Foundations and Trends® in
Theoretical Computer Science
Vol. 2, No 2 (2006) 107–195

DOI: 10.1561/0400000007

Algorithmic Results in List Decoding

Venkatesan Guruswami

Department of Computer Science & Engineering, University of Washington
Seattle WA 98195, USA, venkat@cs.washington.edu

Abstract

Error-correcting codes are used to cope with the corruption of data by noise during communication or storage. A code uses an encoding procedure that judiciously introduces redundancy into the data to produce an associated codeword. The redundancy built into the codewords enables one to decode the original data even from a somewhat distorted version of the codeword. The central trade-off in coding theory is the one between the data rate (amount of non-redundant information per bit of codeword) and the error rate (the fraction of symbols that could be corrupted while still enabling data recovery). The traditional decoding algorithms did as badly at correcting any error pattern as they would do for the worst possible error pattern. This severely limited the maximum fraction of errors those algorithms could tolerate. In turn, this was the source of a big hiatus between the error-correction performance known for probabilistic noise models (pioneered by Shannon) and what was thought to be the limit for the more powerful, worst-case noise models (suggested by Hamming).

In the last decade or so, there has been much algorithmic progress in coding theory that has bridged this gap (and in fact nearly eliminated it for codes over large alphabets). These developments rely on an error-recovery model called “list decoding,” wherein for the pathological error patterns, the decoder is permitted to output a small list of candidates that will include the original message. This book introduces and motivates the problem of list decoding, and discusses the central algorithmic results of the subject, culminating with the recent results on achieving “list decoding capacity.”

Part I

General Literature

1

Introduction

1.1 Codes and noise models

Error-correcting codes enable reliable transmission of information over a noisy communication channel. The idea behind error-correcting codes is to *encode* the message to be transmitted into a longer, *redundant* string (called a *codeword*) and then transmit the codeword over the noisy channel. The redundancy is judiciously chosen in order to enable the receiver to *decode* the transmitted codeword even from a somewhat distorted version of the codeword. Naturally, the larger the amount of noise (quantified appropriately, according to the specific channel noise model) one wishes to correct, the greater the redundancy that needs to be introduced during encoding. A convenient measure of the redundancy is the *rate* of an error-correcting code, which is the ratio of the number of information bits in the message to the number of bits in the codeword. The larger the rate, the less redundant the encoding.

The trade-off between the rate and the amount of noise that can be corrected is a fundamental one, and understanding and optimizing the precise trade-off is one of the central objectives of coding theory. The optimal rate for which reliable communication is possible on a given noisy channel is typically referred to as “capacity.” The challenge is to construct codes with rate close to capacity, together with efficient algorithms for encoding and error correction (decoding).

The underlying model assumed for the channel noise crucially governs the rate at which one can communicate while tolerating noise. One of the simplest models is the binary symmetric channel; here the channel flips each bit independently with a certain cross-over probability p. It is well-known that the capacity of this channel equals $1 - H(p)$ where $H(x) = -x\log_2 x - (1-x)\log_2(1-x)$ is the binary entropy function. In other words, there are codes of rate up to $1 - H(p)$ that achieve probability of miscommunication approaching 0 (for large message lengths), and for rates above $1 - H(p)$, no such codes exist.

The above was a *stochastic* model of the channel, wherein we took an optimistic view that we knew the precise probabilistic behavior of the channel. This stochastic approach was pioneered by Shannon in his landmark 1948 paper that marked the birth of the field of information theory [65]. An alternate, more combinatorial approach, put forth by Hamming [46], models the channel as a jammer or adversary that can corrupt the codeword arbitrarily, subject to a bound on the total number of errors it can cause. This is a stronger noise model since one has to deal with *worst-case* or adversarial, as opposed to typical, noise patterns. Codes and algorithms designed for worst-case noise are more robust and less sensitive to inaccuracies in modeling the precise channel behavior (in fact, they obviate the need for such precise modeling!).

This survey focuses on the worst-case noise model. Our main objective is to highlight that even against adversarial channels, one can achieve the information-theoretically optimal trade-off between rate and fraction of decodable errors, matching the performance possible against weaker, stochastic noise models. This is shown for an error recovery model called *list decoding*, wherein for the pathological, worst-case noise patterns, the decoder is permitted to output a small list of candidate messages that will include the correct message. We next motivate the list decoding problem, and discuss how it offers the hope of achieving capacity against worst-case errors.

Remark 1. [Arbitrarily varying channel]

The stochastic noise model assumes knowledge of the precise probability law governing the channel. The worst-case model takes a conservative, pessimistic view of the power of the channel assuming only a limit on the total amount of noise. A hybrid model called *Arbitrarily Varying Channel* (AVC) has also been proposed to study communication under channel uncertainty. Here the channel is modeled as a jammer which can select from a family of strategies (corresponding to different probability laws) and the sequence of selected strategies, and hence the channel law, is not known to the sender. The strategy can in general vary arbitrarily from symbol to symbol, and the goal is to do well against the worst possible sequence. A less powerful model is that of the *compound channel* where the jammer has a choice of strategies, but the chosen channel law does not change during the transmission of various symbols of the codeword. AVCs have been the subject of much research – the reader can find a good introduction to this topic as well as numerous pointers to the extensive literature in a survey by Lapidoth and Narayan [54]. To the author's understanding, it seems that much of the work has been of a non-constructive flavor, driven by the information-theoretic motivation of determining the capacity under different AVC variants. There has been less focus on explicit constructions of codes or related algorithmic issues.

1.2 List decoding: Context and motivation

Given a received word **r**, which is a distorted version of some codeword **c**, the decoding problem strives to find the original codeword **c**. The natural error recovery approach is to place one's bet on the codeword that has the highest likelihood of being the one that was transmitted, conditioned on receiving **r**. This task is called *Maximum Likelihood Decoding* (MLD), and is viewed as the holy grail in decoding. MLD amounts to finding the codeword *closest* to **r** under an appropriate distance measure on distortions (for which a larger distortion is less likely than a smaller one). In this survey, we will measure distortion by the Hamming metric, i.e., the distance between two strings $x, y \in \Sigma^n$

is the number of coordinates $i \in \{1, 2, \ldots, n\}$ for which $x_i \neq y_i$. MLD thus amounts to finding the codeword closest to the received word in Hamming distance. No approach substantially faster than a brute-force search is known for MLD for any non-trivial code family. One, therefore settles for less ambitious goals in the quest for efficient algorithms. A natural relaxed goal, called *Bounded Distance Decoding* (BDD), would be to perform decoding in the presence of a bounded number of errors. That is, we assume at most a fraction p of symbols are corrupted by the channel, and aim to solve the MLD problem under this promise. In other words, we are only required to find the closest codeword when there is a codeword not too far away (within distance pn) from the received word.

In this setting, the basic trade-off question is: What is the largest fraction of errors one can correct using a family of codes of rate R? Let $C : \Sigma^{Rn} \to \Sigma^n$ be the encoding function of a code of rate R (here n is the *block length* of the code, and Σ is the *alphabet* to which codeword symbols belong). Now, a simple pigeonholing argument implies there must exist $x \neq y$ such that the codewords $C(x)$ and $C(y)$ agree on the first $Rn - 1$ positions. In turn, this implies that when $C(x)$ is transmitted, the channel could distort it to a received word $\mathbf{r}$ that is equidistant from both $C(x)$ and $C(y)$, and differs from each of them in about a fraction $(1 - R)/2$ of positions. Thus, unambiguous bounded distance decoding becomes impossible for error fractions exceeding $(1 - R)/2$.

However, the above is not a compelling reason to be pessimistic about correcting larger amounts of noise. This is due to the fact that received words such as $\mathbf{r}$ reflect a pathological case. The way Hamming spheres pack in high-dimensional space, even for p much larger than $(1 - R)/2$ (and in fact for $p \approx 1 - R$) there exist codes of rate R (over a larger alphabet Σ) for which the following holds: for *most* error patterns $\mathbf{e}$ that corrupt fewer than a fraction p of symbols, when a codeword $\mathbf{c}$ gets distorted into $\mathbf{z}$ by the error pattern $\mathbf{e}$, there will be no codeword besides $\mathbf{c}$ within Hamming distance pn of $\mathbf{z}$.[1] Thus, for typical noise

[1] This claim holds with high probability for a random code drawn from a natural ensemble. In fact, the proof of Shannon's capacity theorem for q-ary symmetric channels can be viewed in this light. For Reed–Solomon codes, which will be our main focus later on, this claim has been shown to hold, see [19, 58, 59].

patterns one can hope to correct many more errors than the above limit faced by the worst-case error pattern. However, since we assume a worst-case noise model, we do have to deal with bad received words such as **r**. List decoding provides an elegant formulation to deal with worst-case errors without compromising the performance for typical noise patterns – the idea is that in the worst-case, the decoder may output multiple answers. Formally, the decoder is required to output a list of all codewords that differ from the received word in a fraction p of symbols.

Certainly returning a small list of possibilities is better and more useful than simply giving up and declaring a decoding failure. Even if one deems receiving multiple answers as a decoding failure, as mentioned above, for many error patterns in the target noise range, the decoder will output a unique answer, and we did not have to model the channel stochastics to design our code or algorithm! It may also be possible to pick the correct codeword from the list, in case of multiple answers, using some semantic context or side information (see [23]). Also, if in the output list, there is a unique closest codeword, we can also output that as the maximum likelihood choice. In general, list decoding is a stronger error-recovery model than outputting just the closest codeword(s), since we require that the decoder output all the close codewords (and we can always prune the list as needed). For several applications, such as concatenated code constructions and also those in complexity theory, having the entire list adds more power to the decoding primitive than deciding solely on the closest codeword(s).

Some other channel and decoding models. We now give pointers to some other relaxed models where one can perform unique decoding even when the number of errors exceeds half the minimum Hamming distance between two codewords. We already mentioned one model where an auxiliary channel can be used to send a small amount of side information which can be used to disambiguate the list [23]. Another model that allows one to identify the correct message with high probability is one where the sender and recipient share a secret random key, see [53] and a simplified version in [67].

Finally, there has been work where the noisy channel is modeled as a *computationally bounded* adversary (as opposed to an all-powerful adversary), that must introduce the errors in time polynomial in the block length. This is a very appealing model since it is a reasonable hypothesis that natural processes can be implemented by efficient computation, and therefore real-world channels are, in fact, computationally bounded. The computationally bounded channel model was put forth by Lipton [56]. Under standard cryptographic assumptions, it has been shown that in the private key model where the sender and recipient share a secret random seed, it is possible to decode correctly from error rates higher than half-the-minimum-distance bound [21,48]. Recently, similar results were established in a much simpler cryptographic setting, assuming only that one-way functions exist, and that the sender has a public key known to the receiver (and possibly to the channel as well) [60].

1.3 The potential of list decoding

The number of codewords within Hamming distance pn of the worst-case received word $\mathbf{r}$ is clearly a lower bound on the runtime of any list decoder that corrects a fraction p of errors. Therefore, in order for a polynomial time list decoding algorithm to exist, the underlying codes must have the *a priori* combinatorial guarantee of being *p-list-decodable*, namely every Hamming ball of radius pn has a small number, say $L(n)$, of codewords for some polynomially bounded function $L(\cdot)$.[2] This "packing" constraint poses a combinatorial upper bound on the rate of the code; specifically, it is not hard to prove that we must have $R \leqslant 1 - p$ or otherwise the worst-case list size will grow faster than any polynomial in the block length n.

Remarkably, this simple upper bound can actually be met. In other words, for every p, $0 < p < 1$, there exist codes of rate $R = 1 - p - o(1)$ which are p-list-decodable. That is, non-constructively we can show the existence of codes of rate R that offer the potential of list decoding up to a fraction of errors approaching $(1 - R)$. We will refer to the quantity $(1 - R)$ as the *list decoding capacity.* Note that the list decoding

[2] Throughout the survey, we will be dealing with the asymptotics of a family of codes of increasing block lengths with some fixed rate.

capacity is *twice* the fraction of errors that one could decode if we insisted on a unique answer always – quite a substantial gain! Since the message has Rn symbols, information-theoretically we need at least a fraction R of correct symbols at the receiving end to have any hope of recovering the message. Note that this lower bound applies even if we somehow *knew* the locations of the error and could discard those misleading symbols. With list decoding, therefore, we can potentially reach this information-theoretic limit and decode as long as we receive slightly more than Rn correct symbols (the correct symbols can be located arbitrarily in the received word, with arbitrary noise affecting the remaining positions).

To realize this potential, however, we need an *explicit description* of such capacity-achieving list-decodable codes, and an efficient algorithm to perform list decoding up to the capacity (the combinatorics only guarantees that every Hamming ball of certain radius has a small number of codewords, but does not suggest any efficient algorithm to actually find those codewords). The main technical result in this survey will achieve precisely this objective – we will give explicit codes of rate R with a polynomial time list decoding algorithm for a fraction $(1 - R - \varepsilon)$ of errors, for any desired $\varepsilon > 0$.

The above description was deliberately vague on the size of the alphabet Σ. The capacity $1 - R$ for codes of rate R applies in the limit of large alphabet size. It is also of interest to ask how well list decoding performs for codes over a fixed alphabet size q. For the binary $(q = 2)$ case, to correct a fraction p of errors, list decoding offers the potential of communicating at rates up to $1 - H(p)$. This is exactly the capacity of the binary symmetric channel with cross-over probability p that we discussed earlier. With list decoding, therefore, we can deal with worst-case errors without any loss in rate. For binary codes, this remains a non-constructive result and constructing explicit codes that achieve list decoding capacity remains a challenging goal.

1.4 The origins of list decoding

List decoding was proposed in the late 50s by Elias [13] and Wozencraft [78]. Curiously, the original motivation in [13] for formulating list decoding was to prove matching upper and lower bounds on the decoding error probability under maximum likelihood decoding on the binary symmetric channel. In particular, Elias showed that, when the decoder is allowed to output a small list of candidate codewords and a decoding error is declared only when the original codeword is not on the output list, the average error probability of all codes is almost as good as that of the best code, and in fact almost all codes are almost as good as the best code. Despite its origins in the Shannon stochastic school, it is interesting that list decoding ends up being the right notion to realize the true potential of coding in the Hamming combinatorial school, against worst-case errors.

Even though the notion of list decoding dates back to the late 1950s, it was revived with an algorithmic focus only recently, beginning with the Goldreich–Levin algorithm [17] for list decoding Hadamard codes, and Sudan's algorithm in the mid 1990s for list decoding Reed–Solomon codes [69]. It is worth pointing out that this modern revival of list decoding was motivated by questions in computational complexity theory. The Goldreich–Levin work was motivated by constructing hard-core predicates, which are of fundamental interest in complexity theory and cryptography. The motivation for decoding Reed–Solomon and related polynomial-based codes was (at least partly) establishing worst-case to average-case reductions for problems such as the permanent. These and other more recent connections between coding theory (and specifically, list decoding) and complexity theory are surveyed in [29, 70, 74] and [28, Chapter 12].

1.5 Scope and organization of the book

The goal of this survey is to obtain algorithmic results in list decoding. The main technical focus will be on giving a complete presentation of the recent algebraic results achieving list decoding capacity. We will only provide pointers or brief descriptions for other works on list decoding.

The survey is divided into two parts. The first part (Chapters 1–5) covers the general literature, and the second part focuses on achieving list decoding capacity. The author's Ph.D. dissertation [28] provides a more comprehensive treatment of list decoding. In comparison with

[28], most of Chapter 5 and the entire Part II of this survey discuss material developed since [28].

We now briefly discuss the main technical contents of the various chapters. The basic terminology and definitions are described in Chapter 2. Combinatorial results which identify the potential of list decoding in an existential, non-constructive sense are presented in Chapter 3. In particular, these results will establish the capacity of list decoding (over large alphabets) to be $1 - R$. We begin the quest for explicitly and algorithmically realizing the potential of list decoding in Chapter 4, which discusses a list decoding algorithm for Reed–Solomon (RS) codes – the algorithm is based on bivariate polynomial interpolation. We conclude the first part with a brief discussion in Chapter 5 of algorithmic results for list decoding certain codes based on expander graphs.

In Chapter 6, we discuss folded Reed–Solomon codes, which are RS codes viewed as a code over a larger alphabet. We present a decoding algorithm for folded RS codes that uses multivariate interpolation plus some other algebraic ideas concerning finite fields. This lets us approach list decoding capacity. Folded RS codes are defined over a polynomially large alphabet, and in Chapter 7 we discuss techniques that let us bring down the alphabet size to a constant independent of the block length. We conclude with some notable open questions in Chapter 8.

2

Definitions and Terminology

2.1 Basic coding terminology

We review the terminology that will be needed and used throughout the survey. Facility with basic algebra concerning finite fields and field extensions is assumed, though we recap some basic notation and facts at the end of this Chapter. Some comfort with probabilistic and combinatorial arguments, and analysis of algorithms would be a plus.

Let Σ be a finite alphabet. An error-correcting code, or simply code, over the alphabet Σ, is a subset of Σ^n for some positive integer n. The elements of the code are referred to as *codewords*. The number n is called the *block length* of the code. If $|\Sigma| = q$, we say that the code is *q-ary*, with the term binary used for the $q = 2$ case. Note that the alphabet size q may be a function of the block length. Associated with an error-correcting code C is an encoding function $E : \{1, 2, \ldots, |C|\} \to \Sigma^n$ that maps a message to its associated codeword. Sometimes, we find it convenient to abuse notation and identify the encoding function with the code, viewing the code itself as a map from messages to codewords.

The *rate* of a q-ary error-correcting code is defined to be $\frac{\log_q |C|}{n}$. The rate measures the amount of actual information transmitted per bit of channel use. The larger the rate, less redundant the encoding. The *minimum distance* (or simply distance) of a code C is equal to the minimum Hamming distance between two distinct codewords of C. It is often convenient to measure distance by a normalized quantity in the range $[0,1]$ called the *minimum relative distance* (or simply relative distance), which equals the ratio of the minimum distance to the block length. A large relative distance bestows good error-correction potential on a code. Indeed, if a codeword is corrupted in less than a fraction $\delta/2$ of the positions, where δ is the relative distance, then it may be correctly recovered as the unique codeword that is closest to the received word in Hamming distance.

A word on the asymptotics is due. We will be interested in families of codes with increasing block lengths all of which have rate bounded below by an absolute constant $R > 0$ (i.e., the rate does not tend to 0 as the block length grows). Moreover, we would also like the relative distance of all codes in the family to be bounded below by some absolute constant $\delta > 0$. If the alphabet of the code family is fixed (independent of the block length), such code families are said to be *asymptotically good*. There is clearly a trade-off between the rate R and relative distance δ. The *Singleton bound* is a simple bound which says that $R \leqslant 1 - \delta$. This bound is achieved by certain codes (called MDS codes) over large alphabets (with size growing with the block length), but there are tighter bounds known for codes over alphabets of fixed constant size. For example, for q-ary codes, the Plotkin bound states that $R \leqslant 1 - \frac{q\delta}{q-1}$ for $0 \leqslant \delta < 1 - 1/q$, and the rate must vanish for $\delta \geqslant 1 - 1/q$, so that one must have $\delta < 1 - 1/q$ in order to have positive rate. The *Gilbert-Varshamov* bound asserts that there exist q-ary codes with rate $R \geqslant 1 - H_q(\delta)$ where $H_q(x)$ is the q-ary entropy function defined as $H_q(x) = x\log_q(q-1) - x\log_q x - (1-x)\log_q(1-x)$. The *Elias–Bassalygo* upper bound states that $R \leqslant 1 - H_q(J(\delta,q))$ where $J(\delta,q) = (1 - 1/q)\left(1 - \sqrt{1 - \frac{q\delta}{q-1}}\right)$. We will run into the quantity $J(\delta,q)$ in Chapter 3 when we discuss the Johnson bound on list decoding radius.

A particularly important subclass of codes are *linear codes*. A linear code is defined over an alphabet that is a finite field, say $\mathbb{F}$. A linear code of block length n is simply a subspace of $\mathbb{F}^n$ (viewed as a vector space over $\mathbb{F}$). The dimension of C as a vector space is called the *dimension* of the code. Note that if the dimension of C is k then $|C| = |\mathbb{F}|^k$ and the rate of C equals k/n. The encoding function of a linear code can be viewed as a linear transformation $E : \mathbb{F}^k \to \mathbb{F}^n$, where $E(\mathbf{x}) = \mathbf{G}\mathbf{x}$ for a matrix $\mathbf{G} \in \mathbb{F}^{n\times k}$ called the *generator matrix*.

2.2 Definitions for list decoding

The problem of *list decoding* a code C of block length n up to a fraction p of errors (or radius p) is the following: given a received word, output the list of all codewords $c \in C$ within Hamming distance pn from it. To perform this task in worst-case polynomial time for a family of codes, we need the *a priori* combinatorial guarantee that the output list size will be bounded by a polynomial in the block length, irrespective of which word is received. This motivates the following definition.

Definition 2.1. ((*p,L*)-list-decodability) For $0 < p < 1$ and an integer $L \geqslant 1$, a code $C \subseteq \Sigma^n$ is said to be list decodable up to a fraction p of errors with list size L, or more succinctly (p,L)*-list-decodable*, if for every $\mathbf{y} \in \Sigma^n$, the number of codewords $\mathbf{c} \in C$ within Hamming distance pn from $\mathbf{y}$ is at most L. For a function $\hat{L} : \mathbb{Z}^+ \to \mathbb{Z}^+$, and $0 < p < 1$, a family of codes is said to be $(p,\hat{L})$-list-decodable if every code C in the family is $(p,\hat{L}(n))$-list-decodable, where n is the block length of C. When the function $\hat{L}$ takes on the constant value L for all block lengths, we simply say that the family is (p,L)-list-decodable.

Note that a code being $(p,1)$-list-decodable is equivalent to saying that its relative distance is greater than $2p$. We will also need the following definition concerning a generalization of list decoding.

Definition 2.2. (List recovering) For $0 \leqslant p < 1$ and integers $1 \leqslant \ell \leqslant L$, a code $C \subseteq \Sigma^n$ is said to be (p,ℓ,L)*-list-recoverable* if for all sequences of subsets $S_1, S_2, \ldots, S_n$ with each $S_i \subset \Sigma$ satisfying $|S_i| \leqslant \ell$, there are at most L codewords $\mathbf{c} = (c_1, \ldots, c_n) \in C$ with the property that $c_i \in S_i$ for at least $(1-p)n$ values of $i \in \{1,2,\ldots,n\}$. The value

ℓ is referred to as the *input list size.* A similar definition applies to families of codes.

