and in particular the transmitted codeword, and naturally places its bet on that codeword. The latter situation occurs if there is no codeword within distance $(d-1)/2$ of the received word, and it brings out the serious shortcoming of unique decoding, which we discuss below.

It is true that *some* patterns of $d/2$ errors, as in Figure 1.2, are uncorrectable due to there being multiple codewords at a distance $d/2$ from the received word. However, the situation in Figure 1.2 is quite pathological and it is actually the case that for *most* received words there will be only a single codeword that is closest to it. Moreover, the sparsity of the codewords implies that most words in the ambient space fall outside the region covered by the (disjoint) spheres of radius $(d-1)/2$ around the codewords. Together, these facts imply that most received words have a unique closest codeword (and thus it is reasonable to expect that the decoding algorithm correct them to their closest codeword), and yet unique decoding algorithms simply fail to decode them. Indeed, as Shannon's work [160] already pointed out, for "good" codes (namely, those that approach capacity), if errors happen randomly according to some reasonable probabilistic model, then with high probability the received word will not be correctable by unique decoding algorithms!

In summary, on a overwhelming majority of error patterns, unique decoding uses the excuse that there is no codeword within a distance $(d-1)/2$ from the received word to completely give up on decoding those patterns. This limitation is in turn due to the requirement that the decoding *always* be unique or unambiguous, which, as argued earlier, means there are some (pathological) patterns of $d/2$ errors which are not correctable. It turns out that there is

a meaningful relaxation of unique decoding which circumvents this predicament and permits one to decode beyond the perceived "half-the-distance barrier" faced by unique decoding. This relaxed notion of decoding, called *list decoding*, is the subject of this book, and we turn to its definition next.

1.3 List Decoding

1.3.1 Definition

List decoding was introduced independently by Elias [48] and Wozencraft [199] in the late 50's. List decoding is a relaxation of unique decoding that allows the decoder to output a *list* of codewords as answers. The decoding is considered successful as long as the codeword corresponding to the correct message is included in the list. Formally, the list decoding problem for a code $E : \Sigma^k \to \Sigma^n$ is defined as follows: Given a received word $\mathbf{r} \in \Sigma^n$, find and output a list of all messages m such that the Hamming distance between $\mathbf{r}$ and $E(m)$ is at most e. Here e is a parameter which is the number of errors that the list decoding algorithm is supposed to tolerate. The case $e = (d-1)/2$ gives the unique decoding situation considered earlier.

List decoding permits one to decode beyond the half-the-distance barrier faced by unique decoding. Indeed, in the situation of Figure 1.2, the decoder can simply output both the codewords that are at a distance of $d/2$ from the received word. In fact, list decoding remains a feasible notion even when the channel effects $e \gg d/2$ errors.

An important parameter associated with list decoding is the size of the list that the decoder is allowed to output. Clearly with a list size equal to one, list decoding just reduces to unique decoding. It is also undesirable to allow very large list sizes. This is due to at least two reasons. First, there is the issue of how useful a very large list is, since it is reasonable that the receiver might finally want to pick one element of the list using additional rounds of communication or using some tie-breaking criteria. Second, the decoding complexity is at least as large as the size of the list that the algorithm must output in the worst-case. Since we want efficient, polynomial time, decoding procedures, the list size should be at most a polynomial in the message length, and ideally at most a constant that is independent of the message length.

It turns out that even with a list size that is a small constant (say, 20), *any* code of distance d can be list decoded well beyond $d/2$ errors (for a wide range of distances d). The bottom line, therefore, is that allowing the decoder to output a small list of codewords as answers opens up the possibility of doing much better than unique decoding. In other words, list decoding is *combinatorially feasible*.

1.3.2 Is List Decoding a Useful Relaxation of Unique Decoding?

But the above discussion does not answer the obvious question concerning list decoding that comes to one's mind when first confronted with its definition: how useful is the notion of list decoding itself? What does one do with a list of answers, and when too many errors occur, why is receiving an ambiguous list of answers better than receiving no answer at all? We now proceed to answer these questions.

Firstly, notice that list decoding only gives more options than unique decoding. One can always go over the list output by the algorithm and check if there is any codeword within distance $(d-1)/2$ of the received word, thereby using it to perform unique decoding. But the advantage of list decoding is that it also enables meaningful decoding of received words that have no codeword within distance $(d-1)/2$ from them. As discussed earlier, since the codewords are far apart from one another and sparsely distributed, most received words in fact fall in this category. For a large fraction of such received words, one can show that in fact there is at most one codeword within distance e from them, for some bound e which is much greater than $d/2$. Therefore, list decoding up to e errors will usually (i.e., for most received words) produce lists with at most one element, thereby obviating the need of dealing with more than one answer being output! In particular, for a channel that effects errors randomly, this implies that with high probability, list decoding, when it succeeds, will output exactly one codeword.

