Explicit Codes Achieving List Decoding Capacity: Error-correction with Optimal Redundancy[*]

VENKATESAN GURUSWAMI[1†] ATRI RUDRA[2‡]

[1] Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195
venkat@cs.washington.edu

[2] Department of Computer Science and Engineering
State University of New York at Buffalo
Buffalo, NY 14260
atri@cse.buffalo.edu

Abstract

We present error-correcting codes that achieve the information-theoretically best possible trade-off between the rate and error-correction radius. Specifically, for every $0 < R < 1$ and $\varepsilon > 0$, we present an explicit construction of error-correcting codes of rate R that can be list decoded in polynomial time up to a fraction $(1-R-\varepsilon)$ of *worst-case* errors. At least theoretically, this meets one of the central challenges in algorithmic coding theory.

Our codes are simple to describe: they are *folded Reed-Solomon codes*, which are in fact *exactly* Reed-Solomon (RS) codes, but viewed as a code over a larger alphabet by careful bundling of codeword symbols. Given the ubiquity of RS codes, this is an appealing feature of our result, and in fact our methods directly yield better decoding algorithms for RS codes when errors occur in *phased bursts*.

The alphabet size of these folded RS codes is polynomial in the block length. We are able to reduce this to a constant (depending on ε) using ideas concerning "list recovery" and expander-based codes from [11, 12]. Concatenating the folded RS codes with suitable inner codes also gives us polynomial time constructible binary codes that can be efficiently list decoded up to the Zyablov bound, i.e., up to twice the radius achieved by the standard GMD decoding of concatenated codes.

[*]This version will appear in *IEEE Transactions on Information Theory*, Vol. 54, No. 1, January 2008. A preliminary version of this paper [14] appears in the *Proceedings of 38th Annual ACM Symposium on Theory of Computing* under the title "Explicit capacity-achieving list-decodable codes."

[†]Research supported by NSF Career award CCF-0343672, an Alfred P. Sloan Research Fellowship, and a David and Lucile Packard Foundation Fellowship.

[‡]Research supported by NSF CCF-0343672. This work was done when the author was at University of Washington.

1 Introduction

1.1 Background on List Decoding

Error-correcting codes enable reliable communication of messages over a noisy channel by cleverly introducing redundancy into the message to encode it into a codeword, which is then transmitted on the channel. This is accompanied by a decoding procedure that recovers the correct message even when several symbols in the transmitted codeword are corrupted. In this work, we focus on the adversarial or worst-case model of errors — we do not assume anything about how the errors and error locations are distributed beyond an upper bound on the total number of errors that may be caused. The central trade-off in this theory is the one between the amount of redundancy needed and the fraction of errors that can be corrected. The redundancy is measured by the *rate* of the code, which is the ratio of the the number of information symbols in the message to that in the codeword — thus, for a code with encoding function $E : \Sigma^k \to \Sigma^n$, the rate equals k/n. The *block length* of the code equals n, and Σ is its *alphabet*.

The goal in decoding is to find, given a noisy received word, the actual codeword that it could have possibly resulted from. If we target correcting a fraction ρ of errors (ρ will be called the error-correction radius or decoding radius), then this amounts to finding codewords within (normalized Hamming) distance ρ from the received word. We are guaranteed that there will be a unique such codeword provided *every* two distinct codewords differ on at least a fraction 2ρ of positions, or in other words the relative distance of the code is at least 2ρ. However, since the relative distance δ of a code must satisfy $\delta \leqslant 1 - R$ where R is the rate of the code (by the Singleton bound), the best trade-off between ρ and R that unique decoding permits is $\rho = \rho_U(R) = (1 - R)/2$. But this is an overly pessimistic estimate of the error-correction radius, since the way Hamming spheres pack in space, for *most* choices of the received word there will be at most one codeword within distance ρ from it even for ρ much greater than $\delta/2$. Therefore, *always* insisting on a unique answer will preclude decoding most such received words owing to a few pathological received words that have more than one codeword within distance roughly $\delta/2$ from them.

A notion called list decoding provides a clean way to get around this predicament, and yet deal with worst-case error patterns. Under list decoding, the decoder is required to output a list of all codewords within distance ρ from the received word. The notion of list decoding itself is quite old and dates back to work in 1950's by Elias [4] and Wozencraft [29]. However, the algorithmic aspects of list decoding were not revived until the more recent works [6, 27] which studied the problem for complexity-theoretic motivations.

Let us call a code C (ρ, L)-*list decodable* if the number of codewords within distance ρ of any received word is at most L. To obtain better trade-offs via list decoding, we need (ρ, L)-list decodable codes where L is bounded by a polynomial function of the block length, since this is an *a priori* requirement for polynomial time list decoding. How large can ρ be as a function of R for which such (ρ, L)-list decodable codes exist? A standard random coding argument shows that we can have $\rho \geqslant 1 - R - o(1)$ over large enough alphabets, cf. [30, 5], and a simple counting argument shows that ρ can be at most $1 - R$. Therefore the *list decoding capacity*, i.e., the information-theoretic limit of list decodability, is given by the trade-off $\rho_{\text{cap}}(R) = 1 - R = 2\rho_U(R)$. Thus list decoding holds the promise of correcting *twice* as many errors as unique decoding, for *every* rate.

We note that since the original message $\mathcal{M}$ has Rn symbols, it is information-theoretically impossible to perform the decoding if at most a fraction $(R - \varepsilon)$ of the received symbols agree with the encoding of $\mathcal{M}$ (for some $\varepsilon > 0$). This holds even for the erasure channel, and even if we are told in advance which symbols will be erased! Therefore, for any given rate, list decoding allows one to decode up to the largest fraction of errors that one can meaningfully hope to correct.

The above-mentioned list decodable codes are, however, non-constructive. In order to realize the potential of list decoding, one needs explicit constructions of such codes, and on top of that, polynomial time algorithms to perform list decoding. After essentially no progress in this direction in over 30 years, the work of Sudan [27] and improvements to it in [17], achieved efficient list decoding up to $\rho_{\mathrm{GS}}(R) = 1 - \sqrt{R}$ errors for an important family of codes called Reed-Solomon codes. Note that $1 - \sqrt{R} > \rho_U(R) = (1 - R)/2$ for every rate R, $0 < R < 1$, so this result showed that list decoding can be effectively used to go beyond the unique decoding radius for every rate (see Figure 1). The ratio $\rho_{\mathrm{GS}}(R)/\rho_U(R)$ approaches 2 for rates $R \to 0$, enabling error-correction when the fraction of errors approaches 100%, a feature that has found numerous applications outside coding theory, see for example [28], [8, Chap. 12].

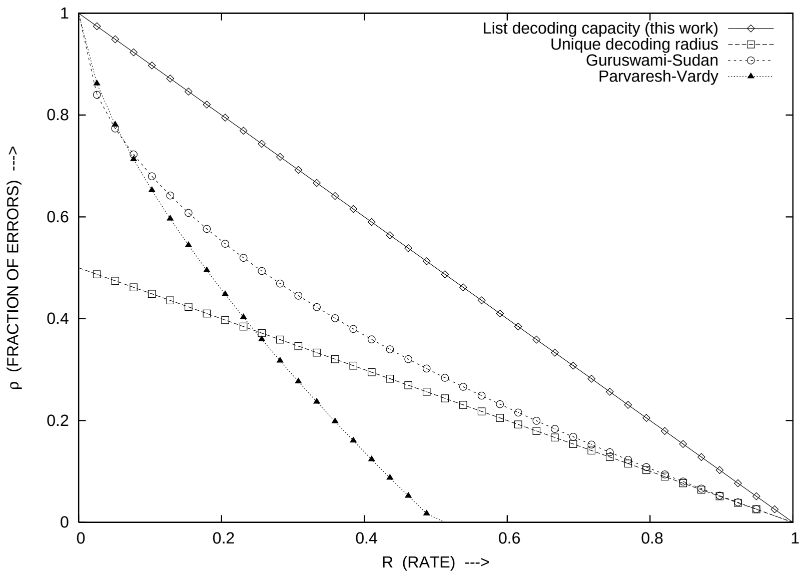

Figure 1: Error-correction radius ρ plotted against the rate R of the code for known algorithms. The best possible trade-off, i.e., capacity, is $\rho = 1 - R$, and our work achieves this.

Unfortunately, the improvement provided by [17] over unique decoding diminishes for larger rates, which is actually the regime of greater practical interest. For rates $R \to 1$, the ratio $\frac{\rho_{\mathrm{GS}}(R)}{\rho_U(R)}$ approaches 1, and already for rate $R = 1/2$ the ratio is at most 1.18. Thus, while the results of [27, 17] demonstrated that list decoding always, for every rate, enables correcting more errors than unique decoding, they fell short of realizing the full quantitative potential of list decoding.

The bound $\rho_{\mathrm{GS}}(R)$ stood as the best known error-correction radius for efficient list decoding for several years. In fact constructing (ρ, L)-list decodable codes of rate R for $\rho > \rho_{\mathrm{GS}}(R)$ and polynomially bounded L, regardless of the complexity of actually performing list decoding to radius ρ, itself was elusive. Some of this difficulty was due to the fact that $1 - \sqrt{R}$ is the largest radius for which small list size can be shown generically, via the so-called Johnson bound to argue about the number of codewords in Hamming balls using only information on the relative distance of the code, cf. [7].