Note that a code being $(p,1,L)$-list-recoverable is the same as it being (p,L)-list-decodable. For (p,ℓ,L)-list-recovering with $\ell > 1$, even a noise fraction $p = 0$ leads to non-trivial problems. In this *noise-free* (i.e., $p = 0$) case, for each location we are given ℓ possibilities one of which is guaranteed to match the codeword, and the objective is to find all codewords for which this property holds. A special case of such noise-free list recovering is when ℓ codewords are given in scrambled order and the goal is to recover all of them. We will consider this toy decoding problem for Reed–Solomon codes in Chapter 4, and in fact it will be the stepping stone for our list decoding algorithms.

2.3 Useful code families

We now review some of the code constructions that will be heavily used in this survey. We begin with the class of *Reed–Solomon Codes*, which are an important, classical family of algebraic error-correcting codes.

Definition 2.3. A *Reed–Solomon code*, $\mathsf{RS}_{\mathbb{F},S}[n,k]$, is parameterized by integers n,k satisfying $1 \leqslant k \leqslant n$, a finite field $\mathbb{F}$ of size at least n, and a tuple $S = (\alpha_1,\alpha_2,\ldots,\alpha_n)$ of n *distinct* elements from $\mathbb{F}$. The code is described as a subset of $\mathbb{F}^n$ as:

$$\mathsf{RS}_{\mathbb{F},S}[n,k] = \{(p(\alpha_1),p(\alpha_2),\ldots,p(\alpha_n)) \mid p(X) \in \mathbb{F}[X] \text{ is a polynomial of degree } \leqslant k\}.$$

In other words, the message is viewed as a polynomial, and it is encoded by evaluating the polynomial at n distinct field elements $\alpha_1,\ldots,\alpha_n$. The resulting code is linear of dimension $k+1$, and its minimum distance equals $n-k$, which is the best possible for dimension $k+1$ (attains the Singleton bound).

Concatenated codes: Reed–Solomon codes are defined over a large alphabet, one of size at least the block length of the code. A simple, but powerful technique called *code concatenation* can be used to construct codes over smaller alphabets starting with codes over a larger alphabets.

The basic idea behind code concatenation is to combine two codes, an *outer* code C_{out} over a larger alphabet (of size Q, say), and an *inner* code C_{in} with Q codewords over a smaller alphabet (of size q, say), to get a combined q-ary code that, loosely speaking, inherits the good features of both the outer and inner codes. These were introduced by Forney [15] in a classic work. The basic idea is very natural: to encode a message using the *concatenated code*, we first encode it using C_{out}, and then in turn encode each of the resulting symbols into the corresponding codeword of C_{in}. Since there are Q codewords in C_{in}, the encoding procedure is well defined. Note that the rate of the concatenated code is the product of the rates of the outer and inner codes.

The big advantage of concatenated codes for us is that we can get a good list decodable code over a small alphabet (say, binary codes) based on a good list decodable outer code over a large alphabet (like a Reed–Solomon code) and a "suitable" binary inner code. The block length of the inner code is small enough to permit a brute-force search for a "good" code in reasonable time.

Code concatenation works rather naturally in conjunction with list recovering of the outer code to give algorithmic results for list decoding. The received word for the concatenated code is broken into blocks corresponding to the inner encodings of the various outer symbols. These blocks are list decoded, using a brute-force inner decoder, to produce a small set of candidates for each symbol of the outer codeword. These sets can then be used as input to a list recovering algorithm for the outer code to complete the decoding. It is not difficult to prove the following based on the above algorithm:

Lemma 2.1. If the outer code is (p_1, ℓ, L)-list-recoverable and the inner code is (p_2, ℓ)-list-decodable, then the concatenated code is $(p_1 p_2, L)$-list-decodable.

Code concatenation and list recovering will be important tools for us in Chapter 7 where we will construct codes approaching capacity over a fixed alphabet.

2.4 Basic finite field algebra

We recap basic facts and notation concerning finite fields. For any prime p, the set of integers modulo p form a field, which we denote by $\mathbb{F}_p$.

The ring of univariate polynomials in variable X with coefficients from a field $\mathbb{F}$ is denoted by $\mathbb{F}[X]$. A polynomial $f(X)$ is said to be irreducible over $\mathbb{F}$, if $f(X) = r(X)s(X)$ for $r(X), s(X) \in \mathbb{F}[X]$ implies that either $r(X)$ or $s(X)$ is a constant polynomial. A polynomial is said to be monic if its leading coefficient is 1. The ring $\mathbb{F}[X]$ has unique factorization: Every monic polynomial can be written uniquely as a product of monic irreducible polynomials.

If $h(X)$ is an irreducible polynomial of degree e over $\mathbb{F}$, then the quotient ring $\mathbb{F}[X]/(h(X))$, consisting of polynomials modulo $h(X)$, is a finite *field* with $|\mathbb{F}|^e$ elements (just as $\mathbb{F}_p = \mathbb{Z}/(p)$ is a field, where $\mathbb{Z}$ is the ring of integers and p is a prime). The field $\mathbb{F}[X]/(h(X))$ is called an *extension field* of degree e over $\mathbb{F}$; the extension field also forms a vector space of dimension e over $\mathbb{F}$.

The prime fields $\mathbb{F}_p$, and their extensions as defined above, yield all finite fields. The size of a finite field is thus always a prime power. The *characteristic* of a finite field equals p if it is an extension of the prime field $\mathbb{F}_p$. Conversely, for every prime power q, there is a unique (up to isomorphism) finite field $\mathbb{F}_q$. We denote by $\mathbb{F}_q^*$ the set of nonzero elements of $\mathbb{F}_q$. It is known that $\mathbb{F}_q^*$ is a *cyclic* group (under the multiplication operation), generated by some $\gamma \in \mathbb{F}_q^*$, so that $\mathbb{F}_q^* = \{1, \gamma, \gamma^2, \ldots, \gamma^{q-2}\}$. Any such γ is called a *primitive* element. There are in fact $\phi(q-1)$ such primitive elements, where $\phi(q-1)$ is the number of positive integers less than $q-1$ that are relatively prime to $q-1$.

We owe a lot to the following basic property of fields: Let $f(X) \in \mathbb{F}[X]$ be a nonzero polynomial of degree d. Then $f(X)$ has at most d roots in $\mathbb{F}$.

3

Combinatorics of List Decoding

In this chapter, we prove combinatorial results concerning list-decodable codes, and study the relation between the list decodability of a code and its other basic parameters such as minimum distance and rate. We will show that every code can be list decoded using small lists beyond half its minimum distance, up to a bound we call the Johnson radius. We will also prove *existential* results for codes that will highlight the sort of rate vs. list decoding radius trade-off one can hope for. Specifically, we will prove the existence of (p, L)-list-decodable codes of good rate and thereby pinpoint the capacity of list decoding. This then sets the stage for the goal of realizing or coming close to these trade-offs with explicit codes, as well as designing efficient algorithms for decoding up to the appropriate list decoding radius.

3.1 The Johnson bound

If a code has distance d, every Hamming ball of radius less than $d/2$ has at most one codeword. The list decoding radius for a list size of 1 thus equals $\left[\frac{d-1}{2}\right]$. Could it be that already at radius slightly greater than $d/2$ we can have a large number of codewords within some Hamming ball? We will prove a result called the Johnson bound which rules out such a phenomenon. It highlights the potential of list decoding with small lists up to a radius, which we call the *Johnson radius*, that is much larger than $d/2$. In turn, this raises the algorithmic challenge of decoding well-known codes such as Reed–Solomon codes up to the Johnson radius, a task we will undertake in the next chapter.

The Johnson bound is a classical bound in coding theory and is at the heart of the Elias–Bassalygo bound on rate as a function of relative distance. The Johnson bound was stated and proved in the context of list decoding, in a form similar to that presented here, in [20, Section 4.1]. A simpler geometric proof of the Johnson bound appears in [28, Chapter 3]. Here, for sake of variety, we present a combinatorial proof that has not appeared in this form before. The proof was shown to us by Jaikumar Radhakrishnan [64]. In the following, we use $\Delta(\mathbf{a},\mathbf{b})$ to denote the Hamming distance between strings $\mathbf{a}$ and $\mathbf{b}$. We use $\mathcal{B}(\mathbf{r},e)$ (or $\mathcal{B}_q(\mathbf{r},e)$ if we want to make the alphabet size explicit) to denote the Hamming ball of radius e centered at $\mathbf{r}$. For a set S, the notation $\binom{S}{2}$ stands for all subsets of S of size 2.

Theorem 3.1. (Johnson bound) Suppose $\mathbf{r} \in [q]^n$, and $\mathcal{B} \subseteq [q]^n$. Let

$$d = \mathop{\mathbf{E}}_{\{\mathbf{x},\mathbf{y}\}\in\binom{\mathcal{B}}{2}}[\Delta(\mathbf{x},\mathbf{y})];$$

$$e = \mathop{\mathbf{E}}_{\mathbf{x}\in\mathcal{B}}[\Delta(\mathbf{r},\mathbf{x})].$$

Then, $|\mathcal{B}| \leqslant \dfrac{\frac{q}{q-1}\cdot\frac{d}{n}}{\left(1-\frac{q}{q-1}\cdot\frac{e}{n}\right)^2-\left(1-\frac{q}{q-1}\cdot\frac{d}{n}\right)}$, provided the denominator is positive.

Corollary 3.2. Let $\mathcal{C}$ be any q-ary code of block length n and minimum distance $d = \left(1-\frac{1}{q}\right)(1-\delta)n$ for some $\delta \in (0,1)$. Let $e = \left(1-\frac{1}{q}\right)(1-\gamma)n$ for some $\gamma \in (0,1)$ be an integer. Suppose $\gamma^2 > \delta$. Then, for all $\mathbf{r} \in [q]^n$, $|\mathcal{B}_q(\mathbf{r},e)\cap\mathcal{C}| \leqslant \frac{1-\delta}{\gamma^2-\delta}$.

Proof. Let $\mathcal{B} = \mathcal{B}_q(\mathbf{r}, e) \cap \mathcal{C}$. Let

$$\mathop{\mathbf{E}}_{\{\mathbf{x},\mathbf{y}\} \in \binom{\mathcal{B}}{2}} [\Delta(\mathbf{x}, \mathbf{y})] = \left(1 - \frac{1}{q}\right)(1 - \delta')n;$$
$$\text{and} \quad \mathop{\mathbf{E}}_{\mathbf{x} \in \mathcal{B}} [\Delta(\mathbf{r}, \mathbf{x})] = \left(1 - \frac{1}{q}\right)(1 - \gamma')n.$$

We then have $\delta' \leqslant \delta < \gamma^2 \leqslant \gamma'^2$, and by Theorem 3.1,

$$|\mathcal{B}_q(\mathbf{r}, e) \cap \mathcal{C}| \leqslant \frac{1 - \delta'}{\gamma'^2 - \delta'} = 1 + \frac{1 - \gamma'^2}{\gamma'^2 - \delta'} \leqslant 1 + \frac{1 - \gamma^2}{\gamma^2 - \delta} = \frac{1 - \delta}{\gamma^2 - \delta}.$$

□

The bound $n\left(1 - \frac{1}{q}\right)\left(1 - \sqrt{1 - \frac{q}{q-1} \cdot \frac{d}{n}}\right)$ is called the (q-ary) Johnson radius, and in every q-ary code of relative distance d/n, every Hamming ball of this radius is guaranteed to have few codewords.

Proof. (of Theorem 3.1) To keep the notation simple, we will assume that the alphabet is $\{0, 1, \ldots, q-1\}$ and that $\mathbf{r} = 0^n$. Let $\beta = \frac{e}{n}$ and $M = |\mathcal{B}|$. Pick distinct codewords $\mathbf{x} = (x_1, \ldots, x_n)$, $\mathbf{y} = (y_1, \ldots, y_n) \in \mathcal{B}$, uniformly at random. We will obtain a lower bound (in terms of e and M) on the expected number of coordinates where $\mathbf{x}$ and $\mathbf{y}$ agree. We know that this expectation is $n - d$. The theorem will follow by comparing these two quantities.

For $i \in [n]$ and $\alpha \in [q]$, let $k_i(\alpha) = |\{\mathbf{x} \in \mathcal{B} : x_i = \alpha\}|$. Note that $\sum_{\alpha \in [q]} k_i(\alpha) = M$. Also, $k_i(0)$ is the number of codewords in $\mathcal{B}$ that agree with $\mathbf{r}$ at location i. Thus, for $i = 1, 2, \ldots, n$,

$$\begin{aligned}
\Pr[x_i = y_i] &= \binom{M}{2}^{-1} \sum_{\alpha \in [q]} \binom{k_i(\alpha)}{2} \\
&= \binom{M}{2}^{-1} \left[\binom{k_i(0)}{2} + \sum_{\alpha=1}^{q-1} \binom{k_i(\alpha)}{2}\right] \\
&\geqslant \binom{M}{2}^{-1} \left[\binom{k_i(0)}{2} + (q-1)\binom{\frac{M - k_i(0)}{q-1}}{2}\right],
\end{aligned}$$

using Jensen's inequality. Then, the expected number of coordinates where $\mathbf{x}$ and $\mathbf{y}$ agree is

$$\begin{aligned}\sum_{i=1}^{n} \Pr[x_i = y_i] &\geqslant \binom{M}{2}^{-1} \sum_{i=1}^{n} \left[\binom{k_i(0)}{2} + (q-1)\binom{\frac{M-k_i(0)}{q-1}}{2}\right] \\ &\geqslant \binom{M}{2}^{-1} n \left[\binom{k}{2} + (q-1)\binom{\frac{M-k}{q-1}}{2}\right], \qquad (3.1)\end{aligned}$$

using Jensen's inequality again, where $k = \frac{1}{n}\sum_i k_i(0) = M\left(\frac{n-e}{n}\right)$. This expectation is exactly $n-d$. Thus,

$$\begin{aligned}\left(\frac{n-d}{n}\right)\binom{M}{2} &\geqslant \binom{k}{2} + (q-1)\binom{\frac{M-k}{q-1}}{2} \\ \text{i.e.,}\quad \left(\frac{n-d}{n}\right) M(M-1) &\geqslant k(k-1) + (M-k)\left(\frac{M-k}{q-1} - 1\right).\end{aligned}$$

Substituting $1 - \frac{e}{n}$ for $\frac{k}{M}$, the above is equivalent to

$$\left(1 - \frac{d}{n}\right)(M-1) \geqslant \left(1 - \frac{e}{n}\right)^2 M + \frac{e^2}{(q-1)n^2} M - 1,$$

which upon rearranging gives

$$M \leqslant \frac{\frac{q}{q-1} \cdot \frac{d}{n}}{\left(1 - \frac{q}{q-1} \cdot \frac{e}{n}\right)^2 - \left(1 - \frac{q}{q-1} \cdot \frac{d}{n}\right)}.$$

□

We can also state the following alphabet independent version of the Johnson bound:

Theorem 3.3. **(Large alphabet)** Let $\mathcal{C}$ be a q-ary code of block length n and distance d. Suppose $\mathbf{r} \in [q]^n$ and $(n-e)^2 > n(n-d)$. Then,

$$|\mathcal{B}(\mathbf{r}, e) \cap \mathcal{C}| \leqslant \frac{nd}{(n-e)^2 - n(n-d)}.$$

In particular, if $(n-e)^2 \geqslant n(n-d) + 1$, then $|\mathcal{B}(\mathbf{r}, e) \cap \mathcal{C}| \leqslant n^2$.

Proof. The denominator of the upper bound on $|\mathcal{B}|$ in Theorem 3.1 equals

$$\frac{q}{q-1}\left(\frac{q}{q-1}\frac{e^2}{n^2}-\frac{2e}{n}+\frac{d}{n}\right) \geqslant \frac{q}{q-1}\left(\left(1-\frac{e}{n}\right)^2-\left(1-\frac{d}{n}\right)\right).$$

Hence it follows that $|\mathcal{B}| \leqslant \frac{nd}{(n-e)^2-n(n-d)}$. □

Theorem 3.3 says that a code of relative distance δ can be list decoded up to a fraction $(1-\sqrt{1-\delta})$ of errors with polynomial size lists. Note that for $\delta \to 1$, the fraction of errors approaches 1, whereas with unique decoding, we can never correct more than a fraction 1/2 of errors.

The reader may wonder whether the Johnson bound is tight, or whether it may be possible to improve it and show that for every code of a certain relative distance, Hamming balls of radius somewhat larger than the Johnson radius still have polynomially many codewords. It turns out that purely as a function of the distance, the Johnson bound is the best possible. That is, there exist codes which have super-polynomially many codewords in a ball of radius slightly bigger than the Johnson radius. For general codes, this is shown by an easy random coding argument which picks, randomly and independently, several codewords of weight slightly greater than the Johnson radius [20, Section 4.3]. For linear codes, the result is harder to prove, and was shown for the binary case in [24].

We remark that, in a certain precise sense, it is known that for *most* codes, the Johnson bound is *not* tight and list decoding radius for polynomial-sized lists far exceeds the Johnson radius. This follows from random coding arguments used in the proofs of Theorems 3.5 and 3.6 below.

3.2 The capacity of list decoding

We now turn to determining the trade-off between the rate of a code and its list decoding radius, i.e., the "capacity" of list decoding. Throughout, q will be the alphabet size and p the fractional error-correction radius. The function $H_q(x)$ denotes the q-ary entropy function, given by

$$H_q(x) = x\log_q(q-1) - x\log_q x - (1-x)\log_q(1-x). \tag{3.2}$$

The significance of the entropy function for us is the fact that asymptotically $q^{H_q(p)n}$ is a very good estimate of the volume $|\mathcal{B}_q(\mathbf{0},pn)|$ of the q-ary Hamming ball of radius pn. Note that $|\mathcal{B}_q(\mathbf{0},pn)| = \sum_{i=1}^{pn}\binom{n}{i}(q-1)^i$. It is known, see for example [75, Chapter 1], that for $0 < p < 1 - 1/q$ and growing n,

$$q^{H_q(p)n-o(n)} \leqslant |\mathcal{B}_q(\mathbf{0},pn)| \leqslant q^{H_q(p)n}. \tag{3.3}$$

We begin with a simple upper bound on the list decoding radius.

Theorem 3.4. Let $q \geqslant 2$ be an integer, $0 < p < 1 - 1/q$, and $\varepsilon > 0$. Then for all large enough n, if C is a q-ary code of block length n and rate $1 - H_q(p) + \varepsilon$, then there is a center $\mathbf{r} \in [q]^n$ such that $|\mathcal{B}_q(\mathbf{r},pn) \cap C| \geqslant q^{\varepsilon n/2}$. In other words, C cannot be list decoded up to a fraction p of errors with polynomial sized lists.

Proof. Pick a random $\mathbf{r} \in [q]^n$ and consider the random variable $Z = |\mathcal{B}_q(\mathbf{r},pn) \cap C|$. Clearly, $\mathbf{E}[Z] = |C|\frac{|\mathcal{B}_q(\mathbf{0},pn)|}{q^n}$ which is at least $q^{\varepsilon n - o(n)}$ using (3.3). Therefore, there must exist a center $\mathbf{r}$ such that $|\mathcal{B}_q(\mathbf{r},pn) \cap C| \geqslant q^{\varepsilon n - o(n)} \geqslant q^{\varepsilon n/2}$. □

It turns out that the above simple bound is in fact tight and a rate approaching $1 - H_q(p)$ can in fact be attained for correcting a fraction p of errors, as shown below.

Theorem 3.5. **[14]** For integers $q, L \geqslant 2$ and every $p \in (0, 1 - 1/q)$, there exists a family of (p,L)-list-decodable q-ary error-correcting codes of rate R satisfying $R \geqslant 1 - H_q(p) - 1/L$.

Proof. The proof follows a standard "random coding" argument. Fix a large enough block length n; for simplicity assume that $e = pn$ is an integer. The idea is to pick (with replacement) a *random* code of block length n consisting of M codewords, where M is a parameter that will be fixed later in the proof. We will show that with high probability the resulting code will be (p,L)-list decodable.

The probability that a fixed set of $(L+1)$ codewords all lie in a fixed Hamming sphere (in the space $[q]^n$) of radius pn equals $(|\mathcal{B}_q(\mathbf{0},pn)|/q^n)^{L+1}$. By (3.3), this probability is at most $q^{-(L+1)(1-H_q(p))n}$. Therefore, by a union bound, the probability that some tuple of $(L+1)$ codewords all lie in some Hamming ball of radius pn is at most

$$\binom{M}{L+1} \cdot q^n \cdot q^{-(L+1)(1-H_q(p))n}.$$

If $M = q^{rn}$ for $r = 1 - H_q(p) - 1/(L+1)$, then the above probability is at most $1/(L+1)! < 1/3$. Also, among the M chosen codewords there are at least $M/2$ distinct codewords with probability at least $1/2$. Hence, there exists a (p,L)-list-decodable with $q^{(1-H_q(p)-1/(L+1))n}/2 \geqslant q^{(1-H_q(p)-1/L)n}$ distinct codewords, or in other words with rate at least $1 - H_q(p) - 1/L$, as claimed. □

Remark 2. Using a variant of the above random coding argument together with "expurgation" the rate lower bound can be improved slightly to $1 - H_q(p)(1 + 1/L)$; see [28, Theorem 5.5] for details.

The above two results imply that, over alphabets of size q, the *optimal rate possible for list decoding to radius p is* $1 - H_q(p)$. The proof of Theorem 3.5 is non-constructive, and the big challenge is to construct an explicit code with rate close to capacity. It turns out one can also approach capacity using a linear code, as the following result, which first appeared implicitly in the work of Zyablov and Pinsker [79], shows. This result is also non-constructive and is proved by picking a random linear code of the stated rate. We skip the proof here; the reader may find the detailed proof in [28, Theorem 5.6]. The key point is that in any L-tuple of nonzero messages, there must be at least $\log_q(L+1)$ linearly independent messages, and these are mapped to completely independent codewords by a random linear code.

