List-Decoding Reed-Muller Codes over Small Fields

Parikshit Gopalan*
University of Washington
parik@cs.washington.edu

Adam R. Klivans†
UT Austin
klivans@cs.utexas.edu

David Zuckerman
UT Austin‡
diz@cs.utexas.edu

ABSTRACT

We present the first local list-decoding algorithm for the r^{th} order Reed-Muller code $\mathrm{RM}(r, m)$ over $\mathbb{F}_2$ for $r \geq 2$. Given an oracle for a received word $R : \mathbb{F}_2^m \to \mathbb{F}_2$, our randomized local list-decoding algorithm produces a list containing all degree r polynomials within relative distance $(2^{-r} - \varepsilon)$ from R for any $\varepsilon > 0$ in time $\mathrm{poly}(m^r, \varepsilon^{-r})$. The list size could be exponential in m at radius 2^{-r}, so our bound is optimal in the local setting. Since $\mathrm{RM}(r, m)$ has relative distance 2^{-r}, our algorithm beats the Johnson bound for $r \geq 2$.

In the setting where we are allowed running-time polynomial in the block-length, we show that list-decoding is possible up to even larger radii, beyond the minimum distance. We give a deterministic list-decoder that works at error rate below $J(2^{1-r})$, where $J(\delta)$ denotes the Johnson radius for minimum distance δ. This shows that $\mathrm{RM}(2, m)$ codes are list-decodable up to radius η for any constant $\eta < \frac{1}{2}$ in time polynomial in the block-length.

Over small fields $\mathbb{F}_q$, we present list-decoding algorithms in both the global and local settings that work up to the list-decoding radius. We conjecture that the list-decoding radius approaches the minimum distance (like over $\mathbb{F}_2$), and prove this holds true when the degree is divisible by $q - 1$.

1. INTRODUCTION

Most algorithms to decode error-correcting codes require that the received word is within less than half the minimum distance of a codeword, so that the codeword can be uniquely recovered. In the 1950s, Elias [9] and Wozencraft [30] introduced the notion of list-decoding in order to decode beyond this half-minimum-distance barrier. Instead of outputting one codeword, a list-decoding algorithm outputs all codewords within a specified radius of a received word. It took over thirty years before Goldreich and Levin [11] and Sudan [25] gave efficient list-decoding algorithms for Hadamard codes and Reed-Solomon codes, respectively. Since these breakthroughs, there has been much progress in devising list-decoders for various codes, which can decode beyond half the minimum distance. See the surveys by Guruswami [13, 14] and Sudan [26] for further details.

This study has had remarkable impact on other areas in computer science (see [29] and [13, Chapter 12]). In complexity theory, list-decodable codes give a way to amplify a weakly hard function to an extremely hard function [27, 28]. In cryptography, they give constructions of hard-core predicates from one-way functions [11, 1]. From the perspective of computational learning, list-decoding Hadamard codes is equivalent to learning parity functions with queries in the presence of adversarial noise. Such algorithms lie at the core of algorithms for learning fundamental concept classes like decision trees [20] and DNFs [16] (see [22]).

For the applications above, one needs the list-decoders to satisfy a stronger efficiency requirement: they must work in a *local* manner. Here one thinks of the received word R as a function on a huge domain, say $\mathbb{F}_2^m$. The algorithm is given access to an oracle for R, which returns $R(\mathbf{x})$ on query $\mathbf{x} \in \mathbb{F}_2^m$. The list-decoder is required to run in time $\mathrm{poly}(m)$, which means that if the block-length is n, it can only probe the received word at $(\log n)^{O(1)}$ indices.[1] Indeed, the Goldreich-Levin decoder works in this setting. We henceforth refer to algorithms that work in this restricted setting as local algorithms, and those that run in time polynomial in the block-length as global algorithms.

Reed-Muller Codes. The message space of the r^{th}-order Reed-Muller code $\mathrm{RM}(r, m)$ consists of degree r polynomials over $\mathbb{F}_2$ in m variables. The encoding of a polynomial P is the vector of its evaluations at all points in $\mathbb{F}_2^m$. Thus $\mathrm{RM}(r, m)$ is an $[n, k, d]_2$ error-correcting code where $n = 2^m, k = \sum_{i \leq r} \binom{m}{i}, d = 2^{m-r}$. When $r = 1$, we get Hadamard codes. This is a fundamental family of error-correcting codes, see [21, Chapters 13-15]. To quote [21]:

Reed-Muller (or RM) codes are one of the oldest and best understood families of codes ... The great merit of RM codes is that they are relatively easy to decode, using majority-logic circuits.