In a recent breakthrough paper [25], Parvaresh and Vardy presented codes that are list-decodable beyond the $1-\sqrt{R}$ radius for low rates R. The codes they suggest are variants of Reed-Solomon (RS) codes obtained by evaluating $m \geqslant 1$ correlated polynomials at elements of the underlying field (with $m = 1$ giving RS codes). For any $m \geqslant 1$, they achieve the error-correction radius $\rho_{\mathrm{PV}}^{(m)}(R) = 1 - \sqrt[m+1]{m^m R^m}$. For rates $R \to 0$, choosing m large enough, they can list decode up to radius $1 - O(R\log(1/R))$, which approaches the capacity $1-R$. However, for $R \geqslant 1/16$, the best choice of m (the one that maximizes $\rho_{\mathrm{PV}}^{(m)}(R)$) is in fact $m = 1$, which reverts back to RS codes and the error-correction radius $1-\sqrt{R}$. (See Figure 1 where the bound $1-\sqrt[3]{4R^2}$ for the case $m = 2$ is plotted — except for very low rates, it gives a small improvement over $\rho_{\mathrm{GS}}(R)$.) Thus, getting arbitrarily close to capacity for some rate, as well as beating the $1-\sqrt{R}$ bound for every rate, both remained open[1].

1.2 Our Results

In this paper, we describe codes that get arbitrarily close to the list decoding capacity $\rho_{\mathrm{cap}}(R)$ for every rate. In other words, we give explicit codes of rate R together with polynomial time list decoding up to a fraction $1-R-\varepsilon$ of errors for every rate R and arbitrary $\varepsilon > 0$. As remarked before, this attains the information-theoretically best possible trade-off one can hope for between the rate and error-correction radius. While the focus of our presentation is primarily on the major asymptotic improvements we obtain over previous methods, we stress that our results offers a complexity vs. performance trade-of and gives non-trivial improvements, even for large rates and modest block lengths, with a value of the "folding parameter" m as small as 4. A discussion of the bounds for small values of m appears in Section 3.4.

Our codes are simple to describe: they are *folded Reed-Solomon codes*, which are in fact *exactly* Reed-Solomon (RS) codes, but viewed as a code over a larger alphabet by careful bundling of codeword symbols. Given the ubiquity of RS codes, this is an appealing feature of our result, and in fact our methods directly yield better decoding algorithms for RS codes when errors occur in *phased bursts* (a model considered in [22]).

Our result extends easily to the problem of *list recovery* (see Definition 5.1). The biggest advantage here is that we are able to achieve a rate that is independent of the size of the input lists. This is an extremely useful feature in concatenated code constructions. We are able to use this to reduce the alphabet size needed to achieve capacity, and also obtain results for binary codes. We briefly describe these results below.

To get within ε of capacity, the folded RS codes that we construct have alphabet size $n^{O(1/\varepsilon)}$ where n is the block length. By concatenating our codes of rate close to 1 (that are list recoverable) with suitable inner codes followed by redistribution of symbols using an expander graph (similar to a construction for linear-time unique decodable codes in [12]), we can get within ε of capacity with codes over an alphabet of size $2^{O(\varepsilon^{-4}\log(1/\varepsilon))}$. A counting argument shows that codes that can be list decoded efficiently to within ε of the capacity need to have an alphabet size of $2^{\Omega(1/\varepsilon)}$, so the alphabet size we attain is in the same ballpark as the best possible.

For binary codes, the list decoding capacity is known to be $\rho_{\mathrm{bin}}(R) = H^{-1}(1-R)$ where $H(\cdot)$ denotes the binary entropy function [5, 10]. We do not know explicit constructions of binary codes that approach this capacity. However, using our codes in a natural concatenation scheme, we give

[1]Independent of our work, Alex Vardy (personal communication) constructed a variant of the code defined in [25] which could be list decoded with fraction of errors more than $1-\sqrt{R}$ for all rates R. However, his construction gives only a small improvement over the $1-\sqrt{R}$ bound and does not achieve the list decoding capacity.

polynomial time constructible binary codes of rate R that can be list decoded up to a fraction $\rho_{\text{Zyab}}(R)$ of errors, where $\rho_{\text{Zyab}}(R)$ is the "Zyablov bound". See Figure 2 for a plot of these bounds.

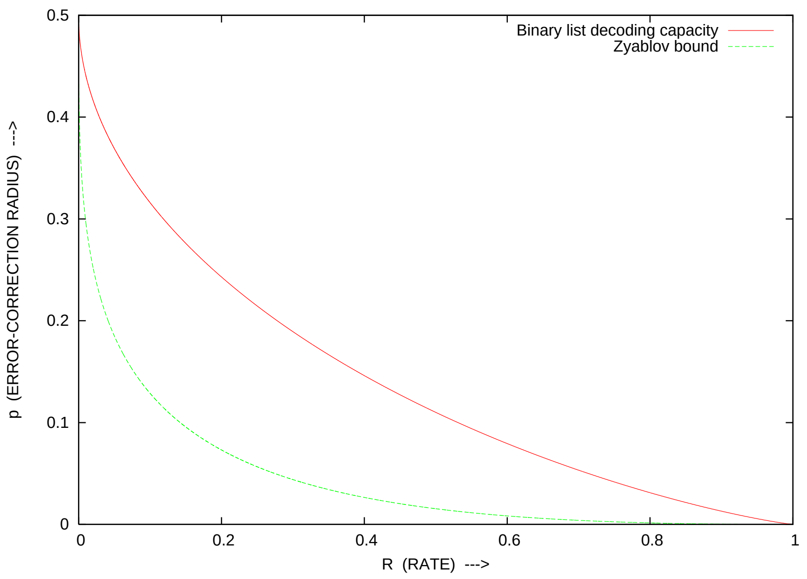

Figure 2: Error-correction radius ρ of our algorithm for binary codes plotted against the rate R. The best possible trade-off, i.e., capacity, is $\rho = H^{-1}(1 - R)$, and is also plotted.

1.3 Bibliographic Remarks

These results were first reported in [14]. We would like to point out that the presentation in this paper is somewhat different from the original papers [25, 14] in terms of technical details, organization, as well as chronology. With the benefit of hindsight, we believe this alternate presentation to be simpler and more self-contained direct than the description in [14], which used the results of Parvaresh-Vardy as a black-box. The exact relationship of our codes to the Parvaresh-Vardy construction is spelled out in detail in Section 2.3. Below, we discuss some technical aspects of the original development of this material, in order to shed light on the origins of our work. We also point the reader to the survey [9] for a detailed treatment of recent advances in algorithms for list decoding.

Two independent works by Coppersmith and Sudan [3] and Bleichenbacher, Kiayias and Yung [2] considered the variant of RS codes where the message consists of two (or more) independent polynomials over some field $\mathbb{F}$, and the encoding consists of the joint evaluation of these polynomials at elements of $\mathbb{F}$ (so this defines a code over $\mathbb{F}^2$).[2] 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

[2]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.

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) grows, the radius approaches the optimal value $1-R$. Since these are large alphabet codes, this model of random errors is not interesting from a coding-theoretic perspective, [3]though the algorithms are interesting from an algebraic viewpoint.

In [24], 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 degree k 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 [24] was to find several interpolation polynomials, but there was no guarantee that they were not all algebraically dependent.)

Then, in [25], 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 $g(X) = f(X)^d \mod E(X)$ for some integer d and an irreducible polynomial $E(X)$ of degree $k+1$.

Then, once we have the interpolation polynomial $Q(X,Y,Z)$, $f(X)$ can be obtained as follows: Reduce the coefficients of $Q(X,Y,Z)$ modulo $E(X)$ to get a polynomial $T(Y,Z)$ with coefficients from $\mathbb{F}[X]/(E(X))$ and then find roots of the univariate polynomial $T(Y,Y^d)$. This was the key idea in [25] 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 to using independent polynomials, however, does not come for free. 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 behind our work is 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.

1.4 Organization

We begin with a description of our code construction, folded Reed-Solomon codes, and outline their relation to Parvaresh-Vardy codes in Section 2. In Section 3, we present and analyze a trivariate interpolation based decoder for folded RS codes, which lets us approach a decoding radius of $1-R^{2/3}$

[3]This is because, as pointed out by Piotr Indyk, over large alphabets one can reduce decoding from uniformly random errors to decoding from erasures with a negligible loss in rate. The idea is to pad each codeword symbol with a small trail of 0's; a uniformly random error is highly unlikely to keep each of these 0's intact, and can thus be detected and declared as an erasure. Now recall that decoding from a fraction $1-R$ of erasures with rate R is easy using Reed-Solomon codes.

with rate R. In Section 4, we extend the approach to $(s+1)$-variate interpolation for any $s \geqslant 3$, allowing us to decode up to radius $1 - R^{s/(s+1)}$, and by picking s large enough obtain our main result (Theorem 4.4) on explicit codes achieving list decoding capacity. In Section 5, we generalize our decoding algorithm to the list recovery setting with almost no loss in rate, and use this powerful primitive to reduce the alphabet size of our capacity-achieving codes to a constant depending only on distance to capacity as well as to construct binary codes list-decodable up to the Zyablov bound. Finally, we close with some remarks in Section 6.

2 Folded Reed-Solomon Codes

In this section, we will use a simple variant of Reed-Solomon codes called folded Reed-Solomon codes for which we can beat the $1 - \sqrt{R}$ decoding radius possible for RS codes. In fact, by choosing parameters suitably, we can decode close to the optimal fraction $1 - R$ of errors with rate R.

2.1 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.

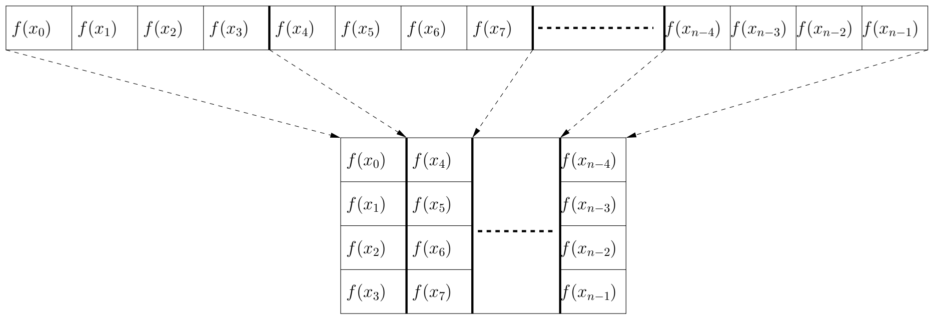

Figure 3: Folding of the Reed Solomon code with parameter $m = 4$.