Theorem 3.6. For every prime power $q \geqslant 2$, every $p \in (0, 1 - 1/q)$, and every integer $L \geqslant 2$, there exists a family of (p, L)-list-decodable q-ary *linear* error-correcting codes of rate

$$R \geqslant 1 - H_q(p) - \frac{1}{\log_q(L+1)}.$$

Remark 3. In order for the rate to be within ε of the capacity, the list size needed for linear codes as per Theorem 3.6 is exponentially worse than for general codes. This is not known to be inherent, and we suspect that it is not the case. We conjecture a trade-off similar to Theorem 3.5 can also be obtained using linear codes. For the binary case $q = 2$, this was shown in [30] via a subtle use of the semi-random method. Generalizing this claim to larger alphabets has remained open. It is also known that for each fixed constant L, the rate of (p, L)-list-decodable binary codes is strictly less than $1 - H(p)$ [8], and so unbounded list size is needed to achieve the capacity of list decoding. A similar result should hold over all alphabets; for q-ary alphabets with $q > 2$ this has been shown assuming the convexity of a certain function [9]. Without any assumption, it is shown in [9] that for any fixed L, the rate of (p, L)-list decodable q-ary codes becomes zero for p strictly less than $1 - 1/q$.

3.2.1 Implication for large alphabets

We now inspect the behavior of the list decoding capacity $1 - H_q(p)$ as the alphabet size q grows. We have

$$1 - H_q(p) = 1 - p + p\log_q\left(\frac{q}{q-1}\right) - \frac{H(p)}{\log q},$$

where $H(x)$ is the binary entropy function and $\log x$ denotes logarithm to the base 2. In the limit of large q, the list decoding capacity thus approaches $1 - p$. In other words, we can list decode a code of rate R up to a fraction of errors approaching $1 - R$, as mentioned in Chapter 1.

Since $1 - H_q(p) \geqslant 1 - p - \frac{1}{\log q}$, we can get within ε of capacity, i.e., achieve a rate of $1 - p - \varepsilon$ for list decoding up to a fraction p of errors, over an alphabet of size $2^{O(1/\varepsilon)}$. Moreover, a list size of $O(1/\varepsilon)$ suffices for this result. Conversely, if we fix p and let $\varepsilon \to 0$, then an alphabet size of $2^{\Omega(1/\varepsilon)}$ is necessary in order to achieve a rate of $1 - p - \varepsilon$ for list decoding up to a fraction p of errors. We record this discussion below:

Corollary 3.7. Let $0 < R < 1$, and $\varepsilon > 0$ be sufficiently small. Then there exists a family of error-correcting codes of rate R over an alphabet of size $2^{O(1/\varepsilon)}$ that are $(1 - R - \varepsilon, O(1/\varepsilon))$-list-decodable. Conversely, for a fixed p and $\varepsilon \to 0$, if there exists a family of q-ary $(p - \varepsilon)$-list-decodable codes of rate $1 - p$, then $q \geqslant 2^{\Omega(1/\varepsilon)}$ (here the constant in the $\Omega(1/\varepsilon)$ depends on p).

The central goal in this survey will be to achieve the above result constructively, i.e., give explicit codes of rate R and efficient algorithms to list decode them up to a fraction $(1 - R - \varepsilon)$ of errors. We will meet this goal, except we need a list size that is much larger than the existential $O(1/\varepsilon)$ bound above (the alphabet size of our codes will be $2^{O(1/\varepsilon^4)}$, which is not much worse compared to the $2^{\Omega(1/\varepsilon)}$ lower bound).

4

Decoding Reed–Solomon Codes

In this chapter, we will present a list decoding algorithm for Reed–Solomon codes that can decode up to the Johnson radius, namely a fraction $1 - \sqrt{1-\delta}$ of errors, where δ is the relative distance. Since $\delta = 1 - R$ for RS codes of rate R, this enables us to correct a fraction $1 - \sqrt{R}$ of errors with rate R.

4.1 A toy problem: Decoding a mixture of two codewords

We begin with a simple setting, first considered in [5], that will motivate the main approach underlying the list decoding algorithm. Under list decoding, if we get a received word that has two closest codewords, we need to return both those codewords. In this section, we develop an algorithm that, given received words that are formed by "mixing up" two codewords, recovers those codewords. In a sense, this is the simplest form in which a list decoding style question arises.

4.1.1 Scrambling two codewords

Let $C = \mathsf{RS}_{\mathbb{F},S}[n,k]$ be the Reed–Solomon code obtained by evaluating degree k polynomials over a field $\mathbb{F}$ at a set $S = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ of n

distinct field elements. Let us assume $k < n/2$ for this section. Suppose we are given two unknown codewords, corresponding to two unknown polynomials $p_1(X)$ and $p_2(X)$, in a scrambled fashion as follows: For each $i = 1, 2, \ldots, n$, we are given a pair (a_i, b_i) such that either $a_i = p_1(\alpha_i)$ and $b_i = p_2(\alpha_i)$, or $a_i = p_2(\alpha_i)$ and $b_i = p_1(\alpha_i)$. The goal is to recover the polynomials $p_1(X)$ and $p_2(X)$ from this data.

For each i, since we know both $p_1(\alpha_i)$ and $p_2(\alpha_i)$ in some order, we can compute $p_1(\alpha_i) + p_2(\alpha_i) = a_i + b_i$, as well as $p_1(\alpha_i)p_2(\alpha_i) = a_i b_i$. In other words, we know the value of the polynomial $S(X) \overset{\text{def}}{=} p_1(X) + p_2(X)$ and $P(X) = p_1(X)p_2(X)$ at all the α_is. Now the degree of both $S(X)$ and $P(X)$ is at most $2k < n$. Therefore, using polynomial interpolation, we can completely determine both these polynomials from their values at all the α_is.

Now consider the bivariate polynomial $Q(X,Y) = Y^2 - S(X)Y + P(X)$. We just argued that we can compute the polynomial $Q(X,Y)$. But clearly $Q(X,Y) = (Y - p_1(X))(Y - p_2(X))$, and therefore we can find $p_1(X)$ and $p_2(X)$ by factorizing the bivariate polynomial $Q(X,Y)$ into its irreducible factors. In fact, we only need to find the roots of $Q(X,Y)$ (treated as a polynomial in Y with coefficients from the ring $\mathbb{F}[X]$). This task can be accomplished in polynomial time (details appear in Section 4.5).

4.1.2 Mixing two codewords

We now consider a related question, where for each codeword position i, we are given either y_i which equals either $p_1(\alpha_i)$ or $p_2(\alpha_i)$ (and we do not know which value we are given). Given such a mixture of two codewords, our goal is to identify $p_1(X)$ and $p_2(X)$. Now clearly, we may only be given the value $p_1(\alpha_i)$ for all i, and in this case we have no information about $p_2(X)$. Under the assumption $k < n/6$, the algorithm below will identify both the polynomials if they are both well represented in the following sense: for both $j = 1, 2$, we have $p_j(\alpha_i) = y_i$ for at least $1/3$ of the α_is.

The following simple observation sets the stage for the algorithm: For each $i \in [n]$, $(y_i - p_1(\alpha_i))(y_i - p_2(\alpha_i)) = 0$. In other words, the polynomial $Q(X,Y) = (Y - p_1(X))(Y - p_2(X))$ satisfies $Q(\alpha_i, y_i) = 0$ for every $i \in [n]$.

Unlike the previous section, we are now no longer be able to construct the polynomial $Q(X,Y)$ directly by computing the sum and product of $p_1(X)$ and $p_2(X)$. Instead, we will directly *interpolate* a polynomial $\tilde{Q}(X,Y)$ that satisfies $\tilde{Q}(\alpha_i, y_i) = 0$ for all $i \in [n]$, with the hope that will help us to find $Q(X,Y)$. Of course, doing this without any restriction on the degree of $\tilde{Q}$ is useless (for instance, we could then just take $\tilde{Q}(X,Y) = \prod_{i=1}^{n}(X - \alpha_i)$, which reveals no information about $p_1(X)$ or $p_2(X)$).

From the existence of $Q(X,Y) = (Y - p_1(X))(Y - p_2(X))$, which vanishes at all the pairs (α_i, y_i), we can require that $\tilde{Q}(X,Y)$ be of the form

$$\tilde{Q}(X,Y) = Y^2 + Y\left(\sum_{j=0}^{k} q_{1j}X^j\right) + \sum_{j=0}^{2k} q_{2j}X^j. \tag{4.1}$$

The conditions $\tilde{Q}(\alpha_i, y_i) = 0$ pose a system of n linear equations in the variables q_{1j}, q_{2j}. From the existence of $Q(X,Y)$, we know this system has a solution. Therefore, we can interpolate a $\tilde{Q}(X,Y)$ of the form (4.1) by solving this linear system. The following simple lemma shows the utility of any such polynomial $\tilde{Q}$ that we find.

Lemma 4.1. If $p(X)$ is a polynomial of degree at most $k < n/6$ such that $p(\alpha_i) = y_i$ for at least $n/3$ values of $i \in [n]$, then $\tilde{Q}(X, p(X)) \equiv 0$, or in other words $Y - p(X)$ is a factor of $\tilde{Q}(X,Y)$.

Proof. Define the univariate polynomial $R(X) \overset{\text{def}}{=} Q(X, p(X))$. Let $S = \{i \mid p(\alpha_i) = y_i;\ \ 1 \leqslant i \leqslant n\}$. For each $i \in S$, we have $R(\alpha_i) = \tilde{Q}(\alpha_i, p(\alpha_i)) = \tilde{Q}(\alpha_i, y_i) = 0$. Since the α_is are distinct, $R(X)$ has at least $|S| \geqslant n/3$ roots. On the other hand, the degree of $R(X)$ is at most $2k < n/3$. The polynomial $R(X)$ has more roots than its degree, which implies that it must be the zero polynomial. □

Corollary 4.2. If each received symbol y_i equals either $p_1(\alpha_i)$ or $p_2(\alpha_i)$, and moreover $y_i = p_j(\alpha_i)$ for at least $n/3$ values of $i \in [n]$ for each $j = 1, 2$, then the interpolated polynomial $\tilde{Q}(X,Y)$ equals $(Y - p_1(X))(Y - p_2(X))$.

Therefore, once we find $\tilde{Q}(X,Y)$, we can recover the solutions $p_1(X)$ and $p_2(X)$ by a root-finding step.

4.2 Reed–Solomon list decoding

We now turn to the problem of list decoding Reed–Solomon codes. In this section, we will describe an algorithm due to Sudan [69] that decodes close to a fraction 1 of errors for low rates. Let C_{RS} be an $[n, k+1, n-k]_q$ Reed–Solomon code over a field $\mathbb{F}$ of size $q \geqslant n$, with a degree k polynomial $p(X) \in \mathbb{F}[X]$ being encoded as $(p(\alpha_1), p(\alpha_2), \ldots, p(\alpha_n))$. The problem of decoding such an RS code up to e errors reduces to the following *polynomial reconstruction* problem with agreement parameter $t = n - e$:

Problem 4.1. (Polynomial reconstruction)
INPUT: Integers k, t, and n distinct pairs $\{(\alpha_i, y_i)\}_{i=1}^n$ where $\alpha_i, y_i \in \mathbb{F}$.
OUTPUT: A list of all polynomials $p(X) \in \mathbb{F}[X]$ of degree at most k which satisfy $p(\alpha_i) = y_i$ for at least t values of $i \in [n]$.

Note that the polynomial reconstruction problem is more general than RS list decoding since the α_is are not required to be distinct.

Definition 4.1. ((1,*k*)-weighted degree) For a polynomial $Q(X,Y) \in \mathbb{F}[X,Y]$, its $(1,k)$-weighted degree is defined to be the maximum value of $\ell + kj$ taken over all monomials $X^\ell Y^j$ that occur with a nonzero coefficient in $Q(X,Y)$.

The following fact generalizes Lemma 4.1 and has an identical proof:

Lemma 4.3. Suppose $Q(X,Y)$ is a nonzero polynomial with $(1,k)$-weighted degree at most D satisfying $Q(\alpha_i, y_i) = 0$ for every $i \in [n]$. Let $p(X)$ be a polynomial of degree at most k such that $p(\alpha_i) = y_i$ for at least $t > D$ values of $i \in [n]$. Then $Q(X, p(X)) \equiv 0$, or in other words $Y - p(X)$ is a factor of $Q(X,Y)$.

In view of the above lemma, if we could interpolate a nonzero polynomial $Q(X,Y)$ of $(1,k)$-weighted degree less than t satisfying

$Q(\alpha_i, y_i) = 0$ for all $i \in [n]$, then we can find all polynomials that have agreement at least t with the points amongst the factors of $Q(X,Y)$. In Section 4.1.2, we knew that such a polynomial $Q(X,Y)$ of $(1,k)$-weighted degree $2k$ existed since $(Y - p_1(X))(Y - p_2(X))$ was an explicit example of such a polynomial. Note that once we are assured of the existence of such a polynomial, we can find one by solving a linear system. But now that y_is are arbitrary, how do we argue about the existence of a Q-polynomial of certain $(1,k)$-weighted degree? It turns out a simple counting argument is all that it takes to guarantee this.

Lemma 4.4. Given an arbitrary set of n pairs $\{(\alpha_i, y_i)\}_{i=1}^n$ from $\mathbb{F} \times \mathbb{F}$, there exists a nonzero polynomial $Q(X,Y)$ of $(1,k)$-weighted degree at most D satisfying $Q(\alpha_i, y_i) = 0$ for all $i \in [n]$ provided $\binom{D+2}{2} > kn$. Moreover, we can find such a $Q(X,Y)$ in polynomial time by solving a linear system over $\mathbb{F}$.

Proof. A polynomial $Q(X,Y)$ with $(1,k)$-weighted degree at most D can be expressed as

$$Q(X,Y) = \sum_{j=0}^{\lfloor D/k \rfloor} \sum_{\ell=0}^{D-jk} q_{\ell j} X^{\ell} Y^{j}. \tag{4.2}$$

The conditions $Q(\alpha_i, y_i) = 0$ for all $i \in [n]$ give a system of n homogeneous linear equations in the unknowns $q_{\ell j}$. A nonzero solution to this system is guaranteed to exist if the number of unknowns, say U, exceeds the number n of equations (and if a nonzero solution exists, we can clearly find one in polynomial time by solving the above linear system). We turn to estimating U.

$$\begin{aligned} U &= \sum_{j=0}^{\lfloor D/k \rfloor} \sum_{\ell=0}^{D-jk} 1 = \sum_{j=0}^{\lfloor D/k \rfloor} (D - jk + 1) \\ &= (D+1)\left(\left\lfloor \frac{D}{k} \right\rfloor + 1\right) - \frac{k}{2} \left\lfloor \frac{D}{k} \right\rfloor \left(\left\lfloor \frac{D}{k} \right\rfloor + 1\right) \\ &\geqslant \frac{(D+1)(D+2)}{2k}. \end{aligned}$$

Thus if $\binom{D+2}{2} > kn$, we have $U > n$, and a nonzero solution exists to the linear system. □

The above discussion motivates the following algorithm for polynomial reconstruction, for an agreement parameter $t > \sqrt{2kn}$:

Step 1: (Interpolation) Let $D = [\sqrt{2kn}]$. Find a nonzero polynomial $Q(X,Y)$ of $(1,k)$-weighted degree at most D satisfying $Q(\alpha_i, y_i) = 0$ for $i = 1,2,\ldots,n$. (Lemma 4.4 guarantees the success of this step.)

Step 2: (Root finding/Factorization) Find all degree k polynomials $p(X)$ such that $Q(X,p(X)) \equiv 0$. For each such polynomial, check if $p(\alpha_i) = y_i$ for at least t values of $i \in [n]$, and if so, include $p(X)$ in the output list. (Lemma 4.3 guarantees that this step will find all relevant polynomials with agreement $t > D = [\sqrt{2kn}]$.)

The following records the performance of this algorithm. The claim about the output list size follows since the number of factors $Y - p(X)$ of $Q(X,Y)$ is at most the degree of $Q(X,Y)$ in Y, which is at most $\left[\frac{D}{k}\right]$.

Theorem 4.5. The above algorithm solves the polynomial reconstruction problem in polynomial time if the agreement parameter t satisfies $t > \left[\sqrt{2kn}\right]$. The size of the list output by the algorithm never exceeds $\sqrt{2n/k}$.

The above implies that a Reed–Solomon code of rate R can be list decoded to a fraction $1 - \sqrt{2R}$ of errors using lists of size $O(\sqrt{1/R})$. In particular, for low-rate codes, we can efficiently correct close to a fraction 100% of errors! This qualitative feature is extremely useful in many areas of complexity theory, such as constructions of hardcore predicates from one-way functions, randomness extractors and pseudorandom generators, hardness amplification, transformations from worst-case to average-case hardness, etc. – we point to the surveys [29,70,74] and [28, Chapter 12] for further information.

Remark 4. (Unique decoding) By interpolating a nonzero polynomial of the form $Q(X,Y) = A(X)Y + B(X)$, where the degree of $A(X), B(X)$ are at most $\lfloor \frac{n-k-1}{2} \rfloor$ and $\lfloor \frac{n+k-1}{2} \rfloor$, respectively, we can recover the unique polynomial $f(X)$ of degree at most k, if one exists, that is a solution to the polynomial reconstruction problem for agreement parameter $t > \frac{n+k}{2}$. In fact, in this case the solution polynomial is just $-B(X)/A(X)$. The correctness follows from the following two facts:

(i) Let $e(X)$ to be the error-locator polynomial of degree at most $n - t < \frac{n-k}{2}$ that has roots at all the error locations. Then the choice $A(X) = e(X)$ and $B(X) = -f(X)e(X)$ gives a nonzero polynomial $Q(X,Y)$ meeting the degree restrictions and satisfying the interpolation conditions, and

(ii) the $(1,k)$-weighted degree of such a polynomial $Q(X,Y)$ is at most $\lfloor \frac{n-k-1}{2} \rfloor + k = \lfloor \frac{n+k-1}{2} \rfloor < t$.

This gives a polynomial time unique decoding algorithm for RS codes that corrects up to a fraction $(1 - R)/2$ of errors. This form of the algorithm is due to Gemmell and Sudan [16], based on the approach of Welch and Berlekamp [77]. The first polynomial time algorithm for unique decoding RS codes was discovered as early as 1960 by Peterson [63].

4.3 Improved decoding via interpolation with multiplicities

4.3.1 Geometric motivation for approach

We now illustrate of how the above interpolation based decoding works using geometric examples. This will also motivate the idea of using multiplicities in the interpolation stage to improve the performance of the decoding algorithm.

To present the examples, we work over the field $\mathbb{R}$ of real numbers. The collection of pairs $\{(\alpha_i, y_i) : 1 \leqslant i \leqslant n\}$ thus consists of n points in the plane. We will illustrate how the algorithm finds all polynomials

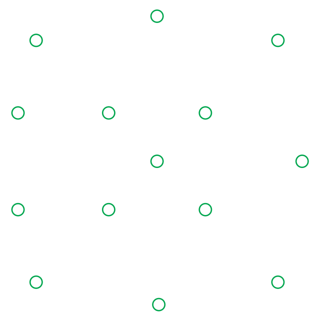

Fig. 4.1 Example 1: The set of 14 input points. We assume that the center-most point is the origin and assume a suitable scaling of the other points.

of degree $k = 1$, or in other words, lines, that pass through at least a certain number t of the n points.

Example 1: For the first example, we take $n = 14$ and $t = 5$. The 14 points on the plane are as in Figure 4.1.

We want to find all lines that pass through at least 5 of the above 14 points. Since $k = 1$, the $(1,k)$-weighted degree of a bivariate polynomial is simply its total degree. The first step of the algorithm must fit a nonzero polynomial $Q(X,Y)$ such that $Q(\alpha_i, y_i) = 0$ for all 14 points. By Lemma 4.4, we can find such a polynomial of total degree 4.

One choice of a degree 4 polynomial that passes through the above 14 points is $Q(X,Y) = Y^4 - X^4 - Y^2 + X^2$. To see this pictorially, let us plot the curve of all points on the plane where Q has zeroes. This gives Figure 4.2 below.

Note that the two relevant lines that pass through at least 5 points emerge in the picture. Algebraically, this corresponds to the fact that $Q(X,Y)$ factors as $Q(X,Y) = (X^2 + Y^2 - 1)(Y + X)(Y - X)$, and the last two factors correspond to the two lines that are the solutions. The fact that the above works correctly, i.e., the fact that the relevant lines must be factors of *any* degree 4 fit through the 14 points, is a consequence of Lemma 4.3 applied to this example (with the choice $D = 4$ and $t = 5$). Geometrically, this corresponds to the fact if a line intersects a degree 4 curve in more than 4 points then the line must in fact be a factor of the curve.

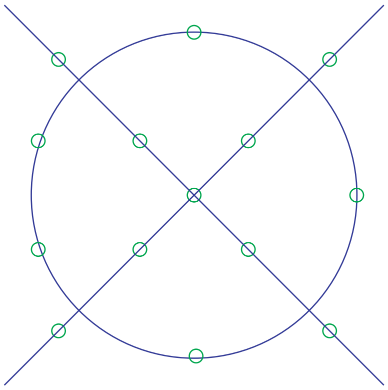

Fig. 4.2 A degree 4 fit through the 14 points. The curve is given by the equation: $Y^4 - X^4 - Y^2 + X^2 = 0$.

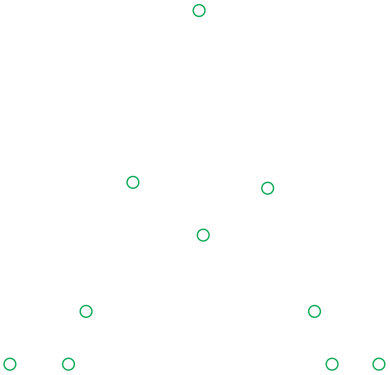

Fig. 4.3 Example 2: The set of 10 input points.

Example 2: For the second example, consider the 10 points in the plane as in Figure 4.3. We want to find all lines that pass through at least 4 of the above 10 points. If we only wanted lines with agreement at least 5, the earlier method of fitting a degree 4 curve is guaranteed to work. Figure 4.4 shows the set $\mathcal{L}$ of all the lines that pass through at least 4 of the given points. Note that there are five lines that must be output. Therefore, if we hope to find all these as factors of some curve, that curve must have degree at least 5. But then Lemma 4.3 does not apply since the agreement parameter $t = 4$ is less than 5, the degree of $Q(X,Y)$.

