Achieving List Decoding Capacity Using Folded Reed-Solomon Codes

Venkatesan Guruswami and Atri Rudra

***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 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 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*.

These results were first reported in [1]. The description in this paper, though, is different and the codes are based on a different, more flexible, version of folding. The algebraic argument underlying the decoding algorithm is also simpler, and leads to a slightly better bound on decoding complexity and worst-case list size.

I. Introduction

A. Background on List Decoding

Under the problem of *list decoding* an error-correcting code C of block length n up to a fraction p of errors, we are given a noisy received word $\mathbf{r}$, and the goal is to output all codewords of C within Hamming distance pn of $\mathbf{r}$. Even in a worst-case noise model, list decoding opens up the possibility of decoding from a number of errors exceeding the packing radius of the code (the idea being that for pathological noise patterns the decoder may output multiple answers).

In order for a family of codes to be efficiently decodable up to a fraction p of errors, we need the *a priori* guarantee that every Hamming ball of radius pn has few (at most a polynomial in the block length) codewords. A standard random coding argument shows that we can have $\rho \geqslant 1 - R - o(1)$ over large enough alphabets, cf. [2], [3], and a simple counting argument shows that ρ must 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 $\rho_{\rm cap}(R) = 1 - R$. Since the message has Rn correct symbols, one needs at least a fraction R of correct symbols in order to do meaningful error-recovery. (Indeed, this holds even for the weaker model of erasures, and even if the adversary announces which symbols it plans to erase!) In other words, list decoding enables the possibility of achieving the information-theoretically optimal trade-off between the rate of a code and the fraction of errors that can be corrected, even for an adversarial/worst-case noise model.

As is typical with capacity theorems in coding theory, the above-mentioned list decodable codes are 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 [4] and improvements to it in [5], achieved efficient list decoding up to $\rho_{\rm GS}(R) = 1 - \sqrt{R}$ errors for the important family of Reed-Solomon (RS) codes. The latter algorithm in [5] could also handle different weights on different coordinates by encoding them into an appropriate number of interpolation multiplicities, and in an important follow-up work, Koetter and Vardy [6] developed soft-decoding algorithms for Reed-Solomon codes that leverage this fact.

The $1 - \sqrt{R}$ bound stood as the best known error-correction radius for efficient list decoding for several years, and improving upon it emerged as a central open problem in the subject. In a recent breakthrough, Parvaresh and Vardy [7] 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 $\rho_{\rm PV}^{(m)}(R) = 1 - \sqrt[m+1]{m^m R^m}$. 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 $\rho_{\rm PV}^{(m)}(R)$) 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 small rates, it gives a very small improvement over the $1 - \sqrt{R}$ bound achieved for RS codes. Thus, getting arbitrarily close to capacity for some rate, as well as beating the $1 - \sqrt{R}$ bound for every rate, both remained open.

B. Our Result

In this paper, we describe codes that get arbitrarily close to the list decoding capacity $\rho_{\rm cap}(R)$ 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 $\varepsilon > 0$.

These results were first reported in [1]. We would like to point out that the presentation in this paper is somewhat different from the original papers [7], [1] in terms of technical

The authors are with the Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195.

V. Guruswami was supported in part by NSF CCF-0343672, a Sloan Research Fellowship, and a David and Lucile Packard Foundation Fellowship.

A. Rudra was supported in part by NSF CCF-0343672.

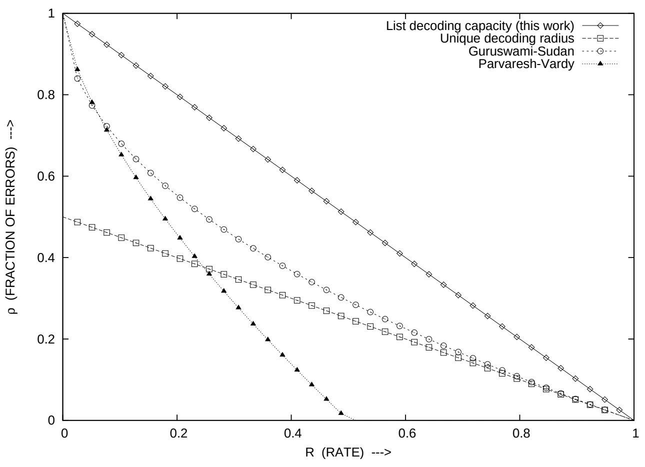

Fig. 1. Error-correction radius ρ plotted against the rate R of the code for known algorithms. The best possible trade-off, i.e., list decoding capacity, is $\rho = 1 - R$, and the codes and algorithms reported here achieve this. Note that this is twice the best fraction of errors correctable by unique decoding algorithms that only correct up to half-the-distance.

details, organization, as well as chronology. Our description closely follows that of an upcoming survey [8] where the interested reader may find further background and details on algebraic list decoding. With the benefit of hindsight, we believe this alternate presentation to be simpler than the description in [1].

Below, we discuss some technical aspects of the original development of this material, in order to shed light on the origins of our work.

C. Bibliographic Remarks

Two independent works by Coppersmith and Sudan [9] and Bleichenbacher, Kiayias and Yung [10] 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) 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.