Let $m \geqslant 1$ be an integer parameter called the *folding parameter*. Define $n \leqslant n'$ to be the largest integer that is divisible by m. Let C be the $[n, k]_{\mathbb{F}}$ RS code that is defined by the set of evaluation points $1, \gamma, \gamma^2, \ldots, \gamma^{n-1}$. In other words, C is obtained from C' by truncating the last $n' - n$ symbols. Note that m divides n.

Definition 2.1 (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 $\mathbb{F}^m$, where $n \leqslant |\mathbb{F}| - 1$ is the largest integer that is divisible by m. The encoding of a message $f(X)$, a polynomial over $\mathbb{F}$ of degree at most k, has as its j'th symbol, for $0 \leqslant j < n/m$, the m-tuple $(f(\gamma^{jm}), f(\gamma^{jm+1}), \cdots, f(\gamma^{jm+m-1}))$. In other words, the codewords of $\mathsf{FRS}_{\mathbb{F},\gamma,m,k}$ 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 3. 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 4 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 $1 - R$.

Remark 2.1 (Origins of term "folded RS codes")**.** The terminology of folded RS codes was coined in [22], where an algorithm to correct random errors in such codes was presented (for a noise model similar to the one used in [3, 2] 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).

2.2 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 to the bound for RS codes.

Consider the folded RS code 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. It is not too hard to see that correcting mT errors, where the errors occur in T contiguous blocks involves far few error patterns than correcting mT errors that can be arbitrarily distributed. As a concrete example, 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 that corrupts every 4'th 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 become clear once we describe our algorithms later in the paper.

We note that above "simplification" of the channel is not attained for free since the alphabet size increases after the folding operation[4]. 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, existing alphabet reduction techniques, which are used in Section 5.3, can handle polynomially

[4]However, we note that most of the operations in decoding still take place in the original field.

large alphabets, so this does not pose a big problem. Moreover, the benefits of our results kick in already for very small values of m (see Section 3.4).

2.3 Relation to Parvaresh Vardy codes

In this subsection, we relate folded RS codes to the Parvaresh-Vardy (PV) codes [25], which among other things will help make the ideas presented in the previous subsection more concrete.

The basic idea in the PV codes is to encode a polynomial f by the evaluations of $s \geqslant 2$ polynomials $f_0 = f, f_1, \ldots, f_{s-1}$ where $f_i(X) = f_{i-1}(X)^d \mod E(X)$ for an appropriate power d (and some irreducible polynomial $E(X)$) — let us call s the order of such a code. Our first main idea is to pick the irreducible polynomial $E(X)$ (and the parameter d) in such a manner that every polynomial f of degree at most k satisfies the following identity: $f(\gamma X) = f(X)^d \mod E(X)$, where γ is the generator of the underlying field. Thus, a folded RS code with bundling using an γ as above is in fact exactly the PV code of order $s = m$ for the set of evaluation points $\{1, \gamma^m, \gamma^{2m}, \ldots, \gamma^{(n/m-1)m}\}$. This is nice as it shows that PV codes can meet the Singleton bound (since folded RS codes do), but as such does not lead to any better codes for list decoding.

Here comes our second main idea. Let us compare the folded RS code to a PV code of order 2 (instead of order m) for the set of evaluation points $\{1, \gamma, \ldots \gamma^{m-2}, \gamma^m, \ldots, \gamma^{n-m}, \ldots, \gamma_{n-2}\}$. We find that in the PV encoding of f, for every $0 \leqslant i \leqslant n/m - 1$ and every $0 < j < m - 1$, $f(\gamma^{mi+j})$ appears exactly twice (once as $f(\gamma^{mi+j})$ and another time as $f_1(\gamma^{-1}\gamma^{mi+j})$), whereas it appears only once in the folded RS encoding. (See Figure 4 for an example when $m = 4$ and $s = 2$.) In

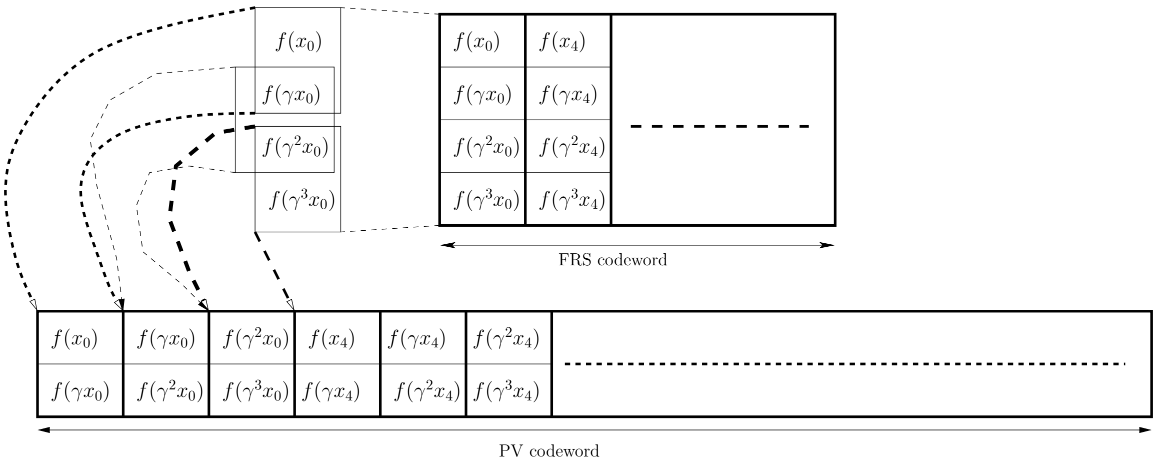

Figure 4: The correspondence between a folded Reed-Solomon code (with $m = 4$ and $x_i = \gamma^i$) and the Parvaresh Vardy code (of order $s = 2$) evaluated over $\{1, \gamma, \gamma^2, \gamma^4, \ldots, \gamma^{n-4}, \ldots, \gamma^{n-2}\}$. The correspondence for the first block in the folded RS codeword and the first three blocks in the PV codeword is shown explicitly in the left corner of the figure.

other words, the PV and folded RS codes have the same information, but the rate of the folded RS codes is bigger by a factor of $\frac{2m-2}{m} = 2 - \frac{2}{m}$. Decoding the folded RS codes from a fraction ρ of errors reduces to correcting the same fraction ρ of errors for the PV code. But the rate vs. error-correction radius trade-off is better for the folded RS code since it has (for large enough m, almost) twice the rate of the PV code.

In other words, our folded RS codes are chosen such that they are "compressed" forms of suitable PV codes, and thus have better rate than the corresponding PV code for a similar error-correction performance. This is where our gain is, and using this idea we are able to construct folded RS codes of rate R that are list decodable up to radius roughly $1 - \sqrt[s+1]{R^s}$ for any $s \geqslant 1$. Picking s large enough lets us get within any desired ε from capacity.

3 Trivariate interpolation based decoding

The list decoding algorithm for RS codes from [27, 17] is based on bivariate interpolation. The key factor driving the agreement parameter t needed for the decoding to be successful was the ($(1, k)$-weighted) degree D of the interpolated bivariate polynomial. Our quest for an improved algorithm for folded RS codes 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 $\mathbb{F}^3$. This enables performing the interpolation with $D = O(\sqrt[3]{k^2 n})$, 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, this is a somewhat simplistic hope and additional ideas are needed to make this approach work. We now turn to the task of developing a trivariate interpolation based decoder and proving that it can indeed decode up to a fraction $1 - R^{2/3}$ of errors.

3.1 Facts about trivariate interpolation

We begin with some basic definitions and facts concerning trivariate polynomials.

Definition 3.1. *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 3.2 (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 3.1. *Let $\{(\alpha_i, y_{i1}, y_{i2})\}_{i=1}^{n_0}$ be an arbitrary set of n_0 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, $1 \leqslant i \leqslant n_0$. Let $f(X), g(X)$ be polynomials of degree at most k such that for at least $t > D/r$ values of i, we have $f(\alpha_i) = y_{i1}$* and *$g(\alpha_i) = y_{i2}$. Then, $Q(X, f(X), g(X)) \equiv 0$.*

Proof. If we define $R(X) = Q(X, f(X), g(X))$, then $R(X)$ is a univariate polynomial of degree at most D. Now, for every i for which $f(\alpha_i) = y_{i1}$ *and* $g(\alpha_i) = y_{i2}$, $(X - \alpha_i)^r$ divides $R(X)$ (this follows from the definition of what it means for Q to have a zero of multiplicity r at $(\alpha_i, f(\alpha_i), g(\alpha_i))$). Therefore if $rt > D$, then $R(X)$ has more roots (counting multiplicities) than its degree, and so it must be the zero polynomial. □

Lemma 3.2. *Given an arbitrary set of n_0 triples $\{(\alpha_i, y_{i1}, y_{i2})\}_{i=1}^{n_0}$ 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 $i \in \{1, 2 \ldots, n_0\}$, provided*

$$\frac{D^3}{6k^2} > n_0 \binom{r+2}{3} . \tag{1}$$

Moreover, we can find such a $Q(X, Y_1, Y_2)$ in time polynomial in n_0, r by solving a system of homogeneous linear equations over $\mathbb{F}$.