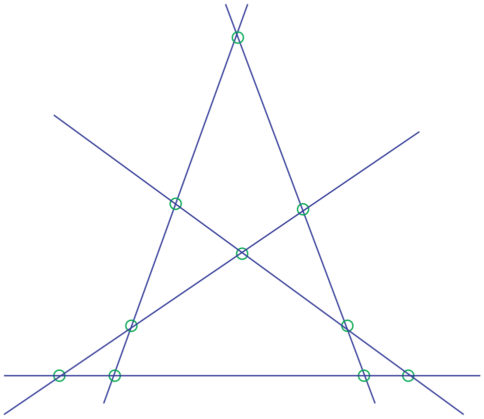

Fig. 4.4 The five lines that pass throught at least 4 of the 10 points.

The example illustrates an important phenomenon which gives the cue for an improved approach. Each of the 10 points has two lines in $\mathcal{L}$ that pass through it. In turn, this implies that if we hope to find all these lines as factors of some curve, that curve *must pass through each point at least twice*! Now we can not expect a generic curve that is interpolated through a set of points to pass through each of them more than once. This suggests that we should make passing through each point multiple times an *explicit requirement* on the interpolated polynomial. Of course, this stronger property cannot be enforced for free and would require an increase in the degree of the interpolated polynomial. But luckily it turns out that one can pass through each point twice with less than a two-fold increase in degree (a factor of roughly $\sqrt{3}$ suffices), and the trade-off between multiplicities guaranteed vs. degree increase is a favorable one. This motivates the improved decoding algorithm presented in the next section.

4.3.2 Algorithm for decoding up to the Johnson radius

We now generalize Sudan's decoding algorithm by allowing for multiplicities at the interpolation points. This generalization is due to Guruswami and Sudan [40].

Definition 4.2. [Multiplicity of zeroes] A polynomial $Q(X,Y)$ is said to have a zero of multiplicity $r \geqslant 1$ at a point $(\alpha,\beta) \in \mathbb{F}^2$ if $Q(X+\alpha, Y+\beta)$ has no monomial of degree less than r with a nonzero coefficient. (The degree of the monomial X^iY^j equals $i+j$.)

The following lemma is the generalization of Lemma 4.3 that takes multiplicities into account.

Lemma 4.6. Let $Q(X,Y)$ be a nonzero polynomial of $(1,k)$-weighted degree at most D that has a zero of multiplicity r at (α_i, y_i) for every $i \in [n]$. Let $p(X)$ be a polynomial of degree at most k such that $p(\alpha_i) = y_i$ for at least $t > D/r$ values of $i \in [n]$. Then, $Q(X,p(X)) \equiv 0$, or in other words $Y - p(X)$ is a factor of $Q(X,Y)$.

Proof. For Q,p as in the statement of the lemma, define $R(X) = Q(X,p(X))$. The degree of $R(X)$ is at most D, the $(1,k)$-weighted degree of $Q(X,Y)$. Let $i \in [n]$ be such that $p(\alpha_i) = y_i$. Define the polynomial $Q^{(i)}(X,Y) \stackrel{\text{def}}{=} Q(X+\alpha_i, Y+y_i)$. Now

$$\begin{aligned} R(X) = Q(X,p(X)) &= Q^{(i)}(X-\alpha_i, p(X)-y_i) \\ &= Q^{(i)}(X-\alpha_i, p(X)-p(\alpha_i)). \end{aligned} \tag{4.3}$$

Since $Q(X,Y)$ has a zero of multiplicity r at (α_i, y_i), $Q^{(i)}(X,Y)$ has no monomials of total degree less than r. Now, $X-\alpha_i$ clearly divides $p(X)-p(\alpha_i)$. Therefore, every term in $Q^{(i)}(X-\alpha_i, p(X)-p(\alpha_i))$ is divisible by $(X-\alpha_i)^r$. It follows from (4.3) that $R(X)$ is divisible by $(X-\alpha_i)^r$.

Since there are at least t values of $i \in [n]$ for which $p(\alpha_i) = y_i$, we get that $R(X)$ is divisible by a polynomial of degree at least rt. If $rt > D$, this implies that $R(X) \equiv 0$. □

We now state the analog of the interpolation lemma when we want desired multiplicities at the input pairs (α_i, y_i). Note that setting $r = 1$, we recover exactly Lemma 4.4.

Lemma 4.7. Given an arbitrary set of n pairs $\{(\alpha_i, y_i)\}_{i=1}^{n}$ from $\mathbb{F} \times \mathbb{F}$ and an integer parameter $r \geqslant 1$, there exists a nonzero polynomial $Q(X,Y)$ of $(1,k)$-weighted degree at most D such that $Q(X,Y)$ has a zero of multiplicity r at (α_i, y_i) for all $i \in [n]$, provided $\binom{D+2}{2} > kn\binom{r+1}{2}$. Moreover, we can find such a $Q(X,Y)$ in time polynomial in n, r by solving a linear system over $\mathbb{F}$.

Proof. Fix an $i \in [n]$. The coefficient of a particular monomial $X^{j_1}Y^{j_2}$ of $Q^{(i)}(X,Y) \overset{\text{def}}{=} Q(X+\alpha_i, Y+y_i)$ can clearly be expressed as a linear combination of the coefficients $q_{\ell j}$ of $Q(X,Y)$ (where $Q(X,Y)$ is expressed as in (4.2)). Thus the condition that $Q(X,Y)$ has a zero of multiplicity r at (α_i, y_i) can be expressed as a system of $\binom{r+1}{2}$ homogeneous linear equations in the unknowns $q_{\ell j}$, one equation for each pair (j_1, j_2) of nonnegative integers with $j_1 + j_2 < r$. In all, for all n pairs (α_i, y_i), we get $n\binom{r+1}{2}$ homogeneous linear equations. The rest of the argument follows the proof of Lemma 4.4 – the only change is that $n\binom{r+1}{2}$ replaces n for the number of equations. □

The above discussion motivates the following algorithm using interpolation with multiplicities for polynomial reconstruction (the parameter $r \geqslant 1$ equals the number of multiplicities):

Step 1: (Interpolation) Let $D = \lfloor \sqrt{knr(r+1)} \rfloor$. Find a nonzero polynomial $Q(X,Y)$ of $(1,k)$-weighted degree at most D such that $Q(X,Y)$ has a zero of multiplicity r at (α_i, y_i) for each $i = 1, 2, \ldots, n$. (Lemma 4.7 guarantees the success of this step.)

Step 2: (Root finding/Factorization) Find all degree k polynomials $p(X)$ such that $Q(X, p(X)) \equiv 0$. For each such polynomial, check if $p(\alpha_i) = y_i$ for at least t values of $i \in [n]$, and if so, include $p(X)$ in the output list. (Lemma 4.6 guarantees that this step will find all relevant polynomials with agreement $t > D/r$.)

The following records the performance of this algorithm. Again, the size of the output list never exceeds the degree of $Q(X,Y)$ in Y, which is at most $\lfloor D/k \rfloor$.

Theorem 4.8. The above algorithm, with multiplicity parameter $r \geqslant 1$, solves the polynomial reconstruction problem in polynomial time if the agreement parameter t satisfies $t > \big[\sqrt{kn(1+\frac{1}{r})}\big]$.[1] The size of the list output by the algorithm never exceeds $\sqrt{nr(r+1)/k}$.

Corollary 4.9. A Reed–Solomon code of rate R can be list decoded in polynomial time up to a fraction $1 - \sqrt{(1+\varepsilon)R}$ of errors using lists of size $O(\varepsilon^{-1}/\sqrt{R})$.

By letting the multiplicity r grow with n, we can decode as long as the agreement parameter satisfies $t > \sqrt{kn}$. Indeed, if $t^2 > kn$, picking

$$r = 1 + \left[\frac{kn - t}{t^2 - kn}\right],$$

and $D = rt - 1$, both the conditions $t > \frac{D}{r}$ and $\binom{D+2}{2} > n\binom{r+1}{2}$ are satisfied, and thus the decoding algorithm successfully finds all polynomials with agreement at least t. The number of such polynomials is at most $D/k \leqslant nt \leqslant n^2$.

Theorem 4.10. [40] The polynomial reconstruction problem with n input pairs, degree k, and agreement parameter t can be solved in polynomial time whenever $t > \sqrt{kn}$. Further, at most n^2 polynomials will ever be output by the algorithm.

We conclude that an RS code of rate R can be list decoded up to a fraction $1 - \sqrt{R}$ of errors. This equals the Johnson radius $1 - \sqrt{1-\delta}$ of the code, since the relative distance of a RS code of rate R equals $1 - R$. This is the main result of this chapter. Note that for every rate R, $0 < R < 1$, the decoding radius $1 - \sqrt{R}$ exceeds the best decoding radius $(1 - R)/2$ that we can hope for with unique decoding.

[1] Let $x = \sqrt{knr(r+1)}$. The condition $t > D/r$ with $D = [x]$ is implied by $t > [x/r]$, since $\lfloor x/r \rfloor \leqslant [x]/r < [x/r] + 1$.

Remark 5. [Role of multiplicities] Using multiplicities in the interpolation led to the improvement of the decoding radius to match the Johnson radius $1 - \sqrt{R}$ and gave an improvement over unique decoding for every rate R. We want to stress that the idea of using multiplicities plays a crucial role in the final result achieving capacity. The improvement it gives over a version that uses only simple zeroes is substantial for the capacity-approaching codes, and in fact it seems crucial to get any improvement over unique decoding (let alone achieve capacity) for rates $R > 1/2$. Also, multiplicities can be used to naturally encode the relative importance of different symbol positions, and this plays a crucial role in soft-decision decoding which is an important problem in practice, see Section 4.4.2 below.

4.4 Extensions: List recovering and soft decoding

4.4.1 List recovering Reed–Solomon codes

In the polynomial reconstruction problem with input pairs (α_i, y_i), the field elements α_i need not be distinct. Therefore, Theorem 4.10 in fact gives a list recovering algorithm for Reed–Solomon codes. (Recall Definition 2.2, where we discussed the notion of list recovering of codes.) Specifically, given input lists of size at most ℓ for each position of an RS code of block length n and dimension $k+1$, the algorithm can find all codewords with agreement on more than $\sqrt{kn\ell}$ positions. In other words, for any integer $\ell \geqslant 1$, an RS code of rate R and block length n can be $(p, \ell, O(n^2\ell^2))$-list-recovered in polynomial time when $p < 1 - \sqrt{R\ell}$. Note that the algorithm can do list recovery even in the noise-free $(p = 0)$ case, only when $R < 1/\ell$. In fact, as shown in [38], there are inherent combinatorial reasons why $1/\ell$ is a limit on the rate for list recovering certain RS codes, so this is not a shortcoming of just the above algorithm.

Later on, for our capacity-approaching codes, we will not have this strong limitation, and the rate R for list recovering can be a constant independent of ℓ (and in fact can approach 1 as the error fraction $p \to 0$). This strong list recovering property will be crucially used in concatenation schemes that enable the reduction of the alphabet size to a constant.

4.4.2 Soft-decision decoding

List recovering dealt with the case when for each position we had a set of more than one candidate symbols. More generally, we could be given a weight for each of the candidates, with the goal being to find all codewords with good weighted agreement, summed over all positions. The weight for position i and symbol γ would presumably be a measure of the confidence of symbol γ being the i-th symbol of the actual codeword that was transmitted. Making use of such weights in the decoding is called "soft-decision" decoding (the weights constitute 'soft' information). Note that list recovering is just a special case when for each position the weights for some symbols equal 1 and the rest equal 0. Soft-decision decoding is important in practice as the field elements corresponding to each position are obtained by some sort of "demodulation" of real-valued signals, and soft-decision decoding can retain more of the information from this process compared with hard decoding which loses a lot of information by quantizing the signal to a single symbol. It is also useful in decoding concatenated codes, where the inner decoder can provide weights along with the choices it outputs, which can then be used by a soft-decision decoder for the outer code.

As mentioned in [40], the multiplicity based interpolation lends itself naturally to a soft-decision version, since the multiplicity required at a point can encode the importance of that point. Given weights $w_{i,\gamma}$ for positions $i \in [n]$ and field elements $\alpha \in \mathbb{F}$, we set the multiplicity of the point (α_i, γ) to be proportional to $w_{i,\gamma}$. This leads to the following claim, which is explicit for instance in [28, Chapter 6]:

Theorem 4.11. (Soft-decision decoding of RS codes) Consider a Reed–Solomon code of block length n and dimension $k+1$ over a field $\mathbb{F}$. Let $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$ be the evaluation points used for the encoding. Let $\varepsilon > 0$ be an arbitrary constant. For each $i \in [n]$ and $\gamma \in \mathbb{F}$, let $w_{i,\gamma}$ be a non-negative rational number. Then, there exists a deterministic algorithm with runtime $\text{poly}(n, |\mathbb{F}|, 1/\varepsilon)$ that, when given as input the weights $w_{i,\gamma}$ for $i \in [n]$ and $\gamma \in \mathbb{F}$, finds a list of *all* polynomials $p(X) \in \mathbb{F}[X]$ of degree at most k that satisfy

$$\sum_{i=1}^{n} w_{i,p(\alpha_i)} \geqslant \sqrt{k \sum_{i=1}^{n} \sum_{\gamma \in \mathbb{F}} w_{i,\gamma}^2} + \varepsilon \max_{i,\gamma} w_{i,\gamma}. \tag{4.4}$$

Koetter and Vardy [51] developed a "front end" that chooses weights that are optimal in a certain sense as inputs to the above algorithm, based on the channel observations and the channel transition probability matrix. This has led to a soft-decision decoding algorithm for RS codes that has led to substantial improvements in practice.

4.5 Root finding for bivariate polynomials

We conclude this chapter by briefly describing how to efficiently (in time polynomial in k,q) solve the bivariate root finding problem that we encountered in the RS list decoding algorithm:

> Given a bivariate polynomial $Q(X,Y) \in \mathbb{F}_q[X,Y]$ and an integer k, find a list of all polynomials $f(X) \in \mathbb{F}_q[X]$ of degree at most k for which $Y - f(X)$ divides $Q(X,Y)$.

4.5.1 A simple randomized algorithm

We discuss a simple randomized method, described in [5, Section 4.2], that reduces the above problem to univariate polynomial factorization.

It suffices to give an algorithm that either finds a polynomial $f(X)$ of degree at most k such that $Y - f(X)$ is a factor $Q(X,Y)$, or concludes that none exists. We can then run this algorithm, divide $Q(X,Y)$ by $Y - f(X)$ if such a factor is found, and recurse on the quotient.

Pick a monic polynomial $r(X) \in \mathbb{F}_q[X]$ of degree $k+1$ by picking its non-leading coefficients uniformly at random from $\mathbb{F}_q$. Compute the polynomial $R(X) \overset{\text{def}}{=} Q(X,r(X))$. If $R(X) = 0$, then divide $Q(X,Y)$ by $Y - r(X)$, and repeat the process with the resulting quotient polynomial. Otherwise, using a univariate polynomial factorization algorithm (such as Berlekamp's randomized factoring algorithm [6]), compute the list $E_1(X), E_2(X), \ldots, E_t(X)$ of all the *monic* irreducible factors of $R(X)$ that have degree $k+1$.

Now suppose $f(X)$ has degree at most k and $Y - f(X)$ is a factor of $Q(X,Y)$. Then, clearly $r(X) - f(X)$ is a *monic* polynomial of degree $k+1$ dividing the univariate polynomial $R(X)$. Hence, if $r(X) - f(X)$ is *irreducible*, then it must equal one of the $E_j(X)$s. In this case, we can simply check if $Y - r(X) + E_j(X)$ divides $Q(X,Y)$ for each $j = 1, 2, \ldots, t$, and if so output the corresponding polynomial.

The algorithm obviously never errs, if $Q(X,Y)$ has no factor of form $Y - f(X)$ with degree$(f) \leqslant k$. Conversely, if there is such a factor $Y - f(X)$, the algorithm success if and only if $r(X) - f(X)$ is irreducible. The crucial point is that this happens with probability at least $\frac{1}{k+1}\left(1 - \frac{1}{q}\right) \geqslant \frac{1}{2(k+1)}$ [55, p. 84]. Upon running $O(k \log k)$ independent trials of this procedure, the algorithm will succeed in finding the factor with probability at least $1 - 1/k$.

4.5.2 Deterministic algorithm

To get a deterministic algorithm, we will reduce the above bivariate root finding problem to a univariate root finding problem over an extension field $\mathbb{F}_{q^e}$ with $e < q$, under the assumption that $k < q - 1$. Recall that in our application, $k+1$ is the dimension of the RS code, and so it is safe to assume this upper bound on k.

For univariate polynomial factorization (and hence also root finding) of a polynomial $f(X) \in \mathbb{F}_{p^t}[X]$, where p is the characteristic of the field, Berlekamp [6] (see also [76, Exercise 14.40]) gave a deterministic algorithm with running time polynomial in degree(f), t, and p (this was achieved via a deterministic polynomial-time reduction from factoring in $\mathbb{F}_{p^t}[X]$ to root finding in $\mathbb{F}_p[X]$). Combined with our reduction, we get a deterministic algorithm for the above bivariate root finding problem in time polynomial in q, k, as desired.

The reduction is straightforward. We first need to find $h(X) \in \mathbb{F}_q[X]$ of degree $k+1$ or higher that is irreducible (over $\mathbb{F}_q$). We can describe an explicit choice $h(X) = X^{q-1} - \gamma$, where γ is any generator of the cyclic multiplicative group $\mathbb{F}_q^*$ (and can be found by a brute-force search in $\mathbb{F}_q$). The algorithm proceeds as follows. Repeatedly, divide $Q(X,Y)$

by $h(X)$ till we are left with $Q_0(X,Y)$ that is not divisible by $h(X)$. Then, viewing $Q_0(X,Y)$ as a polynomial in Y with coefficients in $\mathbb{F}_q[X]$, reduce the coefficients modulo $h(X)$, to get a nonzero univariate polynomial $T(Y) \in \tilde{\mathbb{F}}[Y]$ where $\tilde{\mathbb{F}}$ is the extension field $\mathbb{F}_q[X]/(h(X))$ of degree $q-1$ over $\mathbb{F}_q$. The desired polynomials $f(X)$ now all occur amongst the roots of $T(Y)$ that lie in $\tilde{\mathbb{F}}$, all of which can be found using Berlekamp's algorithm in time polynomial in degree(T) and q.

We will make even more crucial use of polynomials over the extension field $\mathbb{F}_q[X]/(h(X))$ in Chapter 6.

4.6 Related results on algebraic list decoding

The principle behind the Reed–Solomon list decoding algorithm is in fact quite general, and can be used to decode other algebraic codes. RS codes are an instance (possibly the most important one) of a class of codes called redundant residue codes. In these codes, the message is encoded by a collection of its residues modulo certain divisors. For example, in RS codes, the message polynomial $f(X)$ is encoded by its residues modulo $M_i(X)$, where $M_i(X) = X - \alpha_i$ for $i = 0,1,\ldots,n-1$. If the degree of $f(X)$ is k, the first $k+1$ residues suffice to uniquely determine the message. The remaining are redundant residues included in the codeword for protection against errors. The natural parallel of RS codes in the number-theoretic world are the Chinese Remainder codes. If $p_1 < p_2 < \cdots < p_n$ are distinct primes, and $k < n$, one can encode an integer m, $0 \leqslant m < \prod_{i=1}^{k} p_i$, using its residues modulo the p_is as $(m \mod p_1, m \mod p_2, \ldots, m \mod p_n)$. Once again, any k residues suffice to identify m and the rest are redundant residues. The algebraic ideas underlying the (soft-decision) list decoding algorithm for Reed–Solomon codes can be used to list decode Chinese Remainder codes (up to its respective Johnson bound). For details on this, see [39] (and the earlier works [10, 18]).

Reed–Solomon codes are a specific instance of a general family of codes called algebraic-geometric (AG) codes. Underlying an AG code is an algebraic curve, a linear space $\mathcal{L}$ of functions (from the so-called "function field" associated with the curve) that correspond to the messages, and a set $\mathcal{S}$ of rational points on the curve where each function in $\mathcal{L}$ has no poles. A message is encoded by evaluating the associated function at each rational point in $\mathcal{S}$. Reed–Solomon codes correspond to the special case when the curve is the affine line, and the functions used for encoding correspond to rational functions that have at most a certain number of poles at the point at infinity and no poles elsewhere (this class corresponds precisely to low-degree polynomials).

The affine line over $\mathbb{F}_q$ has $q + 1$ rational points (including the point at infinity). In turn this means that the alphabet size of RS codes must grow with the block length. The benefit of AG codes is that in general, algebraic curves over $\mathbb{F}_q$ can have many more rational points, in fact as many points as one needs. Thus one can define an infinite family of codes of increasing block lengths over a fixed alphabet $\mathbb{F}_q$. The quality of the codes depends on the quality of the curve – as a curve contains more and more rational points, it must necessarily get more "twisted" and complicated, a phenomenon which is quantified by the *genus* of the curve. The best codes in this framework are obtained using curves with the best trade-off between number of rational points and the genus. The distance property of AG codes follows from the fact that a nonzero function cannot have more zeroes than poles (the fact that a nonzero polynomial cannot have more roots than its degree is a special case of this).

The RS list decoding algorithm described in this chapter can be generalized to work for any algebraic-geometric code [40]. The complexity of the algorithm is polynomial assuming availability of a polynomial amount of pre-processed information about the code [43]. For a family of AG codes that achieve the best rate vs. distance trade-off, it was recently shown how to compute the required pre-processed information in polynomial time [36].

Even more generally, there is an abstract algebraic view of the decoding algorithm in the language of rings and ideals. The algorithms for RS codes, Chinese Remainder codes, and AG codes become just specific instantiations of this general algorithmic scheme for specific choices of the underlying ring and ideals. Details of the general algorithm for any "ideal-based" code that satisfies certain abstract axioms can be found in [28, Chapter 7] and [71].

5

Graph-Based List-Decodable Codes

In this chapter, we briefly survey some non-algebraic, graph-theoretic approaches to constructing list-decodable codes and decoding them. Besides providing an interesting alternate approach for list decoding, these results also give linear-time encodable and list-decodable codes. The rate vs. error-correction radius trade-off, however, is not optimized and is significantly worse than what can be achieved with algebraic codes such as RS codes. The results of this chapter are not needed for Part II of the survey, so we will be content with stating the main results and definitions, and giving very high level descriptions of the central ideas. We will also provide pointers to the literature where further details on the proofs can be found.