Furthermore, if the received word is such that list decoding outputs several answers, this is certainly no worse than giving up and reporting a decoding failure (since we can always choose to return a failure if the list decoding does not output a unique answer). But, actually, it is much better. Since the list is guaranteed to be rather small, using an extra round of communication, or some other context or application specific information, it might actually be possible to disambiguate between the answers and pick one of them as the final output. For example, the "ispell" program used to spell-check this book often outputs a list of correct English words that are close to the misspelled word. The author of the document can then conveniently pick one of the words based on what he/she actually intended. As another example, consider the situation where a spacecraft transmits the encoding of, say a picture of Saturn, back to the Earth. It is possible that due to some unpredictable interference in space, the transmission gets severely distorted and the noise is beyond the range unique decoding algorithms can handle. In such a case, it might be unreasonable to request a retransmission from the spacecraft. However, if a list decoding algorithm could recover a small list of candidate messages from the received data, then odds are that only one member of the list will look anything like a picture of Saturn, and we will therefore be able to recover the original transmitted image.

Also, we would like to point out that one can always pick from the list the codeword closest to the received word, if there is a unique such codeword, and

output it. This gives the codeword that has the highest likelihood of being the one that was actually transmitted, even beyond the half-the-distance barrier. Finding such a codeword is referred to as *maximum likelihood decoding* in the literature. See the "interlude" at the end of this section for further discussion about this point, but in a nutshell, list decoding permits one to perform maximum likelihood decoding as long as the number of errors effected by the channel is bounded by the maximum number of errors that the list decoding algorithm is designed to tolerate.

Finally, error-correcting codes and decoding algorithms play an important role in several contexts outside communication, and in fact they have become fundamental primitives in theoretical computer science. In many cases list decoding enhances the power of this connection between coding theory and computer science.

Interlude: Maximum likelihood decoding (MLD) is an alternate notion of decoding considered in the literature. The goal of MLD is to output the codeword closest in Hamming distance to the received word (ties broken arbitrarily). This is considered by many to be the "holy grail" of decoding, since it outputs the codeword with the highest likelihood of being the one that was actually transmitted. MLD clearly generalizes unique decoding, since if there is a codeword within distance $(d-1)/2$ of the received, it must be the unique closest codeword. List decoding and MLD are, however, incomparable in power. List decoding can be used to perform MLD as long as the number of errors is bounded by the amount that the list decoding algorithm was designed to tolerate. In such a case, list decoding is in fact a more general primitive since it gives all close-by codewords, including the closest one(s), while a MLD algorithm rigidly makes up its mind on one codeword. On the other hand, MLD does not assume any bound on the number of errors, while list decoding, owing to the requirement of small list size in the worst-case, does. The main problem with MLD is that it ends up being computationally intractable in general, and extremely difficult to solve even for particular families of codes. In fact, the author is unaware of *any* non-trivial code family for which maximum likelihood decoding is solvable in polynomial time. In contrast, list decoding, as our work demonstrates, is algorithmically tractable for several interesting families of codes. ***End Interlude***

1.3.3 The Challenge of List Decoding

The real problem with list decoding was not that it was not considered to be useful, but that there were no known algorithms to *efficiently* list decode well beyond half-the-distance for any useful family of error-correcting codes (even though it was known that, combinatorially, list decoding offered the potential of decoding many more than $d/2$ errors using small lists). The naive brute-force search algorithms all take exponential time, and we next give some idea of why efficient list decoding algorithms have remained so elusive, despite substantial progress on efficient unique decoding.

Classical unique decoding algorithms decode only up to half-the-distance. In particular, they can never decode when more than half the symbols are in error. List decoding, on the other hand, aims to handle errors well beyond half-the-distance, and consequently, must even deal with situations where more than half the symbols are in error, including cases where *the noise is overwhelming and far out-weighs the correct information.* In fact, list decoding opens the potential of decoding when the noise is close to 100%. Realizing the potential of list decoding in the presence of such extreme amounts of noise poses significant algorithmic challenges under which the ideas used in the classical decoding procedures break down.

1.3.4 Early Work on List Decoding

The early work on list decoding focused only on statistical or combinatorial aspects of list decoding. We briefly discuss these works below. The initial works by Elias [48] and Wozencraft [199], which defined the notion of list decoding, proved tight bounds on the error probability achievable through list decoding on certain probabilistic channels. Results of a similar flavor also appear in [162, 61, 2]. Elias [49] also generalized the zero error capacity of Shannon [161] to list decoding and obtained bounds on the zero error capacity of channels under list decoding with lists of certain size.