This refers to Reed's elegant Majority Logic Decoder, which solves the unique-decoding problem up to error rate $2^{-(r+1)}$, or half the minimum distance [21]. This algorithm can

*Work done in part while the author was at UT-Austin
†Research supported by an NSF CAREER Award and NSF Grant CCF-0728536
‡Research partially supported by NSF grant CCF-0634811.

STOC'08, May 17–20, 2008, Victoria, British Columbia, Canada.

[1] If the received word does not have description size $\mathrm{poly}(m)$, then the definition becomes more subtle. Since we do not encounter this setting, we refer the reader to [27] for the definition.

be adapted to work in the local setting up to error rate $2^{-(r+1)} - \varepsilon$. Following the seminal work of Goldreich-Levin [11], other list-decoders for Hadamard codes were given by Kushilevitz-Mansour[20] and Goldreich *et al.*[12]. The problem of list-decoding RM codes when $r \geq 2$ has been open for a while and has attracted interest in both coding theory and computer science; this question has been raised by [24, 6, 7]. It is easy to see that 2^{-r} is an upper bound on the list-decoding radius in the local setting. Prior to our work, the best known lower bound seems to have been $J(2^{-r})$, where $J(\delta) = \frac{1}{2}(1 - \sqrt{1-2\delta})$ is the Johnson bound for minimum distance δ. In the random noise model, Dumer gives an $m^{O(r)}$ local algorithm that solves the decoding problem up to an error rate of $\frac{1}{2} - o(1)$ [7].

Our Results. We give the first local list-decoding algorithm for $\mathrm{RM}(r, m)$ codes for $r \geq 2$. Our algorithm runs in time $\mathrm{poly}(m^r, \varepsilon^{-r})$ and produces a list of size at most $\varepsilon^{-O(r)}$ containing all codewords within a radius of $2^{-r} - \varepsilon$ from the received word. At radius 2^{-r}, the list size can be exponential in m hence our bound is optimal in the local setting. We show that the list size at distance $2^{-r} - \varepsilon$ is bounded by $\varepsilon^{-O(r)}$, and we exhibit configurations of received words and codewords showing that this is asymptotically optimal. Our algorithm shows that RM codes can be locally decoded well beyond the Johnson bound for $r \geq 2$. When $r = 2$, the Johnson bound gives 0.146 whereas we show that the radius approaches 0.25 (see Figure 1).

For $r \geq 2$, The problem of Reed-Muller list-decoding is interesting in the global setting, where we allow running time $\mathrm{poly}(n) = 2^{O(m)}$. One could hope to list-decode as long as the list size remains bounded by $2^{O(m)}$. Since the total number of codewords is 2^{m^r}, an upper bound of $2^{O(m)}$ is nontrivial. We present a global list-decoder which works up to an error rate of $J(2^{1-r}) - \varepsilon$; the Johnson bound for *twice* the minimum distance. The algorithm runs in time $\varepsilon^{-O(m)}$ where $n = 2^m$ denotes the block-length. Thus $\mathrm{RM}(2, m)$ is list-decodable in the global setting up to a radius η for any constant $\eta < \frac{1}{2}$. For this setting, the best combinatorial and algorithmic bounds previously known seem to be the Johnson bound $J(2^{-r})$ and half the minimum distance $2^{-(r+1)}$ respectively. Figure 1 summarizes the various bounds for RM decoding.