5.1 Reducing list decoding to list recovering

The notion of list recovering (recall Definition 2.2) plays a crucial role in the graph-based constructions. The first step is to reduce list decoding to the problem of list recovering with a much smaller noise fraction (but with large input list sizes). In this section, we will present the details of such a reduction. The reduction uses the notion of a

Ramanujan graph:

Definition 5.1. An undirected d-regular graph is said to be a Ramanujan expander if the second largest eigenvalue of its adjacency matrix in absolute value is at most $2\sqrt{d}$.

We will also use the following operation to combine a bipartite graph with a code to produce a code over a larger alphabet. This operation was first used in [1] to amplify the distance of codes, and has been put to good algorithmic use in many recent works beginning with [31].

Definition 5.2. Given a code $C \subseteq \Sigma^n$ and a bipartite graph $G = (A, B, E)$ with $A = \{1, 2, \ldots, n\}$, $B = \{1, 2, \ldots, m\}$, and with right degree d, the code $G(C) \subseteq (\Sigma^d)^m$ is defined as follows. For any $x \in \Sigma^n$, first define $G(x)$ to be a vector $y \in (\Sigma^d)^m$ created in the following way: For $j \in B$ and $k = 1, \ldots d$, let $\Gamma_k(j) \in A$ be the k-th neighbor of j (as per some arbitrary ordering of the neighbors of each node). The j-th symbol y_j of y is defined as $\langle x_{\Gamma_1(j)}, \ldots, x_{\Gamma_d(j)} \rangle$. In other words, we "send" a copy of each symbol x_i along all edges going out of the vertex $i \in A$, and the symbol y_j is obtained by concatenating all symbols "received" by $j \in B$. The code $G(C)$ is now obtained by taking all vectors $G(c)$ for $c \in C$.

The reduction is described formally below. Note that the error fraction is reduced from $(1 - 1/\ell)$ (which is close to 1 for large ℓ) to γ for an arbitrarily small constant $\gamma > 0$ (at the expense of a larger alphabet size and lower rate).

Lemma 5.1. **[33]** Let $\varepsilon, \gamma \in (0, 1)$ be arbitrary constants. Let C be a code of block length n and rate r over alphabet Σ. Also suppose that C is (γ, ℓ, L)-list-recoverable in time $T(n)$. Further, assume that there exists a d-regular Ramanujan expander H on n vertices for $d \geqslant \frac{4}{\gamma\varepsilon^2}$. Then there exists a code C' of block length n and rate r/d over alphabet Σ^d which is explicitly specified given C, H, and which is $(1 - \frac{1}{\ell} - \varepsilon, L)$-list-decodable in time $O(T(n) + n)$. Furthermore, C' is linear-time encodable if C is.

Proof. Let $G = (A, B, E)$ be an $n \times n$ bipartite graph that is the double cover of H (i.e., A, B are both copies of the vertex set of H, and we connect $u \in A$ with $v \in B$ if (u, v) is an edge of H). The code C' will just be $G(C)$ described in Definition 5.2. The claims about block length, rate, alphabet size, and encoding time follow immediately. Using the fact that H is Ramanujan and $d \geqslant 4/(\gamma\varepsilon^2)$, and the pseudorandom properties of expander graphs (see for instance [4, Chapter 9, Section 2]), one can show the following:

(**) For any $S \subseteq B$, $|S| \geqslant (\frac{1}{\ell} + \varepsilon)|B|$, the fraction of $i \in A$ which have at most a fraction $\frac{1}{\ell}$ of their neighbors inside the set S is at most γ.

The list decoding algorithm for C' proceeds as follows. Given a string $y \in (\Sigma^d)^n$, for each $i \in A$, create a set $S_i \subseteq \Sigma$ consisting of the ℓ most frequent elements in the multiset T_i of elements defined as: $T_i = \{a_k \mid (i, j) \in E;\ y_j = \langle a_1, a_2, \ldots, a_d \rangle;\ \Gamma_k(j) = i\}$.

Let $c' \in C'$ be a codeword such that $c'_j = y_j$ for at least $(\frac{1}{\ell} + \varepsilon)n$ of the positions j. Let $c \in C$ be the codeword for which $c' = G(c)$. By Property (**), we have $c_i \notin S_i$ for at most a γ fraction of the sets S_i. Thus we can find c, and hence c', by running the (γ, ℓ, L)-recovering algorithm for C with the sets S_i as input. The output list size is at most L, and the running time is $T(n)$ plus the time to compute the lists, which is at most $O(nd) = O(n)$ word operations when d is a fixed constant depending only on ε, γ. □

5.2 A toy list recovering problem

In light of Lemma 5.1, if we can construct (γ, ℓ, L)-list-recoverable codes of positive rate for some $\gamma = \gamma_\ell > 0$, then we can also construct $(1 - 2/\ell, L)$-list-decodable codes of positive rate $R_\ell > 0$. Furthermore, if the list recovering algorithm runs in linear time, we will have also have a linear-time list-decoding algorithm for correcting a fraction $(1 - 2/\ell)$ of errors.

In this section, we illustrate the basic approach toward such a result on list recovering by considering a toy problem, namely constructing a family of codes of positive rate that are $(0, 2, L)$-list-recoverable (for some constant L) in linear time (we can actually achieve $L = 2$, but we will not be concerned with this in the following description). This is arguably the simplest list recovering setting: there is no noise, i.e., we are guaranteed that all codeword symbols lie in the respective input sets, and the sets have the smallest level of ambiguity (they identify the codeword symbol as one of two possible values). Below we closely follow the description in [33, Section 3.2].

The $(0,2,L)$-list-recoverable code C' will be the code $G(C)$ based on the following two components:

(1) A code $C \subset \Sigma^n$ that has relative distance greater than 0.9, has constant rate, and which can be decoded from 90% erasures in linear time.[1] Such a code has been constructed in [3].

(2) An $n \times n$ bipartite graph $G = (A,B,E)$ with the following *fault tolerance* property: if we delete $n/10$ nodes in A, then the square of the remaining graph has a connected component containing more than, say, $n/10$ nodes in A. It was shown in [2] that such a graph can be obtained by taking a double cover of a Ramanujan graph with sufficiently high (constant) degree d.

Our goal is to show that given sets $S_1, S_2, \ldots, S_n$, we can reconstruct the list of all codewords $c \in C$ such that $G(c)_j \in S_j$, $j = 1,\ldots,n$, in linear time. For each $i \in A$, $j \in B$, we define $L(i,j)$ to be the set of symbols in Σ that S_j "suggests" as potential symbols for c_i. More formally, $L(i,j)$ contains symbols a_k, such that $\langle a_1,\ldots,a_d\rangle \in S_j$ and $\Gamma_k(j) = i$.

Define $K_i = \cap_{(i,j)\in E} L(i,j)$. We assume that all K_is are non-empty, since otherwise no codeword compatible with the S_is exists. Define I to be the set of indices $i \in A$ such that K_i has size 1. For such is, denote the symbol in K_i by x_i.

Let c be a codeword of C such that $G(c)_i \in S_j$, $j = 1,\ldots,n$. Our goal is to find each such c. Clearly, for all $i \in I$, we must have $c_i = x_i$. The

[1] Under the erasure noise model, certain symbols are lost during transmission, and the rest are received intact. The receiver knows the locations of the erasures as well as of the received symbols. The goal is to decode the erased symbols.

decoding procedure consists of two cases. The first case occurs when the set I has size at least $n/10$. In this case, we know at least 10% of symbols of c, and thus we can recover c using the linear-time erasure-decoding algorithm for the code C. It remains to consider the second case, when the size of the set I is smaller than $n/10$. Consider any $i \notin I$. Observe that, for all $(i,j) \in E$, all sets $L(i,j)$ must be equal to K_i. The set K_i contains two distinct symbols that are candidates for c_i. Note that although for each c only one of this symbols is correct, each symbol in K_i can be correct for *some* codeword c. From now on, we will consider each K_i (respectively S_j) as ordered pairs of two symbols $(K_i)_0$ and $(K_i)_1$ (respectively $(S_j)_0$ and $(S_j)_1$), for an arbitrary ordering.

To recover c from the K_is, we create an auxiliary graph HH. Intuitively, the graph HH is obtained by "splitting" each node in G into two nodes, each corresponding to two decoding options, and then putting edges between compatible options. Formally, $HH = (A',B',E')$ is a bipartite graph such that $A' = A \times \{0,1\}$ and $B' = B \times \{0,1\}$. For each $i \notin I$, and $(i,j) \in E$, let k be such that $\Gamma_k(j) = i$ (i.e., j is the k-th neighbor of i). Then the edge set E' contains edges $\{(i,0),(j,t)\}$ and $\{(i,1),(j,1-t)\}$, where $t \in \{0,1\}$ is such that $(K_i)_0 = ((S_j)_t)_k$.

Define $V(c) = \{(i,t) : i \notin I, c_i = (K_i)_t\}$, i.e., the subgraph of HH that corresponds to the codeword c. In other words, $V(c)$ contains the nodes in A' that are compatible with c. The key fact, easily proved by induction, is that if $(i,t) \in V(c)$, and $(i',t') \in A'$ is reachable from (i,t) in HH, then $(i',t') \in V(c)$. Hence $V(c)$ will be the union of the vertices in A' that belong to some subset of connected components of HH. The fault-tolerance property of G can be shown to imply that one of these connected components must contain at least $n/10$ vertices. Therefore, if we enumerate all large connected components of HH, one of them will give us a large subset S' of $V(c)$ (of size at least $n/10$). Given S', we can recover c from a vector x such that x_i is equal to $(K_i)_t$ if $(i,t) \in S'$, and is declared as an erasure otherwise (note that the fraction of erasures is at most 0.9).

Since the graph H has only $O(n)$ edges, all its connected components, and in particular the large ones, can be found in $O(n)$ time. Thus, the whole decoding process can be performed in linear time. In addition, encoding C' can be done in linear time as well if C is linear-time encodable.

5.3 List recovering from small but positive noise rate

We now turn to the list recovering problem when the input list size is greater than 2, and more significantly when we allow a small fraction of lists to be erroneous. The following result is shown in [33]:

Theorem 5.2. For every integer $\ell \geqslant 1$, there exist $R_\ell > 0$, $\gamma_\ell > 0$, and a finite alphabet Σ_ℓ for which there is an explicit family of codes of rate $R(\ell)$ over alphabet Σ_ℓ that are encodable as well as $(\gamma_\ell, \ell, \ell)$-list-recoverable in linear time.

Combining the above with Lemma 5.1, one gets the following result of Guruswami and Indyk [33] on linear-time list-decodable codes for correcting any desired constant fraction of errors.

Theorem 5.3. For every α, $0 < \alpha < 1$, there exists an explicit family of $(1-\alpha, O(1/\alpha))$-list-decodable codes of positive rate $R(\alpha) > 0$ that can be encoded as well as list decoded from a fraction $(1-\alpha)$ of errors in time linear in the block length.[2]

Below we will sketch the ideas behind the proof of Theorem 5.2. Both expanders and the notion of list recovering play a prominent role throughout the construction and proof in [33].

The construction $(\gamma_\ell, \ell, \ell)$-list recoverable codes proceeds recursively using a construction of $(\gamma_{\ell-1}, \ell-1, \ell-1)$-list-recoverable codes (γ_ℓ will recursively depend on $\gamma_{\ell-1}$). For the recursion, it is convenient to construct a slightly stronger object; for a fixed small β, we will construct $(\gamma_\ell, \beta, \ell, \ell)$-list recoverable codes that have the following property: Given a collection of sets S_i all but a β fraction of which satisfy $|S_i| \leqslant \ell$, there are at most ℓ codewords whose i-th symbol belongs to S_i for at least a fraction $(1-\gamma_\ell)$ of the locations. (The case $\beta = 0$ corresponds to $(\gamma_\ell, \ell, \ell)$-list-recovering.)

[2] The complexity is linear in the *unit cost* RAM model, see [33] for details.

A code C_ℓ which is $(\gamma_\ell, \beta, \ell, \ell)$-list recoverable is constructed as $C_\ell = G_2(G_1(C_{\ell-1}))$ where both G_1, G_2 are $n \times n$ bipartite constant-degree expanders with appropriate properties (say with degrees d_1, d_2 that could depend on ℓ). Let $G_1 = (X, Y, E_1)$ and $G_2 = (Y, Z, E_2)$, so that we identify the left side of G_2 with the right side of the bipartition of G_1 in a natural way. Clearly, linear time encodability as well as positive rate is maintained in this process since d_1, d_2 are constants. The alphabet of C_ℓ is $\Sigma_\ell = \Sigma_{\ell-1}^{d_1 d_2}$.

The list recovering of C_ℓ proceeds in two steps/cases as described below. Note that the input is a collection of sets $S_1, S_2, \dots, S_n$ with each $S_i \subseteq \Sigma_\ell$ where all but βn of them have at most ℓ elements. In the first step, each $i \in Y$ collects a set $L_2(i,j) \subseteq \Sigma_{\ell-1}^{d_1}$ of at most ℓ possible symbols from the set S_j for each of its d_2 neighbors, and then computes $K_i = \cap_{(i,j) \in E_2} L_2(i,j)$. If γ_ℓ is chosen sufficiently small, for each candidate codeword that has to be output, most of the K_is will have the correct symbol. Clearly $|K_i| \leqslant \ell$ for all i. Suppose that in fact at least βn of the K_is satisfy $|K_i| \leqslant \ell - 1$. In this case, intuitively we have made progress since the amount of "ambiguity" in those symbols has gone down from ℓ to $\ell - 1$. However, this happens only for a small fraction β of symbols, but this can be transferred to a large (say, $1 - \beta$) fraction of symbols of $C_{\ell-1}$ using the expander G_1 (the specific property needed for G_1 will thus be that any β fraction of nodes in Y are adjacent to at least a $(1 - \beta)$ fraction of nodes in X). We are now in a position to apply the list recovering algorithm for $C_{\ell-1}$ to finish the decoding.

The more difficult and interesting case is when $|K_i| < \ell$ for less than a fraction β of the nodes $i \in Y$. This means that for most nodes $i \in Y$, all the sets $L_2(i,j)$ are in fact equal to K_i (and contain ℓ distinct elements). One can then define a certain kind "consistency graph" $\tilde{H} = (Y \times \{1,2,\dots,\ell\}, Z \times \{1,2,\dots,\ell\}, \tilde{E})$ whose vertex set corresponds to the ℓ possible choices for the correct symbol at each node of Y, Z. The edge set is defined based on which symbol of $L_2(i,j)$ matches the r-th symbol of K_i for $r = 1, 2, \dots, \ell$ (as per some fixed ordering of the elements in each K_i and $L_2(i,j)$). The problem of finding a consistent codeword can then be cast as finding a very dense subgraph of $\tilde{H}$ that internally closely resembles a copy of the expander G_2. Using the fact that this subgraph resembles G_2 and that G_2 is a high quality

Ramanujan expander, Guruswami and Indyk [33] demonstrate how spectral techniques can be used to find such subgraphs and thus all the solution codewords in $O(n \log n)$ time. (This also suffices to get a linear-time decoding algorithm in the unit cost RAM model using codes over a growing alphabet.) The detailed formalism of the above vague description as well as the technical details of the spectral partitioning are quite complicated. We will therefore refrain from trying to explain the details further and instead point the interested reader to the paper [33] (see also [27, Section 6]).

We note that since ℓ reduces by 1 at each step, and there is at least a constant factor loss in rate at each recursive step, the best rate one can hope for using the above approach is $2^{-O(\ell)}$. However, due to the intricacies of the spectral partitioning step and the resulting dependencies amongst the various parameters, the rate achieved is in fact worse and is only $2^{-2^{O(\ell)}}$. For the case $\gamma_\ell = 0$, Guruswami and Indyk [34] present a more efficient recursive construction of linear-time codes that achieve a rate of $1/\ell^{O(1)}$ (this is accomplished by reducing ℓ by a constant multiplicative factor as opposed to the above additive amount at each step).

5.4 Other graph-based list decoding results

5.4.1 Constructions over smaller alphabets

Graph-based constructions have also been useful in obtaining good list-decodable codes over smaller alphabets compared with purely algebraic codes.

For example, consider using the construction scheme of Definition 5.2 with the following components C, G:

- C is a rate $\Omega(\varepsilon)$ code that is $(1/2, \Theta(1/\varepsilon), \Theta(1/\varepsilon))$-list recoverable. Such a code can be constructed via an appropriate concatenation involving a rate $\Theta(\varepsilon)$ outer RS code, and a good list-recoverable inner code found by brute-force search (see [31] for details).
- G is an $n \times n$ bipartite graph of degree $d = O(1/\varepsilon)$ with the property that any subset of ε fraction of nodes on the right

have a neighborhood that spans more than half the nodes on the left.

It is shown in [31] that this yields $(1-\varepsilon, O(1/\varepsilon))$-list-decodable codes of rate $\Omega(\varepsilon^2)$. This matches the rate achieved by RS codes (up to a constant factor), but the advantage is that the alphabet size can be made a constant that depends only on ε. (Certain algebraic–geometric codes achieve a similar result, but this relies on much deeper mathematics and a complicated decoding algorithm.)

5.4.2 Juxtaposed codes

A different method to achieve a similar goal as above, in fact with even better alphabet sizes (for a slight worsening of the rate), was developed in [32]. The authors used the name "juxtaposed codes" for the resulting constructions. The basic idea behind juxtaposed codes is to encode the same message using several concatenated codes, each one of which works well for some error distribution, and then "juxtapose" symbols from these codes together to obtain a new code over a larger alphabet. More concretely, in this approach, multiple Reed–Solomon codes (of varying rates) are concatenated with several different inner codes (of varying rate and list decodability). Corresponding to each Reed–Solomon and inner code pair, we get one concatenated codeword, and the final encoding of a message is obtained by juxtaposing together the symbols from the various individual concatenated codewords.

The purpose of using multiple concatenated codes is that depending on the distribution of errors in the received word, the portions of it corresponding to a significant fraction of a particular inner encoding will have relatively few errors (the specific inner code for which this happens will depend on the level of non-uniformity in the distribution of errors). These can then be decoded to provide useful information about a large fraction of symbols to the decoder of the corresponding outer Reed–Solomon code. Essentially, depending on how (non)-uniformly the errors are distributed, a certain concatenated code "kicks in" and enables recovery of the message. The use of multiple concatenated codes reduces the rate compared to the expander based constructions, but it turns out this technique gives gains in the alphabet size. For example, for any integer $a \geqslant 1$, one can construct list-decodable up to a fraction $(1-\varepsilon)$ of errors with rate $\Omega(\varepsilon^{2(1+1/a)})$ over an alphabet of size $O(1/\varepsilon^a)$. For $a \leqslant 3$, this even beats the alphabet size (of $O(1/\varepsilon^4)$) achieved by algebraic–geometric codes, albeit with a worse rate. We refer the reader to the original paper [32] for further details.

5.4.3 Extractor-based codes

Using certain randomness extractors as the list-recoverable code in the general scheme of Lemma 5.1, in [26] the author obtained explicit list-decodable codes of close-to-optimal rate to correct a fraction $(1-\varepsilon)$ of errors in *sub-exponential* time (specifically, the achieved rate was $\varepsilon/\log^{O(1)}(1/\varepsilon)$, and recall that the best possible rate one could hope for is ε). This result was the first to beat the quadratic (rate ε^2) barrier for list decoding from a fraction $(1-\varepsilon)$ of errors. (As we saw in the previous chapter, Reed–Solomon codes achieve a rate ε^2 with polynomial time list decoding. Moreover, this is the best that can be achieved by appealing only to the Johnson bound on list decoding radius.) Though the result of [26] is obsolete in light of the capacity-achieving codes we will construct in Part II of this survey, it hints that expander-based constructions, perhaps in conjunction with other coding techniques, have the potential of not only yielding faster algorithms, but also good trade-offs between rate and decoding radius.

Part II

Achieving List Decoding Capacity

6

Folded Reed–Solomon Codes

This part of the survey gives some exciting recent progress that achieves the capacity of list decoding over large alphabets. In this chapter, we present a simple variant of Reed–Solomon codes called folded Reed–Solomon codes for which we can beat the $1 - \sqrt{R}$ decoding radius, we achieved for RS codes in Chapter 4. In fact, by choosing parameters suitably, we can decode close to the optimal fraction $1 - R$ of errors with rate R. In the next chapter, we will discuss techniques that let us achieve a similar result over an alphabet of constant size that depends only on the distance to list decoding capacity.

The starting point for the above capacity-achieving result is the breakthrough work of Parvaresh and Vardy [62] who described a novel variant of Reed–Solomon codes together with a new decoding algorithm. While the new decoding algorithm led to improvements over the decoding radius of $1 - \sqrt{R}$, it only did so for low rates (specifically, for $R < 1/16$). Subsequently, Guruswami and Rudra [37] proved that yet another variant of Reed–Solomon codes, namely "folded" RS codes that are compressed versions of certain Parvaresh–Vardy codes, are able to leverage the PV algorithm, essentially as is, but on codes of high rate. Together, this gives explicit codes with polynomial time decoding algorithms that achieve list decoding capacity.

In this chapter, we will describe this combined code and algorithm. We note that this presentation deviates significantly from the historical development in the original papers [37,62], in that we are using the benefit of hindsight to give a self-contained, and hopefully simpler, presentation. The last section of this chapter contains more comprehensive bibliographic notes on the original development of this material.

6.1 Description of folded codes

Consider a Reed–Solomon code $C = \mathsf{RS}_{\mathbb{F},\mathbb{F}^*}[n,k]$ consisting of evaluations of degree k polynomials over $\mathbb{F}$ at the set $\mathbb{F}^*$ of nonzero elements of $\mathbb{F}$. Let $q = |\mathbb{F}| = n+1$. Let γ be a generator of the multiplicative group $\mathbb{F}^*$, and let the evaluation points be ordered as $1, \gamma, \gamma^2, \ldots, \gamma^{n-1}$. Using all nonzero field elements as evaluation points is one of the most commonly used instantiations of Reed–Solomon codes.

Let $m \geqslant 1$ be an integer parameter called the *folding parameter*. For ease of presentation, we will assume that m divides $n = q-1$.

Definition 6.1. (Folded Reed–Solomon code) The m-folded version of the RS code C, denoted $\mathsf{FRS}_{\mathbb{F},\gamma,m,k}$, is a code of block length $N = n/m$ over $\mathbb{F}^m$. The encoding of a message $f(X)$, a polynomial over $\mathbb{F}$ of degree at most k, has its j-th symbol, for $0 \leqslant j < n/m$, the m-tuple $(f(\gamma^{jm}), f(\gamma^{jm+1}), \ldots, f(\gamma^{jm+m-1}))$. In other words, the codewords of $C' = \mathsf{FRS}_{\mathbb{F},\gamma,m,k}$ are in one-one correspondence with those of the RS code C and are obtained by bundling together consecutive m-tuple of symbols in codewords of C.