The focus in the 80's shifted to questions of a more combinatorial nature, and considered the worst-case list decoding behavior of codes. The central works in this vein are [203, 27, 50]. These established bounds on the number of errors that could be corrected by list decoding using lists of a certain fixed size, for error-correcting codes of a certain rate. This is in the spirit of the questions that we investigate. However, none of these results presented any non-trivial list decoding algorithms.[2] Thus, despite being an extremely useful generalization of unique decoding, the potential of list decoding was largely untapped due to the lack of good algorithms.

1.4 Contributions of This Work

Our work presents a systematic and comprehensive investigation of list decoding, with a focus on algorithmic results. Presenting our contributions in sufficient detail would require several more definitions. Therefore, we only give a high level description of the contributions here, deferring a detailed discussion of the contributions of the individual chapters and how they fit together to the next chapter. Figure 1.4 gives a bird's eye view of the kind of results discussed in this monograph. The following description is probably best read with Figure 1.4 in mind.

[2] Here triviality is used to rule out both brute-force search algorithms and unique decoding algorithms.

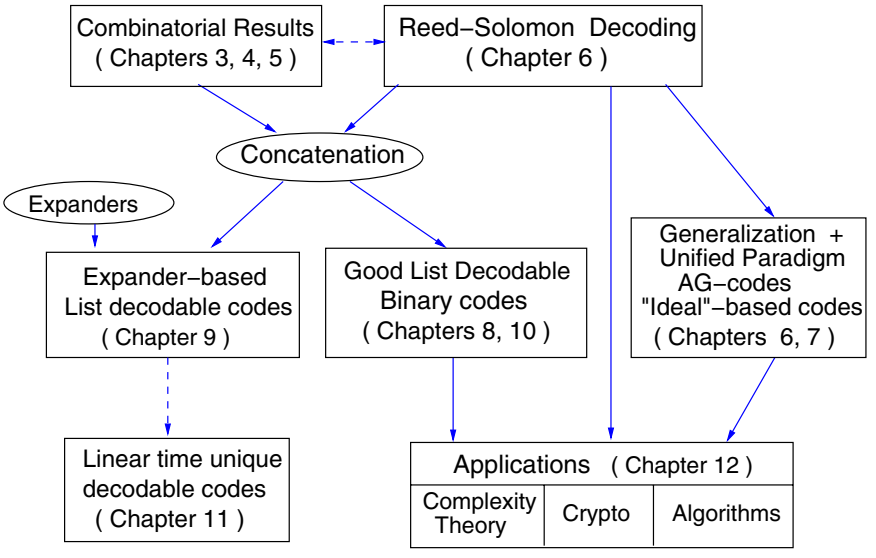

Fig. 1.4. A high level view of various chapters in this book

The first part of this monograph investigates certain combinatorial aspects of list decoding. We study the trade-offs between the list decodability of a code and the more classical parameters like rate and minimum distance. The results yield a significant sharpening of our understanding of the potential and limits of list decoding. Our combinatorial results are important in their own right and also because they set the stage for, and are repeatedly appealed to or used in, several subsequent results. The crux of this work is its algorithmic results, which comprise the second part of the thesis.

The highlight here is a list decoding algorithm for Reed-Solomon codes (Chapter 6). Reed-Solomon codes are among the most important and widely studied families of codes, and several classical unique decoding algorithms are known for them (cf. [132, Chapters 9,10]). However, despite over four decades of research, there was no known algorithm to efficiently list decode Reed-Solomon codes well beyond $d/2$ errors where d is the minimum distance. In Chapter 6, we present the *first* polynomial time list decoding algorithm that corrects more than $d/2$ errors for every value of the rate. This result builds upon an earlier breakthrough result of Sudan [178] who gave such an algorithm for Reed-Solomon codes of rate less than $1/3$. Our result list decodes up to what might well be the true list decoding potential of Reed-Solomon codes. We also generalize the algorithm to algebraic-geometric codes. The novelty of our technique enables us to also get a more general *soft* list decoding algorithm , which can take advantage of reliability information on the various symbols. This is the first non-trivial soft decoding algorithm with a provable

performance guarantee for Reed-Solomon codes since the classic 1966 work of Forney [60] on Generalized Minimum Distance (GMD) decoding.