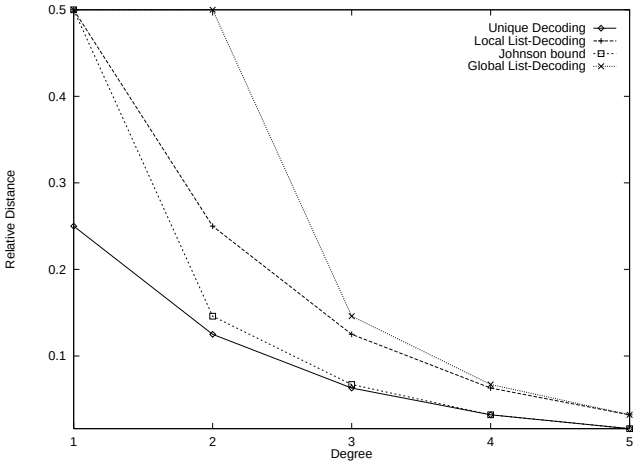

Figure 1: List-Decoding Reed-Muller codes over $\mathbb{F}_2$

Overview of the Algorithm. Our local list-decoder gives a natural generalization of the Goldreich-Levin (GL) decoder for linear polynomials to higher degree. We view the GL decoder as a reduction from list-decoding to unique-decoding (given the right advice); this view extends naturally to higher degrees and other fields. This view of the GL algorithm is along the lines of the description in [10] (where it is attributed to Levin and Rackoff). We recap the GL decoder below.

Fix a linear polynomial L within distance $\eta = \frac{1}{2} - \varepsilon$ from the received word R. Assume that we guess the correct values of L on a random subspace $\mathbf{A}$ of dimension k. Using this, we get a self-corrector for any point $\mathbf{b} \in \mathbb{F}_2^n$: consider the $k+1$ dimensional subspace $\mathbf{A}'$ spanned by $\mathbf{b}$ and the vectors in $\mathbf{A}$. Since the error rate is 0 on $\mathbf{A}$ and close to η on the affine space $\mathbf{b} + \mathbf{A}$ (with failure probability $O(\frac{1}{\varepsilon^2|A|})$ by pairwise independence), the overall error-rate on $\mathbf{A}'$ is likely to be $\eta/2 < \frac{1}{4}$ which is within the unique decoding radius. So we can correct the values on $\mathbf{A}'$ by unique decoding. Using this self-corrector, there are two ways to solve the local decoding problem:

1. Set $|\mathbf{A}| = O(m/\varepsilon^2)$. The self-corrector works for any point with probability $1 - O(1/m)$. We can self-correct at $\mathbf{e}_1, \cdots, \mathbf{e}_m$ and interpolate L from these values.
2. Pick $|\mathbf{A}| = O(1/\varepsilon^2)$. The self-corrector works for any point with constant probability, say 0.9, so we can no longer use the union bound over m points. However, given interpolating sets that can handle a 0.1 fraction of incorrect values, we can still recover L. Such sets can be obtained from a linear code which is uniquely decodable up to error rate 0.1. The list size is governed by the number of guesses for L restricted to $\mathbf{A}$. This approach yields the tight bound of $O(\varepsilon^{-2})$.

Now let us try and generalize this to degree 2. Fix a quadratic polynomial Q within distance $\eta < \frac{1}{4} - \varepsilon$ from the received word R. Again, if we know restriction of Q to a random subspace $\mathbf{A}$, then the error rate over $\mathbf{A}'$ drops to $\eta/2$, which is within the unique-decoding radius for $RM(2, m)$. So we can self-correct on $\mathbf{A}'$ via unique-decoding.

However, the translation from self-correction to local-decoding is now more subtle. The first approach where we take $|A| = O(m^2/\varepsilon^2)$ is better for the purposes of recovering the polynomial. The second approach where we take $|A| = O(\varepsilon^{-2})$ gives better bounds on the list size. Further, guessing the restriction of Q to $|A|$ is no longer efficient: it gives a running time which is quasi-polynomial in m and/or ε^{-1} (ε can be inverse polynomial in m). Our solution is to use the first approach for the algorithm, and the second for the combinatorial bound. Further we generate the advice algorithmically, rather than by guessing.

We begin with Approach 1 where $|A| = O(m^2/\varepsilon^2)$. We observe that Q is likely to be at distance η from R on the subspace $\mathbf{A}$ itself. So rather than guessing the restriction of Q from $m^{O(\log m)}$ choices, if we can solve the list-decoding problem over $\mathbf{A}$, then the restriction of Q is likely to be in the list. But the decoder for $\mathbf{A}$ need not be local, since $\mathbf{A}$ has size $O(m^2/\varepsilon^2)$. Essentially, we have reduced the local decoding problem to the global decoding problem over $\mathbf{A}$. We present an algorithm for the global problem, which for any error-rate η runs in time polynomial in $|\mathbf{A}|$ and the maximum list size possible. For this decoder to run in time $\mathrm{poly}(m)$, we need to bound the list size.