We illustrate the above construction for the choice $m = 4$ in Figure 6.1. The polynomial $f(X)$ is the message, whose Reed–Solomon encoding consists of the values of f at $x_0, x_1, \ldots, x_{n-1}$ where $x_i = \gamma^i$. Then, we perform a folding operation by bundling together tuples of four symbols to give a codeword of length $n/4$ over the alphabet $\mathbb{F}^4$.

Note that the folding operation does not change the rate R of the original Reed–Solomon code. The relative distance of the folded RS code also meets the Singleton bound and is at least $1 - R$.

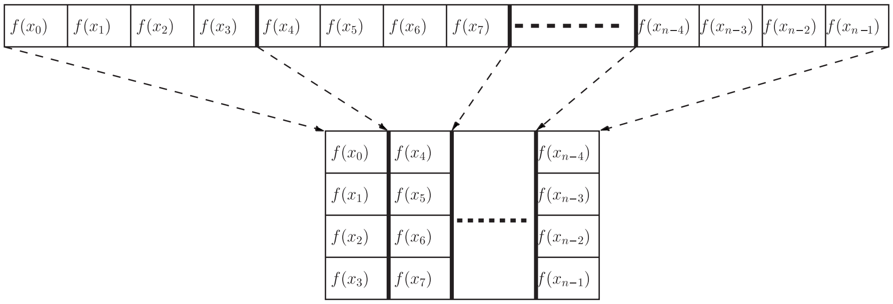

Fig. 6.1 Folding of the Reed–Solomon code with parameter $m = 4$.

6.2 Why might folding help?

Since folding seems like such a simplistic operation, and the resulting code is essentially just a RS code but viewed as a code over a large alphabet, let us now understand why it can possibly give hope to correct more errors compared with the bound for RS codes.

Consider the above example with folding parameter $m = 4$. First of all, decoding the folded RS code up to a fraction p of errors is certainly not harder than decoding the RS code up to the same fraction p of errors. Indeed, we can "unfold" the received word of the folded RS code and treat it as a received word of the original RS code and run the RS list decoding algorithm on it. The resulting list will certainly include all folded RS codewords within distance p of the received word, and it may include some extra codewords which we can, of course, easily prune.

In fact, decoding the folded RS code is a strictly easier task. To see why, say we want to correct a fraction $1/4$ of errors. Then, if we use the RS code, our decoding algorithm ought to be able to correct an error pattern which corrupts every fourth symbol in the RS encoding of $f(X)$ (i.e., corrupts $f(x_{4i})$ for $0 \leqslant i < n/4$). However, after the folding operation, this error pattern corrupts every one of the symbols over the larger alphabet $\mathbb{F}^4$, and thus need not be corrected. In other words, for the same fraction of errors, the folding operation reduces the total number of error patterns that need to be corrected, since the channel has less flexibility in how it may distribute the errors.

It is of course far from clear how one may exploit this to actually correct more errors. To this end, algebraic ideas that exploit the specific nature of the folding and the relationship between a polynomial $f(X)$ and its shifted counterpart $f(\gamma X)$ will be used. These will be clear once we describe our algorithms later in the chapter.

We note that above "simplification" of the channel is not attained for free, since the alphabet size increases after the folding operation. For folding parameter m that is an absolute constant, the increase in alphabet size is moderate and the alphabet remains polynomially large in the block length. (Recall that the RS code has an alphabet size that is linear in the block length.) Still, having an alphabet size that is a large polynomial is somewhat unsatisfactory. Fortunately, our alphabet reduction techniques in the next chapter can handle polynomially large alphabets, so this does not pose a big problem.

6.3 Trivariate interpolation based decoding

In the bivariate interpolation based decoding algorithm for RS codes, the key factor driving the agreement parameter t needed for the decoding to be successful was the ($(1,k)$-weighted) degree D of the polynomial $Q(X,Y)$. Our quest for an improved algorithm will be based on trying to lower this degree D by using more degrees of freedom in the interpolation. Specifically, we will try to use *trivariate interpolation* of a polynomial $Q(X,Y_1,Y_2)$ through n points in $\mathbb{F}^3$. As we will see, this enables performing the interpolation with $D = O((k^2n)^{1/3})$ which is much smaller than the $\Theta(\sqrt{kn})$ bound for bivariate interpolation. In principle, this could lead to an algorithm that works for agreement fraction $R^{2/3}$ instead of $R^{1/2}$. Of course, additional ideas are needed to make this approach work, and we now turn to developing such an algorithm in detail.

6.3.1 Facts about trivariate interpolation

We begin with some basic definitions and facts concerning trivariate polynomials which are straightforward extensions of the bivariate counterparts.

Definition 6.2. For a polynomial $Q(X,Y_1,Y_2) \in \mathbb{F}[X,Y_1,Y_2]$, its $(1,k,k)$-weighted degree is defined to be the maximum value of $\ell + kj_1 + kj_2$ taken over all monomials $X^\ell Y_1^{j_1} Y_2^{j_2}$ that occur with a nonzero coefficient in $Q(X,Y_1,Y_2)$.

Definition 6.3. (Multiplicity of zeroes) A polynomial $Q(X,Y_1,Y_2)$ over $\mathbb{F}$ is said to have a zero of multiplicity $r \geqslant 1$ at a point $(\alpha,\beta_1,\beta_2) \in \mathbb{F}^3$ if $Q(X+\alpha, Y_1+\beta_1, Y_2+\beta_2)$ has no monomial of degree less than r with a nonzero coefficient. (The degree of the monomial $X^i Y_1^{j_1} Y_2^{j_2}$ equals $i + j_1 + j_2$.)

Lemma 6.1. Let $\{(\alpha_i, y_{i1}, y_{i2})\}_{i=1}^n$ be an arbitrary set of n triples from $\mathbb{F}^3$. Let $Q(X,Y_1,Y_2) \in \mathbb{F}[X,Y_1,Y_2]$ be a nonzero polynomial of $(1,k,k)$-weighted degree at most D that has a zero of multiplicity r at $(\alpha_i, y_{i1}, y_{i2})$ for every $i \in [n]$. Let $f(X), g(X)$ be polynomials of degree at most k such that for at least $t > D/r$ values of $i \in [n]$, we have $f(\alpha_i) = y_{i1}$ *and* $g(\alpha_i) = y_{i2}$. Then, $Q(X, f(X), g(X)) \equiv 0$.

Proof. The proof is very similar to that of Lemma 4.6. If we define $R(X) = Q(X, f(X), g(X))$, then $R(X)$ is a univariate polynomial of degree at most D, and for every $i \in [n]$ for which $f(\alpha_i) = y_{i1}$ *and* $g(\alpha_i) = y_{i2}$, $(X - \alpha_i)^r$ divides $R(X)$. Therefore if $rt > D$, then $R(X)$ has more roots (counting multiplicities) than its degree, and so it must be the zero polynomial. □

Lemma 6.2. Given an arbitrary set of n triples $\{(\alpha_i, y_{i1}, y_{i2})\}_{i=1}^n$ from $\mathbb{F}^3$ and an integer parameter $r \geqslant 1$, there exists a nonzero polynomial $Q(X,Y_1,Y_2)$ over $\mathbb{F}$ of $(1,k,k)$-weighted degree at most D such that $Q(X,Y_1,Y_2)$ has a zero of multiplicity r at $(\alpha_i, y_{i1}, y_{i2})$ for all $i \in [n]$, provided $\frac{D^3}{6k^2} > n\binom{r+2}{3}$. Moreover, we can find such a $Q(X,Y_1,Y_2)$ in time polynomial in n, r by solving a system of homogeneous linear equations over $\mathbb{F}$.

Proof. This is just the obvious trivariate extension of Lemma 4.7. The condition that $Q(X,Y_1,Y_2)$ has a zero of multiplicity r at a point amounts to $\binom{r+2}{3}$ homogeneous linear conditions in the coefficients of Q. The number of monomials in $Q(X,Y_1,Y_2)$ equals the number, say $N_3(k,D)$, of triples (i,j_1,j_2) of nonnegative integers which obey $i + kj_1 + kj_2 \leqslant D$. One can show that the number $N_3(k,D)$ is at least as large as the volume of the three-dimensional region $\{x + ky_1 + ky_2 \leqslant D \mid x,y_1,y_2 \geqslant 0\} \subset \mathbb{R}^3$ [62]. An easy calculation shows that the latter volume equals $\frac{D^3}{6k^2}$. Hence, if $\frac{D^3}{6k^2} > n\binom{r+2}{3}$, then the number of unknowns exceeds the number of equations, and we are guaranteed a nonzero solution. □

6.3.2 Using trivariate interpolation for Folded RS codes

Let us now see how trivariate interpolation can be used in the context of decoding the folded RS code $C' = \mathsf{FRS}_{\mathbb{F},\gamma,m,k}$ of block length $N = (q-1)/m$. (Throughout this section, we denote $q = |\mathbb{F}|$, and $n = q-1$.) Given a received word $\mathbf{z} \in (\mathbb{F}^m)^N$ for C' that needs to be list decoded, we define $\mathbf{y} \in \mathbb{F}^n$ to be the corresponding "unfolded" received word. (Formally, let the j-th symbol of $\mathbf{z}$ be $(z_{j,0},\ldots,z_{j,m-1})$ for $0 \leqslant j < N$. Then $\mathbf{y}$ is defined by $y_{jm+l} = z_{j,l}$ for $0 \leqslant j < N$ and $0 \leqslant l < m$.)

Suppose $f(X)$ is a polynomial whose encoding agrees with $\mathbf{z}$ on at least t locations. Then, here is an obvious but important observation:

> For at least $t(m-1)$ values of i, $0 \leqslant i < n$, *both* the equalities $f(\gamma^i) = y_i$ and $f(\gamma^{i+1}) = y_{i+1}$ hold.

Define the notation $g(X) = f(\gamma X)$. Therefore, if we consider the n triples $(\gamma^i, y_i, y_{i+1}) \in \mathbb{F}^3$ for $i = 0,1,\ldots,n-1$ (with the convention $y_n = y_0$), then for at least $t(m-1)$ triples, we have $f(\gamma^i) = y_i$ and $g(\gamma^i) = y_{i+1}$. This suggests that interpolating a polynomial $Q(X,Y_1,Y_2)$ through these n triples and employing Lemma 6.1, we can hope that $f(X)$ will satisfy $Q(X,f(X),f(\gamma X)) = 0$, and then somehow use this to find $f(X)$. We formalize this in the following lemma. The proof follows immediately from the preceding discussion and Lemma 6.1.

Lemma 6.3. Let $\mathbf{z} \in (\mathbb{F}^m)^N$ and let $\mathbf{y} \in \mathbb{F}^n$ be the unfolded version of $\mathbf{z}$. Let $Q(X, Y_1, Y_2)$ be any nonzero polynomial over $\mathbb{F}$ of $(1, k, k)$-weighted degree at D which has a zero of multiplicity r at (γ^i, y_i, y_{i+1}) for $i = 0, 1, \ldots, n-1$. Let t be an integer such that $t > \frac{D}{(m-1)r}$. Then every polynomial $f(X) \in \mathbb{F}[X]$ of degree at most k whose encoding according to $\mathsf{FRS}_{\mathbb{F},\gamma,m,k}$ agrees with $\mathbf{z}$ on at least t locations satisfies $Q(X, f(X), f(\gamma X)) \equiv 0$.

Lemmas 6.2 and 6.3 motivate the following approach to list decoding the folded RS code $\mathsf{FRS}_{\mathbb{F},\gamma,m,k}$. Here $\mathbf{z} \in (\mathbb{F}^m)^N$ is the received word and $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1}) \in \mathbb{F}^n$ is its unfolded version. The algorithm uses an integer multiplicity parameter $r \geqslant 1$, and is intended to work for an agreement parameter $1 \leqslant t \leqslant N$.

Algorithm trivariate-FRS-decoder:

Step 1 (Trivariate interpolation) Define the degree parameter

$$D = \left\lfloor \sqrt[3]{k^2 n r(r+1)(r+2)} \right\rfloor + 1. \tag{6.1}$$

Interpolate a nonzero polynomial $Q(X, Y_1, Y_2)$ with coefficients from $\mathbb{F}$ with the following two properties: (i) Q has $(1, k, k)$-weighted degree at most D, and (ii) Q has a zero of multiplicity r at (γ^i, y_i, y_{i+1}) for $i = 0, 1, \ldots, n-1$ (where $y_n = y_0$). (Lemma 6.2 guarantees the feasibility of this step as well as its computability in time polynomial in n, r.)

Step 2 (Trivariate "root-finding") Find a list of all degree $\leqslant k$ polynomials $f(X) \in \mathbb{F}[X]$ such that $Q(X, f(X), f(\gamma X)) = 0$. Output those whose encoding agrees with $\mathbf{z}$ on at least t locations.

Ignoring the time complexity of Step 2 or the size of the output list for now, we can already claim the following concerning the error-correction performance of this strategy.

Lemma 6.4. The algorithm trivariate-FRS-decoder successfully list decodes the folded Reed–Solomon code $\mathsf{FRS}_{\mathbb{F},\gamma,m,k}$ up to a radius

$$N - \left\lceil N\frac{m}{m-1}\sqrt[3]{\frac{k^2}{n^2}\left(1+\frac{1}{r}\right)\left(1+\frac{2}{r}\right)}\right\rceil - 2.$$

Proof. By Lemma 6.3, we know that any $f(X)$ whose encoding agrees with $\mathbf{z}$ on t or more locations will be output in Step 2, provided $t > \frac{D}{(m-1)r}$. For the choice of D in (6.1), this condition is met for the choice $t = 1 + \left\lfloor \sqrt[3]{\frac{k^2 n}{(m-1)^3}\left(1+\frac{1}{r}\right)\left(1+\frac{2}{r}\right)} + \frac{1}{(m-1)r}\right\rfloor$. The decoding radius is equal to $N - t$, and recalling that $n = mN$, we get bound claimed in the lemma. □

The rate of the folded Reed–Solomon code is $R = ((k+1)/n) > k/n$, and so the fraction of errors corrected is $1 - \frac{m}{m-1}R^{2/3}$. Letting the parameter m grow, we can approach a decoding radius of $1 - R^{2/3}$.

6.4 Root-finding step

In light of the above discussion, the only missing piece in our decoding algorithm is an efficient way to solve the following trivariate "root-finding" type problem:

> Given a nonzero polynomial $Q(X, Y_1, Y_2)$ with coefficients from a finite field $\mathbb{F}$ of size q, a primitive element γ of the field $\mathbb{F}$, and an integer parameter $k < q - 1$, find a list of all polynomials $f(X)$ of degree at most k such that $Q(X, f(X), f(\gamma X)) \equiv 0$.

The following simple algebraic lemma is at the heart of our solution to this problem.

Lemma 6.5. Let $\mathbb{F}$ be the field $\mathbb{F}_q$ of size q, and let γ be a primitive element that generates its multiplicative group. Then we have the following two facts:

(1) The polynomial $h(X) \stackrel{\mathrm{def}}{=} X^{q-1} - \gamma$ is irreducible over $\mathbb{F}$.
(2) Every polynomial $f(X) \in \mathbb{F}[X]$ of degree less than $q-1$ satisfies $f(\gamma X) = f(X)^q \mod h(X)$.

Proof. The fact that $h(X) = X^{q-1} - \gamma$ is irreducible over $\mathbb{F}_q$ follows from a known, precise characterization of all irreducible binomials, i.e., polynomials of the form $X^a - b$, see for instance [55, Chapter 3, Section 5]. For completeness, and since this is an easy special case, we now prove this fact. Suppose $h(X)$ is not irreducible and some irreducible polynomial $f(X) \in \mathbb{F}[X]$ of degree b, $1 \leqslant b < q-1$, divides it. Let ζ be a root of $f(X)$ in the extension field $\mathbb{F}_{q^b}$. We then have $\zeta^{q^b-1} = 1$. Also, $f(\zeta) = 0$ implies $\zeta^{q-1} = \gamma$. These equations together imply $\gamma^{\frac{q^b-1}{q-1}} = 1$. Now, γ is primitive in $\mathbb{F}_q$, so that $\gamma^m = 1$ iff m is divisible by $(q-1)$. We conclude that $q-1$ must divide $1 + q + q^2 + \cdots + q^{b-1}$. This is, however, impossible since $1 + q + q^2 + \cdots + q^{b-1} \equiv b \pmod{(q-1)}$ and $0 < b < q-1$. This contradiction proves that $h(X)$ has no such factor of degree less than $q-1$, and is therefore irreducible.

For the second part, we have the simple but useful identity $f(X)^q = f(X^q)$ that holds for all polynomials in $\mathbb{F}_q[X]$. Therefore, $f(X)^q - f(\gamma X) = f(X^q) - f(\gamma X)$. The latter polynomial is clearly divisible by $X^q - \gamma X$, and thus also by $X^{q-1} - \gamma$. Hence $f(X)^q \equiv f(\gamma X) \pmod{h(X)}$ which implies that $f(X)^q \mod h(X) = f(\gamma X)$ since the degree of $f(\gamma X)$ is less than $q-1$. □

Armed with this lemma, we are ready to tackle the trivariate root-finding problem. This is in analogy with Section 4.5.2.

Lemma 6.6. There is a deterministic algorithm that on input a finite field $\mathbb{F}$ of size q, a primitive element γ of the field $\mathbb{F}$, a nonzero polynomial $Q(X,Y_1,Y_2) \in \mathbb{F}[X,Y_1,Y_2]$ of degree less than q in Y_1, and an integer parameter $k < q-1$, outputs a list of all polynomials $f(X)$ of degree at most k satisfying the condition $Q(X,f(X),f(\gamma X)) \equiv 0$. The algorithm has runtime polynomial in q.

Proof. Let $h(X) = X^{q-1} - \gamma$. We know by Lemma 6.5 that $h(X)$ is irreducible. We first divide out the largest power of $h(X)$ that divides $Q(X,Y_1,Y_2)$ to obtain $Q_0(X,Y_1,Y_2)$, where $Q(X,Y_1,Y_2) = h(X)^b Q_0(X,Y_1,Y_2)$ for some $b \geqslant 0$ and $h(X)$ does not divide $Q_0(X,Y_1,Y_2)$. Clearly, if $f(X)$ satisfies $Q(X,f(X),f(\gamma X)) = 0$, then $Q_0(X,f(X),f(\gamma X)) = 0$ as well, so we will work with Q_0 instead of Q. Let us view $Q_0(X,Y_1,Y_2)$ as a polynomial $T_0(Y_1,Y_2)$ with coefficients from $\mathbb{F}[X]$. Further, reduce each of the coefficients modulo $h(X)$ to get a polynomial $T(Y_1,Y_2)$ with coefficients from the extension field $\tilde{\mathbb{F}} \overset{\text{def}}{=} \mathbb{F}[X]/(h(X))$ (this is a field since $h(X)$ is irreducible over $\mathbb{F}$). We note that $T(Y_1,Y_2)$ is a nonzero polynomial since $Q_0(X,Y_1,Y_2)$ is not divisible by $h(X)$.

In view of Lemma 6.5, it suffices to find degree $\leqslant k$ polynomials $f(X)$ satisfying $Q_0(X,f(X),f(X)^q) \pmod{h(X)} = 0$. In turn, this means it suffices to find elements $\Gamma \in \tilde{\mathbb{F}}$ satisfying $T(\Gamma,\Gamma^q) = 0$. If we define the univariate polynomial $R(Y_1) \overset{\text{def}}{=} T(Y_1,Y_1^q)$, this is equivalent to finding all $\Gamma \in \tilde{\mathbb{F}}$ such that $R(\Gamma) = 0$, or in other words the roots in $\tilde{\mathbb{F}}$ of $R(Y_1)$.

Now $R(Y_1)$ is a nonzero polynomial since $R(Y_1) = 0$ iff $Y_2 - Y_1^q$ divides $T(Y_1,Y_2)$, and this cannot happen as $T(Y_1,Y_2)$ has degree less than q in Y_1. The degree of $R(Y_1)$ is at most dq where d is the total degree of $Q(X,Y_1,Y_2)$. The characteristic of $\tilde{\mathbb{F}}$ is at most q, and its degree over the underlying prime field is at most $q \log q$. Therefore, we can find all roots of $R(Y_1)$ by a deterministic algorithm running in time polynomial in d,q [6] (see discussion in Section 4.5.2). Each of the roots will be a polynomial in $\mathbb{F}[X]$ of degree less than $q-1$. Once we find all the roots, we prune the list and only output those roots $f(X)$ that have degree at most k and satisfy $Q_0(X,f(X),f(\gamma X)) = 0$. □

With this, we have a polynomial time implementation of the algorithm trivariate-FRS-decoder. There is the technicality that the degree of $Q(X,Y_1,Y_2)$ in Y_1 should be less than q. This degree is at most D/k, which by the choice of D in (6.1) is at most $(r+3)\sqrt[3]{n/k} < (r+3)q^{1/3}$. For a fixed r and growing q, the degree is much smaller than q. (In fact, for constant rate codes, the degree is a constant independent of n.) By letting m, r grow in Lemma 6.4, and recalling that the running time is polynomial in n, r, we can conclude the following main result of this section.

Theorem 6.7. For every $\varepsilon > 0$ and R, $0 < R < 1$, there is a family of m-folded Reed–Solomon codes for $m = O(1/\varepsilon)$ which have rate at least R and which can be list decoded up to a fraction $1-(1+\varepsilon)R^{2/3}$ of errors in time polynomial in the block length and $1/\varepsilon$.

6.5 Codes approaching list decoding capacity

Given that trivariate interpolation improved the decoding radius achievable with rate R from $1 - R^{1/2}$ to $1 - R^{2/3}$, it is natural to attempt to use higher order interpolation to improve the decoding radius further. In this section, we discuss the (quite straightforward) technical changes needed for such a generalization.

Consider again the m-folded RS code $C' = \mathsf{FRS}_{\mathbb{F},\gamma,m,k}$ where $\mathbb{F} = \mathbb{F}_q$. Let s be an integer in the range $1 \leqslant s \leqslant m$. We will develop a decoding algorithm based on interpolating an $(s+1)$-variate polynomial $Q(X, Y_1, Y_2, \ldots, Y_s)$. The definitions of the $(1, k, k, \ldots, k)$-weighted degree (with k repeated s times) of Q and the multiplicity at a point $(\alpha, \beta_1, \beta_2, \ldots, \beta_s) \in \mathbb{F}^{s+1}$ are straightforward extensions of Definitions 6.2 and 6.3.