Proof. 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 that 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 3-dimensional region $\{x + ky_1 + ky_2 \leqslant D \mid x, y_1, y_2 \geqslant 0\} \subset \mathbb{R}^3$ [25]. An easy calculation shows that the latter volume equals $\frac{D^3}{6k^2}$. Hence, if $\frac{D^3}{6k^2} > n_0\binom{r+2}{3}$, then the number of unknowns exceeds the number of equations, and we are guaranteed a nonzero solution. (See Remark 3.1 for an accurate estimate of the number of monomials of $(1, k, k)$-weighted degree at most D, which sometimes leads to a better condition under which a polynomial Q with the stated property exists.) $\square$

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 $N = n/m$. (Throughout this section, we will use n to denote the block length of the "unfolded" RS code.) 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 $0 \leqslant l < m$.) Finally define I to be the set $\{0, 1, 2, \ldots, n-1\}\setminus\{m-1, 2m-1, \ldots, n-1\}$ and let $n_0 = |I|$. Note that $n_0 = (m-1)n/m$.

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, $i \in I$, *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_0 triples $(\gamma^i, y_i, y_{i+1}) \in \mathbb{F}^3$ for $i \in I$, 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_0 triples and employing Lemma 3.1, 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 3.1.

Lemma 3.3. *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 that has a zero of multiplicity r at (γ^i, y_i, y_{i+1}) for $i \in I$. 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 3.2 and 3.3 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

$$D = \lfloor \sqrt[3]{k^2 n_0 r(r+1)(r+2)} \rfloor + 1 \ . \tag{2}$$

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 $i \in I$. (Lemma 3.2 guarantees the feasibility of this step as well as its computability in time polynomial in r and n_0 (and hence, n).)

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 for now, we can already claim the following result concerning the error-correction performance of this strategy.

Theorem 3.4. *The algorithm* Trivariate-FRS-decoder *successfully list decodes the folded Reed-Solomon code* $\mathsf{FRS}_{\mathbb{F},\gamma,m,k}$ *up to a number of errors equal to* $\left(N - \left\lfloor N \sqrt[3]{\left(\frac{mk}{(m-1)n}\right)^2 \left(1+\frac{1}{r}\right)\left(1+\frac{2}{r}\right)} \right\rfloor - 2\right)$.

Proof. By Lemma 3.3, 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 (2), this condition is met for the choice $t = 1 + \lfloor \sqrt[3]{\frac{k^2 n_0}{(m-1)^3}\left(1+\frac{1}{r}\right)\left(1+\frac{2}{r}\right)} + \frac{1}{(m-1)r} \rfloor$. The number of errors is equal to $N - t$, and recalling that $n = mN$ and $n_0 = (m-1)n/m$, we get the claimed bound on the list decoding radius. $\square$

The rate of the folded Reed-Solomon code is $R = (k+1)/n > k/n$, and so the fraction of errors corrected (for large enough r) is $1 - \left(\frac{mR}{m-1}\right)^{2/3}$. Note that for $m = 2$, this is just the bound $1-(2R)^{2/3}$ that Parvaresh-Vardy obtained for decoding their codes using trivariate interpolation [25]. The bound becomes better for larger values of m, and letting the folding parameter m grow, we can approach a decoding radius of $1 - R^{2/3}$.

3.3 Root-finding step

In light of the above discussion in Section 3.2, 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 3.5. *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* $E(X) \stackrel{\text{def}}{=} X^{q-1} - \gamma$ *is irreducible over* $\mathbb{F}$.

2. *Every polynomial* $f(X) \in \mathbb{F}[X]$ *of degree less than* $q-1$ *satisfies* $f(\gamma X) = f(X)^q \mod E(X)$.

Proof. The fact that $E(X) = X^{q-1} - \gamma$ is irreducible over $\mathbb{F}_q$ follows from a known, precise characterization of all irreducible binomials, i.e., polynomials of the form $X^a - c$, see for instance [23, Chap. 3, Sec. 5]. For completeness, and since this is an easy special case, we now prove this fact. Suppose $E(X)$ 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 $E(X)$ 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 $f(X)^q \equiv f(\gamma X) \pmod{E(X)}$ which implies that $f(X)^q \mod E(X) = f(\gamma X)$ 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.

Theorem 3.6. *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 $E(X) = X^{q-1} - \gamma$. We know by Lemma 3.5 that $E(X)$ is irreducible. We first divide out the largest power of $E(X)$ that divides $Q(X, Y_1, Y_2)$ to obtain $Q_0(X, Y_1, Y_2)$ where $Q(X, Y_1, Y_2) = E(X)^b Q_0(X, Y_1, Y_2)$ for some $b \geqslant 0$ and $E(X)$ 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 $E(X)$ to get a polynomial $T(Y_1, Y_2)$ with coefficients from the extension field $\tilde{\mathbb{F}} \stackrel{\text{def}}{=} \mathbb{F}[X]/(E(X))$ (this is a field since $E(X)$ 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 $E(X)$.

In view of Lemma 3.5, it suffices to find degree $\leqslant k$ polynomials $f(X)$ satisfying $Q_0(X, f(X), f(X)^q) \pmod{E(X)} = 0$. 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 $R(Y_1) \stackrel{\text{def}}{=} T(Y_1, Y_1^q)$, 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 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 base field is at most $q \lg q$. Therefore, we can find all roots of $R(Y_1)$ by a deterministic algorithm running in time polynomial in d, q [1]. 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 of $f(X)$ 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 (2) 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 Theorem 3.4, and recalling that the running time is polynomial in n, r, we can conclude the following main result of this section.

Theorem 3.7. *For every $\varepsilon > 0$ and R, $0 < R < 1$, there is a family of m-folded Reed-Solomon codes for $m = O(1/\varepsilon)$ that 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 $1/\varepsilon$.*

3.4 Alternate decoding bound for high rates and practical considerations

In the discussion above, the fraction of errors $1-\left(\frac{mR}{m-1}\right)^{2/3}$, call it $\rho_{\mathrm{a}}^{(m,2)}(R)$, approaches $1-R^{2/3}$ (and hence improves upon the bound of $\rho_{\mathrm{GS}}(R) = 1-\sqrt{R}$ in [17]) for every rate R for *large enough* m. For practical implementations the parameter m will be some small fixed integer. Note that for fixed m, the bound of $\rho_{\mathrm{a}}^{(m,2)}(R)$ is useless for $R \geqslant 1-\frac{1}{m}$, whereas the $1-\sqrt{R}$ bound for decoding Reed-Solomon codes [17] is meaningful for all $R < 1$.

Given that one is often interested in high rate codes, this suggests that in order to reap the benefits of our new codes for large rates, the folding parameter needs to be picked large enough. Fortunately, this is not the case, and we now show that one can beat the $1-\sqrt{R}$ bound for all rates R for a *fixed* value of the folding parameter m; in fact, a value as small as $m = 5$ suffices. These bounds also hint at the fact that the improvements offered by the decoding algorithms in this paper are not just asymptotic and kick in for parameter choices that could be practical.

Our goal now is to sketch how a minor change to the algorithm in Section 3.2 allows us to correct a fraction

$$\rho_{\mathrm{b}}^{(m,2)}(R) = \frac{m}{m+1}\left(1-R^{2/3}\right) \tag{3}$$

of errors. The bound of $\rho_{\mathrm{b}}^{(m,2)}(R)$ gives a larger decoding radius than $\rho_{\mathrm{a}}^{(m,2)}(R)$ for large rates R. A more precise comparison of the bounds $\rho_{\mathrm{a}}^{(m,2)}, \rho_{\mathrm{b}}^{(m,2)}$ and ρ_{GS} is done at the end of this subsection. The improvement of the decoding radius to $\rho_{\mathrm{b}}^{(m,2)}(R)$ for large rates (and hence small error fractions) comes via another way to analyze (a variant of) the algorithm in Section 3.2, which was suggested to us by Jørn Justesen. The algorithm is the same as in Section 3.2 except that the set of interpolating points is slightly different. In particular in the trivariate interpolating step, we choose $I = \{0, 1, \ldots, n-2\}$. Let $n_0 = |I| = n-1$. The crucial observation here is that an erroneous symbol $z_j \in \mathbb{F}^m$ (for some position $0 \leqslant j < N$ in the received word $\mathbf{z}$) translates to at most $m+1$ errors among the interpolation tuples in the trivariate interpolation step. More precisely, given that $0 \leqslant e \leqslant N$ is the number of errors,

> For at least $t' = n_0 - e(m+1)$ values of i, $i \in I$, *both* the equalities $f(\gamma^i) = y_i$ and $f(\gamma^{i+1}) = y_{i+1}$ hold.

By Lemmas 3.1, 3.2 and the degree bound (2), the algorithm outlined above will work as long as

$$n_0 - e(m+1) > \sqrt[3]{k^2 n_0 \left(1+\frac{1}{r}\right)\left(1+\frac{2}{r}\right)} + \frac{1}{r}.$$

Recalling that $n_0 = n - 1 < n$, the above is satisfied if

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

Recalling that $n = Nm$, the above is satisfied if

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

Noting that $m \geqslant 1$, leads to the following analog of Theorem 3.4:

Theorem 3.8. *The version of algorithm* Trivariate-FRS-decoder *discussed above, successfully list decodes the folded Reed-Solomon code* $\mathsf{FRS}_{\mathbb{F},\gamma,m,k}$ *as long as the number of errors is less than* $\left\lfloor \left(\frac{m}{m+1}\right) N \left(1 - \sqrt[3]{\left(\frac{k}{n}\right)^2 \left(1 + \frac{1}{r}\right)\left(1 + \frac{2}{r}\right)}\right)\right\rfloor - 1$.

For large enough r, the above implies that rate R folded RS codes can be list decoded up to a fraction $\rho_{\mathrm{b}}^{(m,2)}(R) = \left(\frac{m}{m+1}\right)\left(1 - R^{2/3}\right)$ of errors.