[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.

In [11], 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 [11] was to find several interpolation polynomials, but there was no guarantee that they were not all algebraically dependent.)

Then, in [7], 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 [7] 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. 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 [12], where an algorithm to correct random errors in such codes was presented (for a noise model similar to the one used in [9], [10] 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).

II. 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.

A. 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 $1, \gamma, \gamma^2, \dots, \gamma^{n-1}$. 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 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 $\mathbb{F}^m$. 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 $C' = \mathsf{FRS}_{\mathbb{F},\gamma,m,k}$ 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.

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 $1 - R$.

B. 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. 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 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 be come 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. 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 from Section V can handle polynomially large alphabets, so this does not pose a big problem.

III. Trivariate interpolation based decoding

The list decoding algorithm for RS codes from [4], [5] 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 $D = O((k^2n)^{1/3})$, 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, 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.

A. 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 $\{(\alpha_i, y_{i1}, y_{i2})\}_{i=1}^n$ 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: 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 3.2: 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: 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 $\{x + ky_1 + ky_2 \leqslant D \mid x, y_1, y_2 \geqslant 0\} \subset \mathbb{R}^3$ [7]. 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. ■

B. 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 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 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 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

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

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

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

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 (1), this condition is met for the choice $t =$

$1 + \lfloor \sqrt[3]{\frac{k^2 n}{(m-1)^3}\left(1+\frac{1}{r}\right)\left(1+\frac{2}{r}\right)} + \frac{1}{(m-1)r} \rfloor$. 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 (for large enough r) is $1 - \frac{m}{m-1}R^{2/3}$. And letting the parameter m grow, we can approach a decoding radius of $1 - R^{2/3}$.

C. 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 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 $E(X) \overset{\text{def}}{=} X^{q-1} - \gamma$ 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 [13, 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.

Lemma 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 $\tilde{\mathbb{F}} \overset{\text{def}}{=} \mathbb{F}[X]/(E(X))$ (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 $R(Y_1) \overset{\text{def}}{=} T(Y_1, Y_1^q)$, 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 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 [14]. 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 (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 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 $1/\varepsilon$.

IV. 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. 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 3.2 and 3.3.

Lemma 4.1: 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 $Q(X, f(X), f(\gamma X), \ldots, f(\gamma^{s-1}X)) \equiv 0$.

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

$$D = \left\lfloor \left(k^s n r(r+1)\cdots(r+s)\right)^{1/(s+1)} \right\rfloor + 1 \ . \qquad (2)$$

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 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 (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

$$1 - \left(1 + \frac{s}{r}\right) \frac{m}{m-s+1} R^{s/(s+1)} \qquad (3)$$

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 $1 - (1+\varepsilon)R^{s/(s+1)}$ for any desired $\varepsilon > 0$. We state this result formally below.

Theorem 4.2: For every $\varepsilon > 0$, integer $s \geqslant 1$ and $0 < R < 1$, there is a family of m-folded Reed-Solomon codes for $m = O(s/\varepsilon)$ that 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 $(Nm)^{O(s)}(1/\varepsilon)^{O(1)}$ 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, the decoding radius approaches the list decoding capacity $1 - R$, leading to our main result.

Theorem 4.3 (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 $(N/\varepsilon)^{O(1/\varepsilon)}$ where N is the block length of the code. The alphabet size of the code as a function of the block length N is $(N/\varepsilon)^{O(1/\varepsilon^2)}$.

Remark 4.1: It is possible to slightly improve the bound of (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 not use only a fraction $(m-s+1)/m$ of the n $(s+1)$-tuples for interpolation. Specifically, we omit tuples with γ^i for $i \mod m > m-s$. This does not affect the number of $(s+1)$-tuples for which we have agreement (this remains at least $t(m-s+1)$), but the number of interpolation conditions is 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.

V. Achieving Capacity over Bounded Alphabets

The above capacity-achieving codes 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. Fortunately, using ideas concerning list-recovering and expander-based codes from [15], [16], together with the fact that above list decoding algorithm extends to a powerful list-recovering algorithm, we can reduce the alphabet size that is a constant depending only on the distance to capacity. Formally, we can show the following result. For a sketch of the proof, see [1] or [8].

Theorem 5.1: For every R, $0 < R < 1$, every $\varepsilon > 0$, there is a polynomial time constructible family of codes over an alphabet of size $2^{O(1/\varepsilon^4)}$ that have rate at least R and which can be list decoded up to a fraction $(1 - R - \varepsilon)$ of errors in polynomial time.

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. Formally, we can show the following result (details omitted, see [1]).

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

References

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

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

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

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

[5] 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.

[6] 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.

[7] 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*, 2005, pp. 285–294.

[8] V. Guruswami, *List Decoding: Achieving Capacity for Worst-Case Errors*, ser. Foundations and Trends in Theoretical Computer Science (FnT-TCS). NOW publishers, 2006, to appear.

[9] 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*, June 2003, pp. 136–142.

[10] 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*, 2003, pp. 97–108.

[11] 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.

[12] 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.

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

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

[15] 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*, 2001, pp. 658–667.

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