As before let $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1})$ be the unfolded version of the received word $\mathbf{z} \in (\mathbb{F}^m)^N$ of the folded RS code that needs to be decoded. For convenience, define $y_j = y_{j \bmod n}$ for $j \geqslant n$. Following algorithm trivariate-FRS-decoder, for suitable integer parameters D, r, the interpolation phase of the $(s+1)$-variate FRS decoder will fit a nonzero polynomial $Q(X, Y_1, \ldots, Y_s)$ with the following properties:

(1) It has $(1, k, k, \ldots, k)$-weighted degree at most D
(2) It has a zero of multiplicity r at $(\gamma^i, y_i, y_{i+1}, \ldots, y_{i+s-1})$ for $i = 0, 1, \ldots, n-1$.

The following is a straightforward generalization of Lemmas 6.2 and 6.3.

Lemma 6.8.

(1) Provided $\frac{D^{s+1}}{(s+1)!k^s} > n\binom{r+s}{s+1}$, a nonzero polynomial $Q(X,Y_1,\ldots,Y_s)$ with the above stated properties exists and moreover can be found in time polynomial in n and r^s.

(2) Let t be an integer such that $t > \frac{D}{(m-s+1)r}$. Then every polynomial $f(X) \in \mathbb{F}[X]$ of degree at most k whose encoding according to $\mathsf{FRS}_{\mathbb{F},\gamma,m,k}$ agrees with the received word $\mathbf{z}$ on at least t locations satisfies $Q(X,f(X),f(\gamma X),\ldots, f(\gamma^{s-1}X)) \equiv 0$.

Proof. The first part follows from (i) a simple lower bound on the number of monomials $X^aY_1^{b_1}\cdots Y_s^{b_s}$ with $a+k(b_1+b_2+\cdots+b_s) \leqslant D$, which gives the number of coefficients of $Q(X,Y_1,\ldots,Y_s)$, and (ii) an estimation of the number of $(s+1)$-variate monomials of total degree less than r, which gives the number of interpolation conditions per $(s+1)$-tuple.

The second part is similar to the proof of Lemma 6.3. If $f(X)$ has agreement on at least t locations of $\mathbf{z}$, then for at least $t(m-s+1)$ of the $(s+1)$-tuples $(\gamma^i, y_i, y_{i+1},\ldots,y_{i+s-1})$, we have $f(\gamma^{i+j}) = y_{i+j}$ for $j = 0,1,\ldots,s-1$. As in Lemma 6.1, we conclude that $R(X) \stackrel{\text{def}}{=} Q(X,f(X),f(\gamma X),\ldots,f(\gamma^{s-1}X))$ has a zero of multiplicity r at γ^i for each such $(s+1)$-tuple. Also, by design $R(X)$ has degree at most D. Hence if $t(m-s+1)r > D$, then $R(X)$ has more zeroes (counting multiplicities) than its degree, and thus $R(X) \equiv 0$. □

Note the lower bound condition on D above is met with the choice

$$D = \left\lfloor \left(k^s nr(r+1)\cdots(r+s)\right)^{1/(s+1)} \right\rfloor + 1. \tag{6.2}$$

The task of finding a list of all degree k polynomials $f(X) \in \mathbb{F}[X]$ satisfying $Q(X,f(X),f(\gamma X),\ldots,f(\gamma^{s-1}X)) = 0$ can be solved using ideas similar to the proof of Lemma 6.6. First, by dividing out by $h(X)$ enough times, we can assume that not all coefficients of $Q(X,Y_1,\ldots,Y_s)$, viewed as a polynomial in $Y_1, \dots, Y_s$ with coefficients in $\mathbb{F}[X]$, are divisible by $h(X)$. We can then go modulo $h(X)$ to get a nonzero polynomial $T(Y_1, Y_2, \dots, Y_s)$ over the extension field $\tilde{\mathbb{F}} = \mathbb{F}[X]/(h(X))$. Now, by Lemma 6.5, we have $f(\gamma^j X) = f(X)^{q^j} \mod h(X)$ for every $j \geqslant 1$. Therefore, the task at hand reduces to the problem of finding all roots $\Gamma \in \tilde{\mathbb{F}}$ of the polynomial $R(Y_1)$ where $R(Y_1) = T(Y_1, Y_1^q, \dots, Y_1^{q^{s-1}})$. There is the risk that $R(Y_1)$ is the zero polynomial, but it is easily seen that this cannot happen if the total degree of T is less than q. This will be the case since the total degree is at most D/k, which is at most $(r+s)(n/k)^{1/(s+1)} \ll q$.

The degree of the polynomial $R(Y_1)$ is at most q^s, and therefore all its roots in $\tilde{\mathbb{F}}$ can be found in $q^{O(s)}$ time. We conclude that the "root-finding" step can be accomplished in polynomial time.

The algorithm works for agreement $t > \frac{D}{(m-s+1)r}$, which for the choice of D in (6.2) is satisfied if

$$t \geqslant \left(1 + \frac{s}{r}\right) \frac{(k^s n)^{1/(s+1)}}{m - s + 1} + 2.$$

Recalling that the block length of the code is $N = n/m$ and the rate is $(k+1)/n$, the algorithm can decode a fraction of errors approaching

$$1 - \left(1 + \frac{s}{r}\right) \frac{m}{m - s + 1} R^{s/(s+1)}, \tag{6.3}$$

using lists of size at most q^s. By picking r, m large enough compared with s, the decoding radius can be made larger than $1 - (1+\zeta)R^{s/(s+1)}$ for any desired $\zeta > 0$. We state this result formally below.

Theorem 6.9. For every $\varepsilon > 0$, integer $s \geqslant 1$ and $0 < R < 1$, there is a family of m-folded Reed–Solomon codes for $m = O(s/\varepsilon)$, which have rate at least R and which can be list decoded up to a fraction $1 - (1+\varepsilon)R^{s/(s+1)}$ of errors in time $(Nm)^{O(s)}$, where N is the block length of the code. The alphabet size of the code as a function of the block length N is $(Nm)^{O(m)}$.

In the limit of large s (specifically, for $s = \Theta(\varepsilon^{-1} \log(1/R))$), the decoding radius approaches the list decoding capacity $1 - R$, leading to our main result of this chapter.

Theorem 6.10. [Explicit capacity-approaching codes] For every $\varepsilon > 0$ and $0 < R < 1$, there is a family of folded Reed–Solomon codes which have rate at least R and which can be list decoded up to a fraction $1 - R - \varepsilon$ of errors in time $(N/\varepsilon^2)^{O(\varepsilon^{-1}\log(1/R))}$, where N is the block length of the code. The alphabet size of the code as a function of the block length N is $(N/\varepsilon^2)^{O(1/\varepsilon^2)}$.

Remark 6. (Improvement in decoding radius) It is possible to slightly improve the bound of (6.3) to $1 - \left(1 + \frac{s}{r}\right)\left(\frac{mR}{m-s+1}\right)^{s/(s+1)}$ with essentially no effort. The idea is to use only a fraction $(m - s + 1)/m$ of the n $(s + 1)$-tuples for interpolation. Specifically, we omit the tuples $(\gamma^i, y_i, y_{i+1}, \ldots, y_{i+s-1})$ where $i \mod m > m - s$ in the interpolation conditions. The number of $(s + 1)$-tuples for which we have agreement remains at least $t(m - s + 1)$, since we only counted agreements on tuples $(\gamma^i, y_i, y_{i+1}, \ldots, y_{i+s-1})$ for $0 \leqslant i \mod m \leqslant m - s$. However, the number of interpolation conditions is now reduced to $N(m - s + 1) = n(m - s + 1)/m$. This translates into the stated improvement in error correction radius. For clarity of presentation, we simply chose to use all n tuples for interpolation.

6.6 List recovering

Exactly as mentioned in Section 4.4.1 for the case of RS codes, the above algorithms for folded RS codes work seamlessly for the list recovering setting. The performance of the decoder is dictated by the number n of interpolation points, and we did not use that the first coordinates of those tuples were distinct. Therefore, generalizing the bound of (6.3), for any integer $\ell \geqslant 1$, the m-folded RS code of rate R can be (p, ℓ, q^s)-list-recovered for $p = 1 - (1 + \frac{s}{r})\frac{m}{m-s+1}(\ell R^s)^{1/(s+1)}$. For every choice of s, ℓ, and $\varepsilon > 0$, letting r, m grow, we can make $p \geqslant 1 - (1 + \varepsilon)\ell^{1/(s+1)}R^{s/(s+1)}$. Using the choice $s = \Omega(\varepsilon^{-1}\log(\ell/R))$, we can achieve $p \geqslant 1 - R - \varepsilon$, and thus *independent of* ℓ! Therefore, the achievable rate for list recovering (for large enough alphabet size) depends only on the fraction of erroneous input lists but not on the size of the input lists. This feature will be crucially used in the next chapter to reduce the alphabet size needed to achieve capacity in Theorem 6.10.

Theorem 6.11. (Achieving capacity for list recovering) For every $0 < R < 1$, integer $\ell \geqslant 2$ and all small enough $\varepsilon > 0$, there is a family of folded Reed–Solomon codes (over fields of any desired characteristic) which have rate at least R and which can be $(1 - R - \varepsilon, \ell, ((N \log \ell)/\varepsilon^2)^{O(\varepsilon^{-1} \log(\ell/R))})$-list-recovered in time $((N \log \ell)/\varepsilon^2)^{O(\varepsilon^{-1} \log(\ell/R))}$, where N is the block length of the code. The alphabet size of the code as a function of the block length N is $\left(\frac{N \log \ell}{\varepsilon^2}\right)^{O(\varepsilon^{-2} \log \ell)}$.

We can also state a further extension to soft-decision decoding of folded RS codes. We skip the details here, which can be found in [37].

6.7 Bibliography and remarks

Two independent works by Coppersmith and Sudan [11] and Bleichenbacher, Kiayias, and Yung [7] considered the variant of RS codes where the message consists of two (or more) independent polynomials over $\mathbb{F}$, and the encoding consists of the joint evaluation of these polynomials at elements of $\mathbb{F}$ (so this defines a code over $\mathbb{F}^2$).[1] A naive way to decode these codes, which are also called "interleaved Reed–Solomon codes," would be to recover the two polynomials individually, by running separate instances of the RS decoder. Of course, this gives no gain over the performance of RS codes. The hope in these works was that something can possibly be gained by exploiting that errors in the two polynomials happen at "synchronized" locations. However, these works could not give any improvement over the $1 - \sqrt{R}$ bound known for RS codes for worst-case errors. Nevertheless, for *random errors*, where each error replaces the correct symbol by a uniform random field element, they were able to correct well beyond a fraction $1 - \sqrt{R}$ of errors. In fact, as the order of interleaving (i.e., number of independent polynomials)

[1] The resulting code is in fact just a Reed–Solomon code where the evaluation points belong to the subfield $\mathbb{F}$ of the extension field over $\mathbb{F}$ of degree two.

grows, the radius approaches the optimal value $1 - R$. This model of random errors is not very practical or interesting in a coding-theoretic setting, though the algorithms are interesting from an algebraic viewpoint.

The algorithm of Coppersmith and Sudan bears an intriguing relation to multivariate interpolation. Multivariate interpolation essentially amounts to finding a non-trivial linear dependence among the rows of a certain matrix (that consists of the evaluations of appropriate monomials at the interpolation points). The algorithm in [11], instead finds a non-trivial linear dependence among the *columns* of this same matrix! The positions corresponding to columns not involved in this dependence are erased (they correspond to error locations) and the codeword is recovered from the remaining symbols using erasure decoding.

In [61], Parvaresh and Vardy gave a heuristic decoding algorithm for these interleaved RS codes based on multivariate interpolation. However, the provable performance of these codes coincided with the $1 - \sqrt{R}$ bound for Reed–Solomon codes. The key obstacle in improving this bound was the following: for the case when the messages are pairs $(f(X), g(X))$ of polynomials, two algebraically independent relations were needed to identify both $f(X)$ and $g(X)$. The interpolation method could only provide one such relation in general (of the form $Q(X, f(X), g(X)) = 0$ for a trivariate polynomial $Q(X, Y, Z)$). This still left too much ambiguity in the possible values of $(f(X), g(X))$. (The approach in [61] was to find several interpolation polynomials, but there was no guarantee that they were not all algebraically dependent.)

Then, in [62], Parvaresh and Vardy put forth the ingenious idea of obtaining the extra algebraic relation essentially "for free" by enforcing it as an *a priori* condition satisfied at the encoder. Specifically, instead of letting the second polynomial $g(X)$ to be an independent degree k polynomial, their insight was to make it correlated with $f(X)$ by a specific algebraic condition, such as $g(X) = f(X)^d \mod h(X)$ for some integer d and an irreducible polynomial $h(X)$ of degree $k + 1$.

Then, once we have the interpolation polynomial $Q(X, Y, Z)$, $f(X)$ can be found as described in this chapter: Reduce the coefficients of $Q(X, Y, Z)$ modulo $h(X)$ to get a polynomial $T(Y, Z)$ with coefficients from $\mathbb{F}[X]/(h(X))$ and then find roots of the univariate polynomial

$T(Y,Y^d)$. This was the key idea in [62] to improve the $1-\sqrt{R}$ decoding radius for rates less than 1/16. For rates $R \to 0$, their decoding radius approached $1 - O(R\log(1/R))$.

The modification in using independent polynomials does not come for free, however. In particular, since one sends at least twice as much information as in the original RS code, there is no way to construct codes with rate more than 1/2 in the PV scheme. If we use $s \geqslant 2$ correlated polynomials for the encoding, we incur a factor $1/s$ loss in the rate. This proves quite expensive, and as a result the improvements over RS codes offered by these codes are only manifest at very low rates.

The central idea in the work of Guruswami and Rudra on list decoding folded RS codes [37] was to avoid this rate loss by making the correlated polynomial $g(X)$ essentially identical to the first (say $g(X) = f(\gamma X)$). Then the evaluations of $g(X)$ can be inferred as a simple cyclic shift of the evaluations of $f(X)$, so intuitively there is no need to explicitly include those too in the encoding. The folded RS encoding of $f(X)$ compresses all the needed information, without any extra redundancy for $g(X)$. In particular, from a received word that agrees with folded RS encoding of $f(X)$ in many places, we can infer a received word (with symbols in $\mathbb{F}^2$) that matches the value of both $f(X)$ and $f(\gamma X) = g(X)$ in many places, and then run the decoding algorithm of Parvaresh and Vardy.

The terminology of folded RS codes was coined in [52], where an algorithm to correct random errors in such codes was presented (for a noise model similar to the one used in [7, 11] that was mentioned earlier). The motivation was to decode RS codes from many random "phased burst" errors. Our decoding algorithm for folded RS codes can also be likewise viewed as an algorithm to correct beyond the $1-\sqrt{R}$ bound for RS codes if errors occur in large, phased bursts (the actual errors can be adversarial).

7

Achieving Capacity over Bounded Alphabets

The capacity-achieving codes from the previous chapter have two main shortcomings: (i) their alphabet size is a large polynomial in the block length, and (ii) the bound on worst-case list size as well as decoding time complexity grows as $n^{\Omega(1/\varepsilon)}$, where ε is the distance to capacity. In this chapter, we will remedy the alphabet size issue (Section 7.2). We begin by using the folded RS codes in a concatenation scheme to get good list-decodable binary codes.

7.1 Binary codes decodable up to Zyablov bound

The optimal list recoverability of the folded RS codes discussed in Section 6.6 plays a crucial role in establishing the following result concerning list decoding binary codes. The decoding radius achieved matches the standard "product" bound on the designed relative distance of binary concatenated codes, namely the product of the relative distance of an outer MDS code with the relative distance of an inner code that meets the Gilbert–Varshamov bound.

Theorem 7.1. For all $0 < R, r < 1$ and all $\varepsilon > 0$, there is a polynomial time constructible family of binary linear codes of rate at least $R \cdot r$ that can be list decoded in polynomial time up to a fraction $(1 - R)H^{-1}(1 - r) - \varepsilon$ of errors.

Proof. [Sketch] The idea is to use an appropriate concatenation scheme with an outer code C_1 and a binary linear inner code C_2. For C_1, we will use a folded RS code over a field of characteristic 2 as guaranteed by Theorem 6.11, with the following properties: (i) The rate of C_1 is at least R, (ii) it can be $(1 - R - \varepsilon, \ell, L(N))$-list-recovered in polynomial ($N^{O(1/\varepsilon^2)}$, where N is the block length) time for $\ell = \lceil 10/\varepsilon \rceil$, and (iii) its alphabet size is 2^M for $M = O(\varepsilon^{-3} \log N)$. The code C_2 will be a binary linear code of dimension M (so that it can be concatenated with C_1) and rate at least r which is (ρ, ℓ)-list decodable for $\rho = H^{-1}(1 - r - \varepsilon)$. Such a code is known to exist via a random coding argument that employs the semi-random method [30]. Also, a greedy construction of such a code by constructing its M basis elements in turn is known and takes $2^{O(M)}$ time. We conclude that the necessary inner code can be constructed in $N^{O(1/\varepsilon^3)}$, and thus polynomial, time. Note that the concatenated code is a binary linear code of rate at least $R \cdot r$.

The decoding algorithm proceeds in a natural way. Given a received word, we break it up into blocks corresponding to the various inner encodings by C_1. Each of these blocks is list decoded up to a radius ρ, returning a list of at most ℓ possible candidates for each outer codeword symbol. The outer code is then $(1 - R - \varepsilon, \ell, L(N))$-list recovered using these lists as input. To argue about the fraction of errors this algorithm corrects, we note that the algorithm fails to recover a codeword only if on more than a fraction $(1 - R - \varepsilon)$ of the inner blocks, the codeword differs from the received word on more than a fraction ρ of symbols. It follows that the algorithm correctly list decodes up to a radius $(1 - R - \varepsilon)\rho = (1 - R - \varepsilon)(H^{-1}(1 - r) - \varepsilon)$. □

Optimizing over the choice of inner and outer codes rates r, R in the above result, we can decode up to the so-called Zyablov bound, depicted in Figure 7.1.

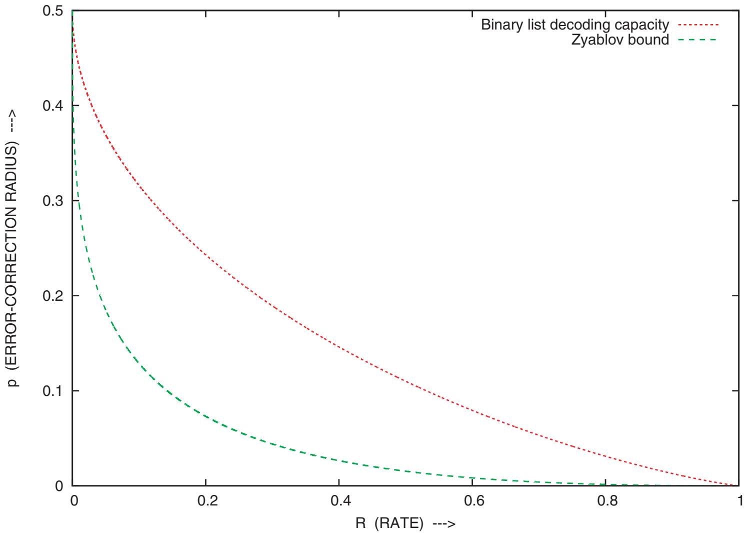

Fig. 7.1 Error-correction radius ρ of our algorithm for binary codes plotted against the rate R. The best possible trade-off, i.e., capacity, is $\rho = H^{-1}(1 - R)$, and is also plotted.

Remark 7. The construction time of the codes of Theorem 7.1 is $n^{\Omega(1/\varepsilon^3)}$, where n is the block length (this complexity bound arises due to the search for an appropriate inner code). A uniformly constructive family of binary codes, i.e., with construction time $f(\varepsilon)n^c$ for some c independent of ε, will be more desirable. The same applies to the decoding complexity.

7.2 Approaching capacity over constant-sized alphabets

We now show how to approach the list decoding capacity $1 - R$ with an alphabet size that is a constant depending only on the distance to capacity.

Theorem 7.2. (Main) For every R, $0 < R < 1$, every $\varepsilon > 0$, there is a polynomial time constructible family of codes over an alphabet of size $2^{O(\varepsilon^{-4}\log(1/\varepsilon))}$ that have rate at least R and which can be list decoded up to a fraction $(1 - R - \varepsilon)$ of errors in polynomial time.

Proof. [Sketch] The theorem can be proved using the code construction scheme used in [3, 35] for linear time unique decodable codes with optimal rate, with different components appropriate for list decoding plugged in. We only highlight the high level ideas and spare the details, which can be found in [35, 37].

As in the above result for binary codes, the list recoverability of folded RS codes will play a crucial role. To motivate the construction, imagine the following concatenation scheme to combine an outer code C_1 with an inner code C_2. For C_1, we use a folded RS code of rate close to 1 that can $(0, \ell, L)$-list-recovered for $\ell = \Theta(1/\varepsilon)$. Here we are exploiting the fact, mentioned in Section 6.6, that the size of the input lists ℓ does not figure in the price we pay in terms of rate.

For C_2, for some small $\delta > 0$, we will use a $(1 - R - \varepsilon/2, O(1/\varepsilon))$-list decodable code with near-optimal rate, say rate at least $(R + \varepsilon/4)$. As mentioned in Corollary 3.7, such a (non-linear) code can be shown to exist over an alphabet of size $2^{O(1/\varepsilon)}$ using random coding arguments. A naive brute-force for such a code, however, is too expensive. Fortunately, a similar guarantee holds for a random code drawn from a much smaller ensemble that was introduced and called pseudolinear codes in [31] (see also [28, Section 9.3]). This enables polynomial time construction of such an inner code C_2. The number of codewords in C_2 equals the alphabet size of the folded RS code C_1, which is polynomial in the block length (for each fixed ε). Therefore, C_2 can be list decoded by a brute-force search over all its codewords in polynomial time. The overall rate of the concatenated code is at least R since the rate of C_1 is very close to 1.