Comparison of the bounds

We now make a comparison between the bounds $\rho_{\mathrm{b}}^{(m,2)}$, $\rho_{\mathrm{a}}^{(m,2)}$ and ρ_{GS}. We first note that $\rho_{\mathrm{b}}^{(m,2)}(R) \geqslant \rho_{\mathrm{a}}^{(m,2)}(R)$ for every rate $R \geqslant \left(1 - \frac{1}{m}\right)\left(\frac{1}{m+1-\sqrt[3]{m(m-1)^2}}\right)^{3/2}$. In particular, $\rho_{\mathrm{b}}^{(m,2)}(R) > \rho_{\mathrm{a}}^{(m,2)}(R)$ for all rates $R \geqslant 1 - \frac{1}{m}$. Let us now compare $\rho_{\mathrm{b}}^{(m,2)}(R)$ and $\rho_{\mathrm{GS}}(R)$. Specifically, we give a heuristic argument to show that for high enough rates R and $m \geqslant 4$, $\rho_{\mathrm{b}}^{(m,2)}(R) > \rho_{\mathrm{GS}}(R)$. Let $R = 1 - \varepsilon$. Then ignoring the $O(\varepsilon^2)$ terms in the Taylor expansions we get $\rho_{\mathrm{GS}}(1-\varepsilon) \approx \varepsilon/2$ and $\rho_{\mathrm{b}}^{(m,2)}(1-\varepsilon) \approx \frac{2m\varepsilon}{3(m+1)}$: the latter quantity is strictly larger than the former for every $m \geqslant 4$. In fact, it can be verified that for all rates $R \geqslant 0.44$, $\rho_{\mathrm{b}}^{(4,2)} > \rho_{\mathrm{GS}}$. Figure 5 plots the tradeoff $\rho_{\mathrm{GS}}(R)$ and $\max\left(\rho_{\mathrm{b}}^{(m,2)}(R), \rho_{\mathrm{a}}^{(m,2)}(R)\right)$ for some small values of $m \geqslant 2$. The limit for large m, which is $1 - R^{2/3}$, is also plotted.

Remark 3.1 (Better bound on $(1,k,k)$-weighted degree)**.** For small values of the parameter r, one should use a better estimate for the degree bound D than the bound (1) based on the volume argument. The number of monomials $X^i Y_1^{j_1} Y_2^{j_2}$ whose $(1,k,k)$-weighted degree is at most D is exactly equal to

$$k\binom{a+2}{3} + (D - ak + 1)\binom{a+2}{2} \tag{4}$$

where $a = \left\lfloor \frac{D}{k} \right\rfloor$. This is often larger than the $\frac{D^3}{6k^2}$ lower bound we used in Lemma 3.2, and certainly for any specific setting of parameters n, k, r, the estimate (4) should be used. A similar remark applies for the bound used in Lemma 4.1 for $(s+1)$-variate interpolation. Since it makes no difference for the asymptotics, we chose to stick with the simpler expressions.

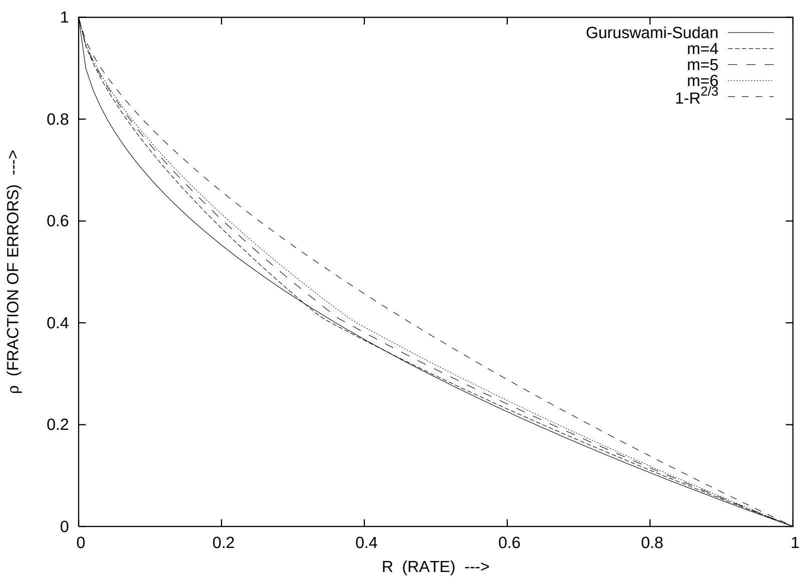

Figure 5: Error-correction radius $\max(\rho_{\rm b}^{(m,2)}(R), \rho_{\rm a}^{(m,2)}(R))$ for $m = 4, 5$. For comparison $\rho_{\rm GS}(R) = 1 - \sqrt{R}$ and the limit $1 - R^{2/3}$ are also plotted. For $m = 5$, the performance of the trivariate interpolation algorithm strictly improves upon that of $\rho_{\rm GS}$ for all rates.

4 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 3.1 and 3.2.

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. Define the set of interpolations points to be

$$I = \{0, 1, 2, \ldots, n-1\} \setminus \left(\bigcup_{j=0}^{n/m-1} \{jm + m - s + 1, jm + m - s + 2, \ldots, jm + m - 1\} \right).$$

The reason for this choice of I is that if the m-tuple containing y_i is correct and $i \in I$, then all the s values $y_i, y_{i+1}, \ldots, y_{i+s-1}$ are correct.

Define $n_0 = |I|$. Note that $n_0 = n(m - s + 1)/m$. Following algorithm $\mathsf{Trivariate\text{-}FRS\text{-}decoder}$, 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 $i \in I$.

The following is a straightforward generalization of Lemmas 3.2 and 3.3.

Lemma 4.1. *1. Provided $\frac{D^{s+1}}{(s+1)!k^s} > n_0\binom{r+s}{s+1}$, 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 $X^a Y_1^{b_1} \cdots Y_s^{b_s}$ 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 3.3. 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 3.1, we conclude that $R(X) \overset{\text{def}}{=} Q(X, f(X), f(\gamma X), \ldots, f(\gamma^{s-1}X))$ 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

$$D = \left\lfloor (k^s n_0 r(r+1)\cdots(r+s))^{1/(s+1)} \right\rfloor + 1 \ . \tag{5}$$

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 Theorem 3.6. First, by dividing out by $E(X)$ enough times, we can assume that not all coefficients of $Q(X, Y_1, \ldots, Y_s)$, viewed as a polynomial in $Y_1, \ldots, Y_s$ with coefficients in $\mathbb{F}[X]$, are divisible by $E(X)$. We can then go modulo $E(X)$ to get a nonzero polynomial $T(Y_1, Y_2, \ldots, Y_s)$ over the extension field $\tilde{\mathbb{F}} = \mathbb{F}[X]/(E(X))$. Now, by Lemma 3.5, we have $f(\gamma^j X) = f(X)^{q^j} \mod E(X)$ 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 $R(Y_1) = T(Y_1, Y_1^q, \ldots, Y_1^{q^{s-1}})$. 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 (5) is satisfied if

$$t \geqslant \frac{(k^s n_0)^{1/(s+1)}}{m-s+1} \left(\prod_{j=1}^{s} \left(1 + \frac{j}{r}\right) \right)^{1/(s+1)} + 2 \ .$$

The above along with the fact that $n_0 = N(m-s+1)$ implies the following, which is multivariate generalization of Theorem 3.4.

Theorem 4.2. *For every integer $m \geqslant 1$ and every s, $1 \leqslant s \leqslant m$, the $(s+1)$-variate FRS decoder successfully list decodes the m-folded Reed-Solomon code $\mathsf{FRS}_{\mathbb{F},\gamma,m,k}$ up to a radius $N - t$ as long as the agreement parameter t satisfies*

$$t \geqslant \sqrt[s+1]{\left(N\frac{k}{m-s+1}\right)^s \prod_{j=1}^{s}\left(1+\frac{j}{r}\right)} + 2 \ . \tag{6}$$

The algorithm runs in $n^{O(s)}$ time and outputs a list of size at most $|F|^s = n^{O(s)}$.

Recalling that the block length of $\mathsf{FRS}_{\mathbb{F},\gamma,m,k}$ is $N = n/m$ and the rate is $(k+1)/n$, the above algorithm can decode a fraction of errors approaching

$$1 - \sqrt[s+1]{\left(\frac{mR}{m-s+1}\right)^s \prod_{j=1}^{s}\left(1+\frac{j}{r}\right)} \tag{7}$$

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

Theorem 4.3. *For every $0 < \delta \leqslant 1$, integer $s \geqslant 1$ and $0 < R < 1$, there is a family of m-folded Reed-Solomon codes for $m = O(s/\delta)$ that have rate at least R and which can be list decoded up to a fraction $1-(1+\delta)R^{s/(s+1)}$ of errors in time $(Nm)^{O(s)}$ and outputs a list of size at most $(Nm)^{O(s)}$ where N is the block length of the code. The alphabet size of the code as a function of the block length N is $(Nm)^{O(m)}$.*

Proof. We first note that (7) is at least

$$1 - \left(1+\frac{s}{r}\right)\left(\frac{m}{m-s+1}\right) R^{s/(s+1)}. \tag{8}$$

We now instantiate the parameters r and m in terms of s and δ:

$$r = \frac{3s}{\delta} \qquad m = \frac{(s-1)(3+\delta)}{\delta} \ .$$

With the above choice, we have

$$\left(1+\frac{s}{r}\right)\frac{m}{m-s+1} = \left(1+\frac{\delta}{3}\right)^2 < 1+\delta \ .$$

Together with the bound (8) on the decoding radius, we conclude that the $(s+1)$-variate decoding algorithm certainly list decodes up to a fraction $1-(1+\delta)R^{s/(s+1)}$ of errors.