Using our decoding algorithms for Reed-Solomon codes at the core, we also obtain several other non-trivial list decoding algorithms. These include novel algorithms for list decoding of several *concatenated codes*.[3] As a result we obtain constructions of binary codes which are efficiently list decodable from extremely large amounts of noise, and which have rate reasonably close to the best possible for such codes. (Prior to our work, there was no known construction of such codes with a positive rate, no matter how low the rate.) We also introduce novel code constructions by combining algebraic list decodable codes with "highly expanding" graphs, and thereby get new list decodable codes which improve these bounds further.

Using an expander-based construction in the same spirit as our construction for list decoding, we also get a significant improvement over a prior result for *unique decoding*. (This shows that techniques developed for list decoding also yield new insights towards solving classically studied questions like unique decoding.) Specifically, we prove that for every $\varepsilon > 0$ and $0 < r < 1$, there are *linear time encodable and decodable* codes of rate r which can be uniquely decoded up to a fraction $(1 - r - \varepsilon)/2$ of errors. By the *Singleton* bound, this fraction of errors is the best possible for unique decoding, and we are able to achieve this optimal trade-off together with linear time algorithms. By concatenation, this also gives linear-time encodable/decodable *binary* codes that (almost) match the rate of the best known polynomial time decodable constructions. In contrast, the linear time encodable/decodable binary codes known prior to this work, due to Spielman [176], could correct only a tiny fraction of errors (of the order of 10^{-6}).

List decoding, while primarily a coding-theoretic notion, has also found applications to other areas of theoretical computer science like complexity theory, cryptography, and algorithms. For these applications unique decoding does not suffice, and moreover, for several of them one needs *efficient* list decoding algorithms. In Chapter 12, we survey some of these "extraneous" applications of list decoding.

Despite its conception more than four decades ago, the long hiatus before efficient algorithms were found means that list decoding is still a subject in its infancy. This book represents the first comprehensive survey of the subject of list decoding. Inspite of its length, we have attempted a cohesive presentation that hopefully succeeds in highlighting the various aspects of list decoding and how they all fit together nicely. There is lot more work to be done on the subject, and it is our hope that this monograph will inspire at least some of it.

[3] Concatenated codes are obtained by combining two codes. The message is first encoded according to the first code, and then each symbol of the resulting codeword is encoded using the second code. A more detailed description will appear in the next chapter, specifically in Section 2.3.

1.5 Background Assumed of the Reader

This work faces the situation of having at least two audiences: computer scientists and coding theorists. Hopefully the style of our presentation will be accessible to people with either background. However, the author being a computer scientist by training, the book is probably more in line with the language and style of presentation that computer scientists are used to. The only real background required to read this book are basic algebra (comfort with finite fields and the like), some amount of probability and combinatorics, etc. Also, the focus of the bulk of the book is quite algorithmic, and hence comfort with the analysis of asymptotic complexity of algorithms would be a big plus.

Some portions of the monograph, by the very nature of the topic they discuss, are necessarily somewhat heavy on rather technical and/or algebraic matter. These include: Chapter 4 on the combinatorial limitations of list decoding, the portion of Chapter 6 that deals with algebraic-geometric codes, and Chapter 7 on the decoding of ideal-based codes. In all these cases, we have attempted to clearly state the necessary facts/theorems that we borrow from algebra. Assuming these facts the rest of the presentation should be generally accessible.

1.6 Comparison with Doctoral Thesis Submitted to MIT

Except for stylistic changes, our presentation for the large part closely follows the version of the author's doctoral dissertation that was submitted to MIT in August 2001.

The subject of list decoding has seen some significant new developments since 2001; however, we have resisted including details of these recent works beyond giving a pointer to them where appropriate (and in some cases, describing the gist of the improvement). A notable exception to this is in Chapter 11 on expander-based unique decodable codes. The results of Sections 11.4 and 11.5 were obtained in work done after the submission of the thesis [82], that improved the bounds discussed in the original version of the thesis (based on an earlier paper [81]). We chose to present the results from [82] because they not only get the "right" trade-offs, but do so with elegant techniques that are not much more difficult compared to the approach of [81]. We also removed some portions of Chapter 11 on near-linear time list-decodable codes that appeared in the thesis version, since these have since become obsolete in light of the near-linear time implementations of the Reed-Solomon list decoding algorithm [4].

Below we mention some other portions where significant revisions were done. Section 4.5 in Chapter 4 was newly added, and the contents of Section 4.6 were revised to highlight the explicit constructions used to prove the

bound in Theorem 4.9. We also added a brief section (Section 4.7.3) on an unconditional proof of tightness of Johnson bound based on the work [87].

Several portions of Chapter 10 were significantly revised based on the journal paper [78], and implicit results on erasure list decoding from the literature that were stated in the language of "Generalized Hamming Weights", are now explicitly referenced and used.