Now suppose a fraction $(1 - R - \varepsilon)$ errors occur on the symbols of the concatenated code. Assume an ideal scenario where these errors are uniformly distributed amongst all the inner blocks, i.e., no block internally suffers from more than a fraction $(1 - R - \varepsilon/2)$ of errors. Then a brute-force list decoding of each of the inner blocks can be used to return a list of at most $\ell = O(1/\varepsilon)$ symbols for each position of C_1. By the assumption on the distribution of errors, each of these lists will contain the correct symbol. Therefore, running the assumed list recovering algorithm for C_1 will recover the correct codeword.

Of course the above assumption on equitable distribution of errors amongst the blocks is not valid. The key idea is to somehow "simulate" such a distribution. For this purpose, the symbols of the concatenated codewords are further redistributed using an *expander graph* to produce a codeword over a larger alphabet (but with no further loss in rate). The "pseudorandom" properties of the expander ensures that for every error pattern affecting a fraction $(1 - R - \varepsilon)$ of positions, once symbols are distributed backwards, *most* (say, a fraction $1 - O(\sqrt{\varepsilon})$) of the blocks corresponding to inner codewords of C_2 incur at most a fraction $(1 - R - \sqrt{\varepsilon})$ of errors. If instead of a $(0, \ell, L)$-list-recoverable code C_1, we use a list-recoverable code that can tolerate a small $O(\sqrt{\varepsilon})$ fraction of erroneous lists (which is still possible with a very high rate of $1 - O(\sqrt{\varepsilon})$), the errors in decoding the few deviant inner blocks can be handled when list recovering C_1.

We skip the formal details that are not hard to work out and follow along the same lines as the arguments in [35]. □

8

Concluding Thoughts

We have surveyed the topic of list decoding with a focus on the recent advances in decoding algorithms for Reed–Solomon codes and close variants, culminating with a presentation of how to achieve the list decoding capacity over large alphabets. We conclude by mentioning some interesting open questions and directions for future work.

8.1 Improving list size of capacity-achieving codes

To list decode a fraction $1 - R - \varepsilon$ of errors with rate R, the decoding complexity as well as worst-case list size needed are $n^{\Omega(1/\varepsilon)}$. It remains a challenge to reduce both of these, and in particular, to achieve a list size bound that is independent of n. Recall that the existential results of Section 3.2.1 imply that a bound of $O(1/\varepsilon)$ suffices.

The large bound on list size for decoding folded RS codes arises due to the large degree (the field size q) of the algebraic relation between and $f(X)$ and its "shift" $f(\gamma X)$. The bound on the list size was derived from the degree of the final univariate polynomial whose roots contain all the solution messages. The degree of this polynomial grew like q^{s-1} leading to the large bound mentioned above. If the degree of the relation can be made an absolute constant independent of q, then we can attain the goal of constant list size.[1] This motivates the concrete question: Does there exist a $\gamma \in \mathbb{F}_q^*$ such that $f(\gamma X)$ and $f(X)$, viewed as elements (via a natural injective map) of some extension field $\tilde{\mathbb{F}}$ over $\mathbb{F}_q$, satisfy $P(f(X), f(\gamma X)) = 0$ for a nonzero polynomial $P(Y,Z) \in \tilde{\mathbb{F}}[Y,Z]$ of degree an absolute constant? In [37], it is argued that if $P(Y,Z)$ has degree 1 in Z, i.e., has the form $Z - p(Y)$, then $p(Y)$ must have degree at least q in Y. Therefore, such a polynomial must be nonlinear in both Y, Z.

Another potential avenue for improving the complexity and list size is via a generalization of folded RS codes to folded *algebraic–geometric* (AG) codes. In [36], the authors define *correlated AG codes*, and describe list decoding algorithms for those codes, based on a generalization of the Parvaresh–Vardy approach to the general class of algebraic–geometric codes (of which RS codes are a special case). However, to relate folded AG codes to correlated AG codes like we did for RS codes requires bijections on the set of rational points of the underlying algebraic curve that have some special, hard to guarantee, property. This step seems like an highly intricate algebraic task, and especially so in the interesting asymptotic setting of a family of asymptotically good AG codes over a fixed alphabet.

8.2 Achieving capacity over small alphabets

One of the daunting challenges in the subject is to construct explicit *binary* codes along with polynomial time decoding algorithms that achieve the (binary) list decoding capacity. The question remains wide open over any fixed alphabet size.

Perhaps a somewhat less daunting challenge is to explicitly achieve list decoding capacity of binary (or q-ary for small q) codes for the *erasure* channel. In the erasure model, some codeword symbols are not received, and the rest are received without error, and the receiver knows

[1] This was the case in the work of Parvaresh and Vardy [62], since they had complete flexibility in picking the degree d (it just had to be larger than some absolute constant) and they defined the correlated polynomial to be $g(X) = f(X)^d \mod h(X)$ (instead of $f(\gamma X)$).

the locations corresponding to the symbols received. With rate R, the existential results indicate that one can list decode from a fraction $1 - R$ of erasures, but the best constructive results are far from this bound. For linear codes, the erasure decoding problem is easy algorithmically, so the question becomes a mainly combinatorial one. We refer the reader to [25] for more context on list decoding under erasures.

8.3 Applications outside coding theory

The motivation of the above questions on achieving list decoding capacity is primarily a coding-theoretic one. Resolving these questions may not directly impact any of the applications of list decoding outside coding theory, for example, the various complexity-theoretic applications. However, codes are by now a central combinatorial object in the toolkit of computer science, and they are intertwined with many other fundamental "pseudorandom" objects such as expander graphs and extractors. Therefore, any new idea to construct substantially better codes could potentially have broader impact. As an example of this phenomenon, we mention the recent work [44], which proved that the Parvaresh–Vardy codes yield excellent *condensers* that achieve near-optimal compression of a weak-random source while preserving essentially all of its entropy. The compressed source has a high enough entropy rate to enable easy extraction of almost all its randomness using previously known methods. Together, this yields simple and self-contained randomness extractors that are optimal up to constant factors, matching the previously best known construction due to [57] (that was quite complicated and built upon on a long line of previous work).

Therefore, the algebraic ideas underlying the recent developments in achieving list decoding capacity have already found powerful applications in areas outside coding theory. In the other direction, pseudorandom constructs have already yielded many new developments in coding theory, such as expander based codes [1, 31, 33, 66, 68], extractor codes [26, 72], and codes from the XOR lemma [47, 73]. It is our hope that exploring these connections in greater depth might lead to further interesting progress in coding theory; for example, it is interesting to see if insights from some exciting recent combinatorial techniques, such as the zig–zag graph product used to construct expanders, can be used to develop novel methods to construct codes.

8.4 Combinatorial questions

Several fundamental combinatorial aspects of list decoding are still not well-understood. For example, to get within ε of list decoding capacity (say for binary codes), what is the list size needed (as a function of ε)? Can one show that it is $\Theta(1/\varepsilon)$, matching the simple random coding argument of Section 3.2? More generally, techniques to lower bound the required list size are of interest; see [45] for recent work showing a $\Omega(1/\gamma^2)$ lower bound for list decoding a fraction $(1 - 1/q - \gamma)$ of errors with q-ary codes.

The large discrepancy between the strength of random coding results for linear and general codes, as outlined in Remark 3, merits further investigation.

The following seems to another intricate combinatorial task: Construct an explicit code and an explicit center with exponentially many codewords in a Hamming ball of normalized radius less than the relative distance around it. In addition to the intrinsic interest in such "bad list decoding configurations," this has potential applications in derandomizing the hardness results for approximating the minimum distance of a linear code [12].

8.5 Decoding concatenated codes using soft information

The approach to construct binary concatenated codes list decodable up to the Zyablov bound uses the list recovering property of the outer folded RS code. The inner decoder returns a list of symbols for each position of the outer folded RS code, but there is no weighting of the various symbols in such a list. Associating such weights and passing soft information that can be exploited by the outer decoder (Section 4.4.2) is a promising approach for improved decoding of concatenated codes. An approach based on convex optimization [49, 50] has been proposed to find the best choice of weights to use in concatenated decoding schemes. The involved computations become quickly infeasible for general, large inner codes, though experimentation with small inner codes having well-understood coset weight distributions is possible. Analytically, however, it remains a big challenge to understand good choices of weights and establish provable bounds on performance of such decoders. Some analytic work, for rather special or tailor-made inner codes, appears in [41, 42].

8.6 New vistas

Much of the success in constructing list decodable codes with good rate has hinged on algebraic ideas. In particular, most results depend directly or indirectly on the list decoding algorithm for Reed–Solomon codes or its variants. We now discuss other vistas in which one might look for codes that depart from the algebraic paradigm. Several connections between list-decodable codes and pseudorandomness have emerged in recent years, and in particular a basic tool in the latter topic called the XOR lemma can be used to construct list-decodable codes [73]. However, the rate of such codes have so far been sub-constant in the block length.

In Chapter 5, we discussed a combinatorial construction of list-decodable codes based on expander graphs due to [33]. These codes have positive rate and can be list decoded from a $(1 - \alpha)$ fraction of errors, for any desired design parameter $\alpha > 0$, by a linear time combinatorial algorithm. The rate of these codes, while positive, is rather small, and not competitive with the algebraic constructions. It remains to be seen if this can be improved with more sophisticated ideas. Some limited progress was made in [34] for the case of erasures, and list recovering with zero error (when all the input lists are guaranteed to contain the respective codeword symbol).

Progress on list decoding of new families of graph-based codes such as low-density parity-check (LDPC) codes would be very exciting. Such codes have been the source of substantial progress in getting close to capacity with very efficient algorithms for basic stochastic channels such as the binary erasure channel, the AWGN channel, and the binary symmetric channel (for basic details and pointers, see, for instance, the survey [22]). Whether algorithmic tools can be developed to list decode LDPC or related codes remains an intriguing question.

Acknowledgments

The author thanks Madhu Sudan and the anonymous reviewer for a careful reading of the manuscript and for their very useful comments on improving the quality and coverage of the presentation.

References

[1] N. Alon, J. Bruck, J. Naor, M. Naor, and R. Roth, "Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs," *IEEE Transactions on Information Theory*, vol. 38, pp. 509–516, 1992.

[2] N. Alon and F. R. K. Chung, "Explicit construction of linear sized tolerant networks," *Discrete Mathematics*, vol. 72, pp. 15–19, 1988.

[3] N. Alon and M. Luby, "A linear time erasure-resilient code with nearly optimal recovery," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1732–1736, 1996.

[4] N. Alon and J. Spencer, *The Probabilistic Method*. John Wiley and Sons, Inc., 1992.

[5] S. Ar, R. Lipton, R. Rubinfeld, and M. Sudan, "Reconstructing algebraic functions from mixed data," *SIAM Journal on Computing*, vol. 28, no. 2, pp. 488–511, 1999.

[6] E. Berlekamp, "Factoring polynomials over large finite fields," *Mathematics of Computation*, vol. 24, pp. 713–735, 1970.

[7] D. Bleichenbacher, A. Kiayias, and M. Yung, "Decoding of interleaved Reed Solomon codes over noisy data," in *Proceedings of the 30th International Colloquium on Automata, Languages and Programming*, pp. 97–108, 2003.

[8] V. M. Blinovsky, "Bounds for codes in the case of list decoding of finite volume," *Problems of Information Transmission*, vol. 22, no. 1, pp. 7–19, 1986.

[9] V. M. Blinovsky, "Code bounds for multiple packings over a nonbinary finite alphabet," *Problems of Information Transmission*, vol. 41, no. 1, pp. 23–32, 2005.

[10] D. Boneh, “Finding smooth integers in short intervals using CRT decoding,” in *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pp. 265–272, 2000.

[11] D. Coppersmith and M. Sudan, “Reconstructing curves in three (and higher) dimensional spaces from noisy data,” in *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pp. 136–142, June 2003.

[12] I. Dumer, D. Micciancio, and M. Sudan, “Hardness of approximating the minimum distance of a linear code,” *IEEE Transactions on Information Theory*, vol. 49, no. 1, pp. 22–37, 2003.

[13] P. Elias, “List decoding for noisy channels,” Technical Report 335, Research Laboratory of Electronics, MIT, 1957.

[14] P. Elias, “Error-correcting codes for list decoding,” *IEEE Transactions on Information Theory*, vol. 37, pp. 5–12, 1991.

[15] G. D. Forney, *Concatenated Codes*. MIT Press, Cambridge, MA, 1966.

[16] P. Gemmell and M. Sudan, “Highly resilient correctors for multivariate polynomials,” *Information Processing Letters*, vol. 43, no. 4, pp. 169–174, 1992.

[17] O. Goldreich and L. Levin, “A hard-core predicate for all one-way functions,” in *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pp. 25–32, May 1989.

[18] O. Goldreich, D. Ron, and M. Sudan, “Chinese remaindering with errors,” *IEEE Transactions on Information Theory*, vol. 46, no. 5, pp. 1330–1338, July 2000.

[19] O. Goldreich, R. Rubinfeld, and M. Sudan, “Learning polynomials with queries: The highly noisy case,” in *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science*, pp. 294–303, 1995.

[20] O. Goldreich, R. Rubinfeld, and M. Sudan, “Learning polynomials with queries: The highly noisy case,” *SIAM Journal on Discrete Mathematics*, vol. 13, no. 4, pp. 535–570, November 2000.

[21] P. Gopalan, R. Lipton, and Y. Ding, “Error correction against computationally bounded adversaries,” *Theory of Computing Systems*, to appear.

[22] V. Guruswami, “Iterative decoding of low-density parity check codes (A Survey),” *CoRR*, cs.IT/0610022, 2006. Appears in Issue 90 of the *Bulletin of the EATCS*.

[23] V. Guruswami, “List decoding with side information,” in *Proceedings of the 18th IEEE Conference on Computational Complexity (CCC)*, pp. 300–309, July 2003.

[24] V. Guruswami, “Limits to list decodability of linear codes,” in *Proceedings of the 34th ACM Symposium on Theory of Computing*, pp. 802–811, 2002.

[25] V. Guruswami, “List decoding from erasures: Bounds and code constructions,” *IEEE Transactions on Information Theory*, vol. 49, no. 11, pp. 2826–2833, 2003.

[26] V. Guruswami, “Better extractors for better codes?,” in *Proceedings of 36th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 436–444, June 2004.

[27] V. Guruswami, “Error-correcting codes and expander graphs,” *SIGACT News*, pp. 25–41, September 2004.

[28] V. Guruswami, "List decoding of error-correcting codes," *Lecture Notes in Computer Science*, no. 3282, Springer, 2004.

[29] V. Guruswami, "List decoding in pseudorandomness and average-case complexity," in *IEEE Information Theory Workshop*, March 2006.

[30] V. Guruswami, J. Hastad, M. Sudan, and D. Zuckerman, "Combinatorial bounds for list decoding," *IEEE Transactions on Information Theory*, vol. 48, no. 5, pp. 1021–1035, 2002.

[31] V. Guruswami and P. Indyk, "Expander-based constructions of efficiently decodable codes," in *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pp. 658–667, 2001.

[32] V. Guruswami and P. Indyk, "Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets," in *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 812–821, 2002.

[33] V. Guruswami and P. Indyk, "Linear-time encodable and list decodable codes," in *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 126–135, June 2003.

[34] V. Guruswami and P. Indyk, "Linear-time list decoding in error-free settings," in *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP)*, pp. 695–707, July 2004.

[35] V. Guruswami and P. Indyk, "Linear-time encodable/decodable codes with near-optimal rate," *IEEE Transactions on Information Theory*, vol. 51, no. 10, pp. 3393–3400, October 2005.

[36] V. Guruswami and A. Patthak, "Correlated algebraic-geometric codes: Improved list decoding over bounded alphabets," in *Proceedings of the 47th IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 227–238, October 2006. Journal version to appear in *Mathematics of Computation*.

[37] V. Guruswami and A. Rudra, "Explicit capacity-achieving list-decodable codes," in *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pp. 1–10, May 2006.

[38] V. Guruswami and A. Rudra, "Limits to list decoding Reed-Solomon codes," *IEEE Transactions on Information Theory*, vol. 52, no. 8, August 2006.

[39] V. Guruswami, A. Sahai, and M. Sudan, "Soft-decision decoding of Chinese remainder codes," in *Proceedings of the 41st IEEE Symposium on Foundations of Computer Science*, pp. 159–168, 2000.

[40] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometric codes," *IEEE Transactions on Information Theory*, vol. 45, pp. 1757–1767, 1999.

[41] V. Guruswami and M. Sudan, "List decoding algorithms for certain concatenated codes," in *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 181–190, 2000.

[42] V. Guruswami and M. Sudan, "Decoding concatenated codes using soft information," in *Proceedings of the 17th Annual IEEE Conference on Computational Complexity (CCC)*, pp. 148–157, 2002.

[43] V. Guruswami and M. Sudan, "On representations of algebraic-geometric codes," *IEEE Transactions on Information Theory*, vol. 47, no. 4, pp. 1610–1613, May 2001.

[44] V. Guruswami, C. Umans, and S. Vadhan, "Extractors and condensers from univariate polynomials," *Electronic Colloquium on Computational Complexity,* Report TR06-134, October 2006.

[45] V. Guruswami and S. Vadhan, "A lower bound on list size for list decoding," in *Proceedings of the 9th International Workshop on Randomization and Computation (RANDOM)*, pp. 318–329, 2005.

[46] R. W. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal*, vol. 29, pp. 147–160, April 1950.

[47] R. Impagliazzo, R. Jaiswal, and V. Kabanets, "Approximately list-decoding direct product codes and uniform hardness amplification," in *Proceedings of the 47th IEEE Symposium on Foundations of Computer Science*, October 2006.

[48] K. Jain and R. Venkatesan, "Efficient code construction via cryptographic assumptions," in *Proceedings of the 41st Annual Allerton Conference on Communication, Control, and Computing*, 2003.

[49] R. Koetter, "On optimal weight assignments for multivariate interpolation list-decoding," in *IEEE Information Theory Workshop*, March 2006.

[50] R. Koetter and A. Vardy, "Soft decoding of Reed Solomon codes and optimal weight assignments," in *ITG Fachtagung*, January 2002. Berlin, Germany.

[51] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Transactions on Information Theory*, vol. 49, no. 11, pp. 2809–2825, 2003.

[52] V. Y. Krachkovsky, "Reed-Solomon codes for correcting phased error bursts," *IEEE Transactions on Information Theory*, vol. 49, no. 11, pp. 2975–2984, November 2003.

[53] M. Langberg, "Private codes or succinct random codes that are (almost) Perfect," in *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, pp. 325–334, 2004.

[54] A. Lapidoth and P. Narayan, "Reliable communication under channel uncertainty," *IEEE Transactions on Information Theory*, vol. 44, no. 6, October 1998.

[55] R. Lidl and H. Niederreiter, *Introduction to Finite Fields and Their Applications.* Cambridge University Press, Cambridge, MA, 1986.

[56] R. J. Lipton, "A new approach to information theory," in *Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pp. 699–708, 1994.

[57] C.-J. Lu, O. Reingold, S. P. Vadhan, and A. Wigderson, "Extractors: Optimal up to constant factors," in *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pp. 602–611, 2003.

[58] R. J. McEliece, "On the average list size for the Guruswami-Sudan decoder," in *7th International Symposium on Communications Theory and Applications (ISCTA)*, July 2003.

[59] R. J. McEliece and L. Swanson, “On the decoder error probability for Reed-Solomon codes,” *IEEE Transactions on Information Theory*, vol. 32, no. 5, pp. 701–703, 1986.

[60] S. Micali, C. Peikert, M. Sudan, and D. A. Wilson, “Optimal error correction against computationally bounded noise,” in *Proceedings of the 2nd Theory of Cryptography Conference (TCC)*, pp. 1–16, 2005.

[61] F. Parvaresh and A. Vardy, “Multivariate interpolation decoding beyond the Guruswami-Sudan radius,” in *Proceedings of the 42nd Allerton Conference on Communication, Control and Computing*, 2004.

[62] F. Parvaresh and A. Vardy, “Correcting errors beyond the Guruswami-Sudan radius in polynomial time,” in *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pp. 285–294, 2005.

[63] W. W. Peterson, “Encoding and error-correction procedures for Bose-Chaudhuri codes,” *IEEE Transactions on Information Theory*, vol. 6, pp. 459–470, 1960.

[64] J. Radhakrishnan, “Proof of q-ary Johnson bound,” 2006. Personal Communication.

[65] C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.

[66] M. Sipser and D. Spielman, “Expander codes,” *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1710–1722, 1996.

[67] A. Smith, “Scrambling adversarial errors using few random bits, optimal information reconciliation, and better private codes,” Cryptology ePrint Archive, Report 2006/020, http://eprint.iacr.org/, 2006.

[68] D. Spielman, “Linear-time encodable and decodable error-correcting codes,” *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1723–1732, 1996.

[69] M. Sudan, “Decoding of Reed-Solomon codes beyond the error-correction bound,” *Journal of Complexity*, vol. 13, no. 1, pp. 180–193, 1997.

[70] M. Sudan, “List decoding: Algorithms and applications,” *SIGACT News*, vol. 31, pp. 16–27, 2000.

[71] M. Sudan, “Ideal error-correcting codes: Unifying algebraic and number-theoretic algorithms,” in *Proceedings of AAECC-14: The 14th Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pp. 36–45, November 2001.

[72] A. Ta-Shma and D. Zuckerman, “Extractor codes,” *IEEE Transactions on Information Theory*, vol. 50, no. 12, pp. 3015–3025, 2004.

[73] L. Trevisan, “List-decoding using the XOR lemma,” in *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science*, pp. 126–135, 2003.

[74] L. Trevisan, “Some applications of coding theory in computational complexity,” *Quaderni di Matematica*, vol. 13, pp. 347–424, 2004.

[75] J. H. van Lint, “Introduction to coding theory,” *Graduate Texts in Mathematics*, vol. 86, 3rd Edition, Springer-Verlag, Berlin, 1999.

[76] J. von zur Gathen, *Modern Computer Algebra*. Cambridge University Press, 1999.

[77] L. R. Welch and E. R. Berlekamp, "Error correction of algebraic block codes," US Patent Number 4,633,470, December 1986.

[78] J. M. Wozencraft, "List decoding," *Quarterly Progress Report, Research Laboratory of Electronics, MIT*, vol. 48, pp. 90–95, 1958.

[79] V. V. Zyablov and M. S. Pinsker, "List cascade decoding," *Problems of Information Transmission*, vol. 17, no. 4, pp. 29–34, (in Russian); pp. 236–240 (in English), 1982, 1981.