The worst case list size is q^s and the claim on the list size follows by recalling that $q \leqslant n+m$ and $N = n/m$. The alphabet size is $q^m = (Nm)^{O(m)}$. The running time has two major components: (1) Interpolating the $s+1$-variate polynomial $Q(\cdot)$, which by Lemma 4.1 is $(nr^s)^{O(1)}$; and (2) Finding all the roots of the interpolated polynomial, which takes $q^{O(s)}$ time. Of the two, the time complexity of the root finding step dominates, which is $(Nm)^{O(s)}$. ☐

In the limit of large s, the decoding radius approaches the list decoding capacity $1-R$, leading to our main result.

Theorem 4.4 (Explicit capacity-approaching codes)**.** *For every $\varepsilon > 0$ and $0 < R < 1$, there is a family of folded Reed-Solomon codes that have rate at least R and which can be list decoded up to a fraction $1 - R - \varepsilon$ of errors in time (and outputs a list of size at most) $(N/\varepsilon^2)^{O(\varepsilon^{-1}\log(1/R))}$ where N is the block length of the code. The alphabet size of the code as a function of the block length N is $(N/\varepsilon^2)^{O(1/\varepsilon^2)}$.*

Proof. Given ε, R, we will apply Theorem 4.3 with the choice

$$s = \left\lceil \frac{\log(1/R)}{\log(1+\varepsilon)} \right\rceil \quad \text{and} \quad \delta = \frac{\varepsilon(1-R)}{R(1+\varepsilon)} \ . \tag{9}$$

The list decoding radius guaranteed by Theorem 4.3 is at least

$$\begin{aligned} 1-(1+\delta)R^{s/(s+1)} &= 1 - R(1+\delta)(1/R)^{1/(s+1)} \\ &\geqslant 1 - R(1+\delta)(1+\varepsilon) \ \ \text{(by the choice of } s \text{ in (9))} \\ &= 1-(R+\varepsilon) \ \ \text{(using the value of } \delta) \ . \end{aligned}$$

We now turn our attention to the time complexity of the decoding algorithm and the alphabet size of the code. To this end we first claim that $m = O(1/\varepsilon^2)$. To see this note that by the definition of s and δ:

$$m = O\left(\frac{s}{\delta}\right) = O\left(s \cdot \frac{R(1+\varepsilon)}{\varepsilon(1-R)}\right) = O\left(\frac{1}{\varepsilon^2} \cdot \frac{R\ln(1/R)}{1-R}\right) = O(1/\varepsilon^2) \ ,$$

where for the last step we used $\ln(1/R) \leqslant \frac{1}{R} - 1$ for $0 < R \leqslant 1$. The claims on the running time, worst case list size and the alphabet size of the code follow from Theorem 4.3 and the facts that $m = O(1/\varepsilon^2)$ and $s = O(\varepsilon^{-1}\log(1/R))$. $\square$

With the proof of our main theoretical result (Theorem 4.4) completed, we close this section with a few remarks.

Remark 4.1 (Improvement to decoding radius for high rates)**.** As in Section 3.4, it is possible to improve the bound of (7) to

$$\max\left(\frac{m}{m+s-1}\left(1 - \sqrt[s+1]{R^s \prod_{j=1}^{s}\left(1+\frac{j}{s}\right)}\right), 1 - \sqrt[s+1]{\left(\frac{mR}{m-s+1}\right)^s \prod_{j=1}^{s}\left(1+\frac{j}{r}\right)}\right) \ .$$

The former bound is better for large rates.

Remark 4.2 (Optimality of degree q of relation between $f(X)$ and $f(\gamma X)$)**.** Let K be the extension field $\mathbb{F}_q[X]/(E(X))$ where $E(X) = X^{q-1} - \gamma$. The elements of K are in one-one correspondence with polynomials of degree less than $q-1$ over $\mathbb{F}_q$. The content of Lemma 3.5, which we made crucial use of above, is that the map $\Gamma : K \to K$ defined by $f(X) \mapsto f(\gamma X)$ is a degree q map over K, i.e., as a polynomial over K, $\Gamma(Z) = Z^q$. The fact that this degree is as large as q is in turn the cause for the large list size that we need for list decoding. It is natural to ask if a different map Γ' could have lower degree (perhaps over a different extension field K_1). Unfortunately, it turns out this is not possible, as argued below.

Indeed, let Γ' be a ring homomorphism of $\mathbb{F}_q[X]$ defined by $\Gamma'(f(X)) = f(G(X))$ for some polynomial G over $\mathbb{F}_q$. Let $E_1(X)$ be an irreducible polynomial over $\mathbb{F}_q$ of degree ℓ, and let $K_1 =$

$\mathbb{F}_q[X]/(E_1(X))$ be the associated extension field. We can view Γ' as a map Γ_1 on K_1 by identifying polynomials of degree less than ℓ with K_1 and defining $\Gamma_1(f(X)) = f(G(X)) \mod E_1(X)$. The key point is that Γ_1 is an $\mathbb{F}_q$-*linear* map on K_1. Expressed as a polynomial over K_1, Γ_1 must therefore be a *linearized polynomial*, [23, Chap. 3, Sec. 4], which has only terms with exponents that are powers of q (including $q^0 = 1$). It turns out that for our purposes Γ_1 cannot have degree 1, and so it must have degree at least q.

5 Extensions and Codes over Smaller Alphabets

5.1 Extension to list recovery

We now present a very useful generalization of the list decoding result of Theorem 4.4 to the setting of *list recovery.* Under the list recovery problem, one is given as input for each codeword position, not just one but a set of several, say l, alphabet symbols. The goal is to find and output all codewords which agree with some element of the input sets for several positions. Codes for which this more general problem can be solved turn out to be extremely valuable as outer codes in concatenated code constructions. In short, this is because one can pass a set of possibilities from decodings of the inner codes and then list recover the outer code with those sets as the input. If we only had a list-decodable code at the outer level, we will be forced to make a unique choice in decoding the inner codes thus losing valuable information.

Definition 5.1 (List Recovery)**.** *A code $C \subseteq \Sigma^n$ is said to be (ζ, l, L)-list recoverable if for every sequence of sets $S_1, \ldots, S_n$ where each $S_i \subseteq \Sigma$ has at most l elements, the number of codewords $c \in C$ for which $c_i \in S_i$ for at least ζn positions $i \in \{1, 2, \ldots, n\}$ is at most L.*

A code $C \subseteq \Sigma^n$ is said to (ζ, l)-list recoverable in polynomial time if it is $(\zeta, l, L(n))$-list recoverable for some polynomially bounded function $L(\cdot)$, and moreover there is a polynomial time algorithm to find the at most $L(n)$ codewords that are solutions to any $(\zeta, l, L(n))$-list recovery instance.

We remark that when $l = 1$, $(\zeta, 1, \cdot)$-list recovery is the same as list decoding up to a $(1 - \zeta)$ fraction of errors. List recovery has been implicitly studied in several works; the name itself was coined in [11].

Theorem 4.4 can be generalized to list recover the folded RS codes. Specifically, for a FRS code with parameters as in Section 4, for an arbitrary constant $l \geqslant 1$, we can (ζ, l)-list recover in polynomial time provided

$$\zeta N \geqslant \sqrt[s+1]{\left(\frac{k}{m-s+1}\right)^s \frac{nl}{m} \prod_{j=1}^{s} \left(1 + \frac{j}{r}\right)} + 2 \; . \tag{10}$$

where $N = n/m$. We briefly justify this claim. The generalization of the list decoding algorithm of Section 4 is straightforward: instead of one interpolation condition for each symbol of the received word, we just impose $|S_i| \leqslant l$ many interpolation conditions for each position $i \in \{1, 2, \ldots, n\}$ (where S_i is the i'th input set in the list recovery instance). The number of interpolation conditions is at most nl, and so replacing n by nl in the bound of Lemma 4.1 guarantees successful decoding. This in turn implies that the condition on the number of agreement of (6) generalizes to the one in (10). This straightforward generalization to list recovery is a positive feature of all interpolation based decoding algorithms [27, 17, 25] beginning with the one due to Sudan [27].

Picking $r \gg s$ and $m \gg s$ in (10), we get (ζ, l)-list recover with rate R for $\zeta \geqslant \left(lR^s\right)^{1/(s+1)}$. Now comes the remarkable fact: we can pick a suitable $s \gg l$ and perform (ζ, l)-list recovery with agreement parameter $\zeta \geqslant R + \varepsilon$ which is independent of l! We state the formal result below (Theorem 4.4 is a special case when $l = 1$).

Theorem 5.1. *For every integer $l \geqslant 1$, for all R, $0 < R < 1$ and $\varepsilon > 0$, and for every prime p, there is an* explicit *family of folded Reed-Solomon codes over fields of characteristic p that have rate at least R and which can be $(R+\varepsilon, l)$-list recovered in polynomial time. The alphabet size of a code of block length N in the family is $(N/\varepsilon^2)^{O(\varepsilon^{-2}\log l/(1-R))}$.*

Proof. (Sketch) Using the exact same arguments as in the proof of Theorem 4.3 to the agreement condition of (10), we get that one can list recover in polynomial time as long as $\zeta \geqslant (1+\delta)(lR^s)^{1/(s+1)}$, for any $\delta > 0$. The arguments to obtains a lower bound of $R + \varepsilon$ are similar to the ones employed in the proof of theorem 4.4. However, s needs to be defined in a slightly different manner:

$$s = \left\lceil \frac{\log(l/R)}{\log(1+\varepsilon)} \right\rceil .$$

Also this implies that $m = O\left(\frac{\log l}{(1-R)\varepsilon^2}\right)$, which implies the claimed bound on the alphabet size of the code. □

Remark 5.1 (Soft Decoding)**.** The decoding algorithm for folded RS codes from Theorem 4.4 can be further generalized to handle soft information, where for each codeword position i the decoder is given as input a non-negative weight $w_{i,z}$ for each possible alphabet symbol z. The weights $w_{i,z}$ can be used to encode the confidence information concerning the likelihood of the the i'th symbol of the codeword being z [21]. For any $\varepsilon > 0$, for suitable choice of parameters, our codes of rate R over alphabet Σ have a soft decoding algorithm that outputs all codewords $c = \langle c_1, c_2, \ldots, c_N \rangle$ that satisfy

$$\sum_{i=1}^{N} w_{i,c_i} > \left((1+\varepsilon)(RN)^s \Big(\sum_{i=1}^{N} \sum_{z \in \Sigma} w_{i,z}^{s+1} \Big) \right)^{1/(s+1)} + \varepsilon N \max_{i,z} w_{i,z} .$$

For $s = 1$, this soft decoding condition is identical to the one for Reed-Solomon codes in [17].

5.2 Binary codes decodable up to Zyablov bound

Concatenating the folded RS codes with suitable inner codes also gives us polytime constructible binary codes that can be efficiently list decoded up to the Zyablov bound, i.e., up to twice the radius achieved by the standard GMD decoding of concatenated codes. The optimal list recoverability of the folded RS codes plays a crucial role in establishing such a result.

Theorem 5.2. *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 $R \cdot r$ which can be list decoded in polynomial time up to a fraction $(1-R)H^{-1}(1-r) - \varepsilon$ of errors.*

Proof. We will construct binary codes with the claimed property by concatenating two codes C_1 and C_2. For C_1, we will use a folded RS code over a field of characteristic 2 with block length n_1, rate at least R, and which can be $(R+\varepsilon, l)$-list recovered in polynomial time for $l = \lceil 10/\varepsilon \rceil$. Let the alphabet size of C_1 be 2^M where $M = O(\varepsilon^{-2}\log(1/\varepsilon)\log n_1)$. For C_2, we will use a binary linear code of dimension M 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 [10]. Also, a greedy construction of such a code by constructing its M basis elements in turn is presented in [10] and this process takes $2^{O(M)}$ time. We conclude that the necessary inner code can be constructed in $n_1^{O(\varepsilon^{-2}\log(1/\varepsilon))}$ time. The code C_1, being a folded RS code over a field of characteristic 2, is $\mathbb{F}_2$-linear, and therefore when concatenated with a binary linear inner code such as C_2, results in a binary linear code. The rate of the concatenated code is 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 set of at most l possible candidates for each outer codeword symbol. The outer code is then $(R+\varepsilon, l)$-list recovered using these sets, each of which has size at most l, 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 $(1-R-\varepsilon)\rho = (1-R-\varepsilon)H^{-1}(1-r-\varepsilon)$. Since $\varepsilon > 0$ was arbitrary, we get the claimed result. $\square$

Optimizing over the choice of inner and outer codes rates r, R in the above results, we can decode up to the Zyablov bound, see Figure 2.

Remark 5.2. In particular, decoding up to the Zyablov bound implies that we can correct a fraction $(1/2-\varepsilon)$ of errors with rate $\Omega(\varepsilon^3)$ for small $\varepsilon \to 0$, which is better than the rate of $\Omega(\varepsilon^3/\log(1/\varepsilon))$ achieved in [13]. However, our construction and decoding complexity are $n^{O(\varepsilon^{-2}\log(1/\varepsilon))}$ whereas these are at most $f(\varepsilon)n^c$ for an absolute constant c in [13]. Also, we bound the list size needed in the worst-case by $n^{O(\varepsilon^{-1}\log(1/\varepsilon))}$, while the list size needed in the construction in [13] is $(1/\varepsilon)^{O(\log\log(1/\varepsilon))}$.

Remark 5.3 (Decoding up to the Blokh-Zyablov bound)**.** In a follow-up paper, we use a similar approach extended to multilevel concatenation schemes together with inner codes that have good "nested" list-decodability properties, to construct binary codes list-decodable up to the *Blokh-Zyablov* bound [15].

5.3 Capacity-Achieving codes over smaller alphabets

Our result of Theorem 4.4 has two undesirable aspects: both the alphabet size and worst-case list size output by the list decoding algorithm are a polynomial of large degree in the block length. We now show that the alphabet size can be reduced to a constant that depends only on the distance ε to capacity.

Theorem 5.3. *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. The theorem is proved using the code construction scheme used in [12] for linear time unique decodable codes with optimal rate, with different components appropriate for list decoding plugged in. We briefly describe the main ideas behind the construction and proof below. The high level approach is to concatenate two codes C_{out} and C_{in}, and then redistribute the symbols of the resulting codeword using an expander graph (Figure 6 depicts this high level structure and should be useful in reading the following formal description). In the following, assume that $\varepsilon < 1/6$ and let $\delta = \varepsilon^2$.

The outer code C_{out} will be a code of rate $(1-2\varepsilon)$ over an alphabet Σ of size $n^{(1/\delta)^{O(1)}}$ that can be $(1-\varepsilon, O(1/\varepsilon))$-list recovered in polynomial time, as guaranteed by Theorem 5.1. That is, the rate of C_{out} will be close to 1, and it can be (ζ, l)-list recovered for large l and $\zeta \to 1$.

The inner code C_{in} will be a $((1-R-4\varepsilon), O(1/\varepsilon))$-list decodable code with near-optimal rate, say rate at least $(R+3\varepsilon)$. Such a code is guaranteed to exist over an alphabet of size $O(1/\varepsilon^2)$ using random coding arguments. A naive brute-force for such a code, however, is too expensive, since we need a code with $|\Sigma| = n^{\Omega(1)}$ codewords. Guruswami and Indyk [11], see also [8, Sec. 9.3], prove that there is a small (quasi-polynomial sized) sample space of *pseudolinear codes* in which most codes have the needed property. Furthermore, they also present a deterministic polynomial time construction of such a code (using derandomization techniques), see [8, Sec. 9.3.3].

The concatenation of C_{out} and C_{in} gives a code C_{concat} of rate at least $(1-2\varepsilon)(R+3\varepsilon) \geqslant R$ over an alphabet Σ of size $|\Sigma| = O(1/\varepsilon^2)$. Moreover, given a received word of the concatenated code, one can find all codewords that agree with the received word on a fraction $R+4\varepsilon$ of locations in at least $(1-\varepsilon)$ of the inner blocks. Indeed, we can do this by running the natural list decoding algorithm, call it $\mathcal{A}$, for C_{concat} that decodes each of the inner blocks to a radius of $(1-R-4\varepsilon)$ returning up to $l = O(1/\varepsilon)$ possibilities for each block, and then $(1-\varepsilon, l)$-list recovering C_{out}.

The last component in this construction is a $D = O(1/\varepsilon^4)$-regular bipartite expander graph which is used to redistribute symbols of the concatenated code in a manner so that an overall agreement on a fraction $R+7\varepsilon$ of the redistributed symbols implies a fractional agreement of at least $R+4\varepsilon$ on most (specifically a fraction $(1-\varepsilon)$) of the inner blocks of the concatenated code. In other words, the expander redistributes symbols in a manner that "smoothens" the distributions of errors evenly among the various inner blocks (except for possibly a ε fraction of the blocks). This expander based redistribution incurs no loss in rate, but increases the alphabet size to $O(1/\varepsilon^2)^{O(1/\varepsilon^4)} = 2^{O(\varepsilon^{-4}\log(1/\varepsilon))}$.

We now discuss some details of how the expander is used. Suppose that the block length of the folded RS code C_{out} is N_1 and that of C_{in} is N_2. Let us assume that N_2 is a multiple of D, say $N_2 = n_2 D$ (if this is not the case, we can make it so by padding at most $D-1$ dummy symbols at a negligible loss in rate). Therefore codewords of C_{in}, and therefore also of C_{concat}, can be thought of as being composed of blocks of D symbols each. Let $N = N_1 n_2$, so that codewords of C_{concat} can be viewed as elements in $(\Sigma^D)^N$.

Let $G = (L, R, E)$ be a D-regular bipartite graph with N vertices on each side (i.e., $|L| = |R| = N$), with the property that for every subset $Y \subseteq R$ of size at least $(R+7\varepsilon)N$, the number of vertices belonging to L that have at most $(R+6\varepsilon)D$ of their neighbors in Y is at most δN (for $\delta = \varepsilon^2$). It is a well-known fact (used also in [12]) that if G is picked to be the double cover of a Ramanujan expander of degree $D \geqslant 4/(\delta\varepsilon^2)$, then G will have such a property.

We now define our final code $C^* = G(C_{\text{concat}}) \subseteq (\Sigma^D)^N$ formally. The codewords in C^* are in one-one correspondence with those of C_{concat}. Given a codeword $c \in C_{\text{concat}}$, its ND symbols (each belonging to Σ) are placed on the ND edges of G, with the D symbols in its i'th block (belonging to Σ^D, as defined above) being placed on the D edges incident on the i'th vertex of L (in some fixed order). The codeword in C^* corresponding to c has as its i'th symbol the collection of D symbols (in some fixed order) on the D edges incident on the i'th vertex of R. See Figure 6 for a pictorial view of the construction.

Note that the rate of C^* is identical to that C_{concat}, and is thus at least R. Its alphabet size is

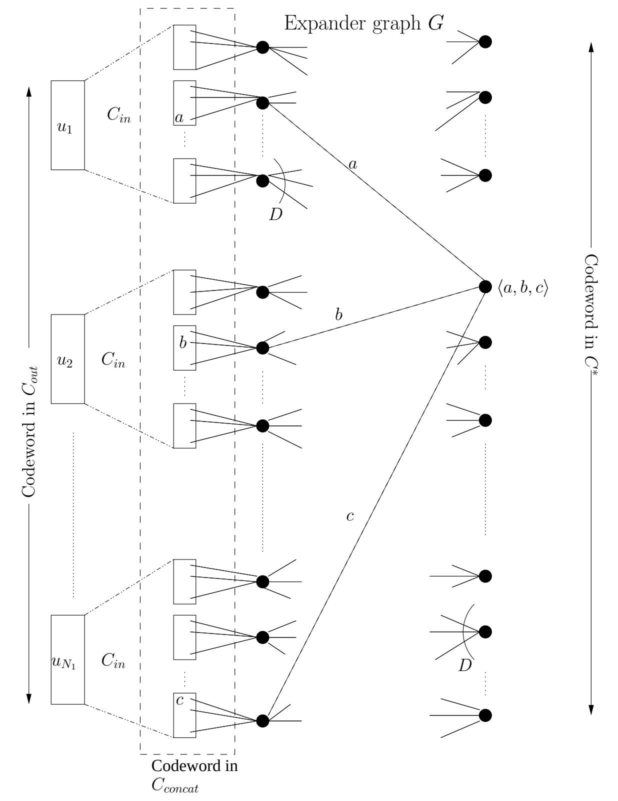

Figure 6: The code C^* used in the proof of Theorem 5.3. We start with a codeword $\langle u_1, \ldots, u_{N_1} \rangle$ in C_{out}. Then every symbol is encoded by C_{in} to form a codeword in C_{concat} (this intermediate codeword is marked by the dotted box). The symbols in the codeword for C_{concat} are divided into chunks of D symbols and then redistributed along the edges of an expander G of degree D. In the figure, we use $D = 3$ for clarity. Also the distribution of three symbols a, b and c (that form a symbol in the final codeword in C^*) is shown.

$|\Sigma|^D = O(1/\varepsilon^2)^{O(1/\varepsilon^4)} = 2^{O(\varepsilon^{-4}\log(1/\varepsilon))}$, as claimed. We will now argue how C^* can be list decoded up to a fraction $(1 - R - 7\varepsilon)$ of errors.

Given a received word $\mathbf{r} \in (\Sigma^D)^N$, the following is the natural algorithm to find all codewords of C^* with agreement at least $(R + 7\varepsilon)N$ with $\mathbf{r}$. Redistribute symbols according to the expander backwards to compute the received word $\mathbf{r}'$ for C_{concat} which would result in $\mathbf{r}$. Then run the earlier-mentioned decoding algorithm $\mathcal{A}$ on $\mathbf{r}'$.

We now briefly argue the correctness of this algorithm. Let $\mathbf{c} \in C^*$ be a codeword with agreement at least $(R + 7\varepsilon)N$ with $\mathbf{r}$. Let $\mathbf{c}'$ denote the codeword of C_{concat} that leads to $\mathbf{c}$ after symbol redistribution by G, and finally suppose $\mathbf{c}''$ is the codeword of C_{out} that yields $\mathbf{c}'$ upon concatenation by C_{in}. By the expansion properties of G, it follows that all but a δ fraction of N D-long blocks of $\mathbf{r}'$ have agreement at least $(R + 6\varepsilon)D$ with the corresponding blocks of $\mathbf{c}'$. By an averaging argument, this implies that at least a fraction $(1 - \sqrt{\delta})$ of the N_1 blocks of $\mathbf{c}'$ that correspond to codewords of C_{in} encoding the N_1 symbols of $\mathbf{c}''$, agree with at least a fraction $(1 - \sqrt{\delta})(R + 6\varepsilon) = (1 - \varepsilon)(R + 6\varepsilon) \geqslant R + 4\varepsilon$ of the symbols of the corresponding block of $\mathbf{r}'$. As argued earlier, this in turn implies that the decoding algorithm $\mathcal{A}$ for C_{concat} when run on input $\mathbf{r}'$ will output a polynomial size list that will include $\mathbf{c}'$. $\square$

6 Concluding Remarks

We close with some remarks and open questions. In the preliminary version [14] of this paper, we noted that the folded RS codes bear some resemblance to certain "randomness extractors" constructed in [26], and wondered if some of the techniques in this work and [25] could be used to construct simple extractors based on univariate polynomials. In a recent work [19], this has been answered in the affirmative in a fairly strong sense. It is shown in [19] that the Parvaresh-Vardy codes yield excellent "randomness condensers," which achieve near-optimal compression of a weak random source while preserving all its min-entropy, and in turn these lead to the best known randomness extractors (that are optimal up to constant factors).

We have solved the qualitative problem of achieving list decoding capacity over large alphabets. Our work could be improved with some respect to some parameters. The size of the list needed to perform list decoding to a radius that is within ε of capacity grows as $n^{O(1/\varepsilon)}$ where n is the block length of the code. It remains an open question to bring this list size down to a constant independent of n, or even to $f(\varepsilon)n^c$ with an exponent c independent of ε (we recall that the existential random coding arguments work with a list size of $O(1/\varepsilon)$). We managed to reduce the alphabet size needed to approach capacity to a constant independent of n. However, this involved a brute-force search for a rather large code. Obtaining a "direct" algebraic construction over a constant-sized alphabet (such as variants of algebraic-geometric (AG) codes) might help in addressing these two issues. To this end, Guruswami and Patthak [13] 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.

Finally, constructing binary codes (or q-ary codes for some fixed, small value of q) that approach the respective list decoding capacity remains a challenging open problem. In recent work [16], we show that there *exist* q-ary linear concatenated codes that achieve list decoding capacity (in the sense that every Hamming ball of radius $H_q^{-1}(1-R-\varepsilon)$ has polynomially many codewords, where R is the rate). In particular, this results holds when the outer code is a folded RS code. This is somewhat encouraging news since concatenation has been the preeminent method to construct good list-decodable codes over small alphabets. But realizing the full potential of concatenated codes and achieving capacity (or even substantially improving upon the Blokh-Zyablov bound) with explicit codes and polynomial time decoding remains a huge challenge. It seems likely that carefully chosen soft information to pass from the inner decodings to the outer algebraic decoder (see [20, 18] for examples of such decoders) may hold the key to further progress in list decoding concatenated codes.

Acknowledgments

We thank Jørn Justesen for suggesting the alternate bound on decoding radius (3) discussed in Section 3.4, and for kindly allowing us to include it in our presentation. We thank an anonymous referee for several useful comments and in particular for encouraging us to highlight the relevance of the results even for small values of the folding parameter; this was the impetus for the discussion in Section 3.4. We thank Piotr Indyk, Charanjit Jutla, Farzad Parvaresh, Anindya Patthak, Madhu

Sudan, and Alexander Vardy for useful discussions and comments.

References

[1] E. Berlekamp. Factoring polynomials over large finite fields. *Mathematics of Computation*, 24:713–735, 1970.

[2] D. Bleichenbacher, A. Kiayias, and M. Yung. Decoding interleaved Reed-Solomon codes over noisy channels. *Theoretical Computer Science*, 379(3):348–360, 2007.

[3] 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*, pages 136–142, June 2003.

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

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

[6] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 25–32, 1989.

[7] V. Guruswami. Limits to list decodability of linear codes. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 802–811, 2002.

[8] V. Guruswami. *List decoding of error-correcting codes.* Number 3282 in Lecture Notes in Computer Science. Springer, 2004.

[9] V. Guruswami. *Algorithmic Results in List Decoding*, volume 2 (Issue 2) of *Foundations and Trends in Theoretical Computer Science (FnT-TCS)*. NOW publishers, 2007.

[10] V. Guruswami, J. Håstad, M. Sudan, and D. Zuckerman. Combinatorial bounds for list decoding. *IEEE Transactions on Information Theory*, 48(5):1021–1035, 2002.

[11] 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*, pages 658–667, 2001.

[12] V. Guruswami and P. Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Transactions on Information Theory*, 51(10):3393–3400, October 2005.

[13] V. Guruswami and A. Patthak. Correlated Algebraic-Geometric codes: Improved list decoding over bounded alphabets. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 227–236, October 2006. Accepted to *Mathematics of Computation.*

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

[15] V. Guruswami and A. Rudra. Better binary list-decodable codes via multilevel concatenation. In *Proceedings of 11th International Workshop on Randomization and Computation*, pages 554–568, August 2007.

[16] V. Guruswami and A. Rudra. Concatenated codes can achieve list decoding capacity. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, January 2008. To appear.

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

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

[19] V. Guruswami, C. Umans, and S. P. Vadhan. Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes. In *Proceedings of the 22nd Annual IEEE Conference on Computational Complexity*, pages 96–108, 2007.

[20] R. Koetter. On optimal weight assignments for multivariate interpolation list-decoding. In *Proc. 2006 IEEE Information Theory Workshop*, pages 37–41, March 2006.

[21] R. Koetter and A. Vardy. Algebraic soft-decision decoding of Reed-Solomon codes. *IEEE Transactions on Information Theory*, 49(11):2809–2825, November 2003.

[22] V. Y. Krachkovsky. Reed-Solomon codes for correcting phased error bursts. *IEEE Transactions on Information Theory*, 49(11):2975–2984, November 2003.

[23] R. Lidl and H. Niederreiter. *Introduction to Finite Fields and their applications.* Cambridge University Press, Cambridge, MA, 1986.

[24] 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.

[25] 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*, pages 285–294, 2005.

[26] R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudo-random generator. *Journal of the ACM*, 52(2):172–216, 2005.

[27] M. Sudan. Decoding of Reed-Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997.

[28] M. Sudan. List decoding: Algorithms and applications. *SIGACT News*, 31:16–27, 2000.

[29] J. M. Wozencraft. List Decoding. *Quarterly Progress Report, Research Laboratory of Electronics, MIT*, 48:90–95, 1958.

[30] V. V. Zyablov and M. S. Pinsker. List cascade decoding. *Problems of Information Transmission*, 17(4):29–34, 1981 (in Russian); pp. 236-240 (in English), 1982.