We prove the combinatorial bound in two steps. In the

first step, we use Approach 2 to show that knowing the values of Q at a random affine space $\mathbf{A}$, where $|\mathbf{A}| = O(\varepsilon^{-2})$, suffices to recover Q. This implies a naive bound of $2^{(\log(\varepsilon^{-1}))^2}$ on the list size. We improve this bound to $\text{poly}(\varepsilon^{-1})$, using facts about the weight distribution of Reed-Muller codes. We show that one can effectively *double* the distance of the $\text{RM}(r, m)$ codes by deleting relatively few codewords. This is easy to prove for $r = 2$; for degree 3 and above it relies on an elegant result by Kasami and Tokura [19]. This allows us to apply the Johnson bound for twice the minimum distance of the original RM code. Our deletion argument can applied to any linear code where one has good bounds on the weight-distribution of codewords beyond the minimum distance and might be useful in other settings. The deletion argument coupled with our global decoder allows us to solve the global decoding problem at error-rates approaching $\frac{1}{2}$ for quadratic polynomials.

Extension to Other Fields. There is a natural generalization of Reed-Muller codes over any finite field $\mathbb{F}_q$; such codes are referred to as generalized Reed-Muller codes. We extend our algorithmic results to decoding generalized Reed-Muller codes over small fields $\mathbb{F}_q$. For $\text{RM}_q(r, m)$ codes, we give a randomized local list-decoding algorithm that works for any error rate η where one can upper bound the list size by a constant (independent of m). This algorithm combines ideas from our local list-decoder for $\mathbb{F}_2$ and the algorithm for list-decoding Reed-Muller codes over large fields by Sudan *et al.*[27]. We give a deterministic global list-decoder that runs in time polynomial in both the block-length and the worst-case list size for any error rate η.

The problem of exactly determining the list-decoding radius in either setting remains open. We conjecture that in the local setting, the list-decoding radius lies close to the minimum distance, as in the $\mathbb{F}_2$ case. We prove that this is indeed the case for all degrees that are divisible by $q - 1$. We prove that for any degree r, the list-decoding radius is at least $\delta_q(r - 1)/2$; this beats the Johnson bound for sufficiently large r. Figure 2 illustrates our bounds for $\mathbb{F}_3$.

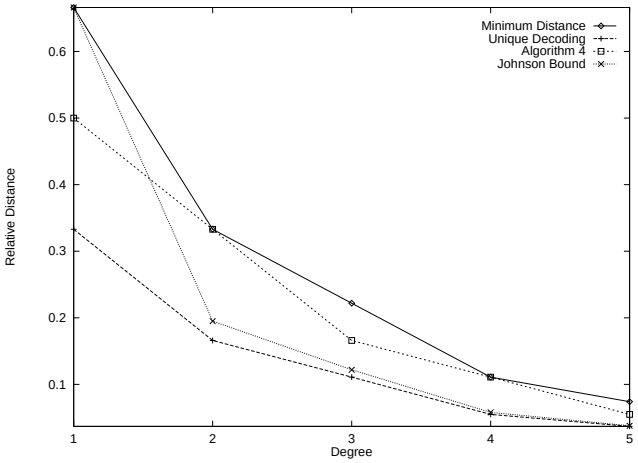

Figure 2: List-Decoding Reed-Muller codes over $\mathbb{F}_3$

Related Work. There has been tremendous progress on list-decoding Reed-Muller codes in the *large field* setting where $d < q$. Two important results in this area include the Gurswami-Sudan list-decoder for Reed-Solomon codes that can correct $\eta = 1 - \sqrt{d/q}$ fraction of errors [15], and the Sudan-Trevisan-Vadhan local list-decoder which corrects $\eta = 1 - \sqrt{2d/q}$ fraction of errors [27]. Pellikaan and Wu give a reduction from Reed-Muller decoding to Reed-Solomon decoding when $d < q$ [23], which allows list-decoding up to $\eta = 1 - \sqrt{d/q}$ in the global setting. Other work addressing the large field setting includes [3, 12]. These results require $d < q$ as they work by reduction to the univariate case.

A closely related problem that has received considerable attention lately is that of locally testing RM codes. Blum *et al.* [4] give an efficient 3-query tester for Hadamard codes. Recently, Alon *et al.*[2] showed that RM codes are locally testable for $r \geq 2$, by giving a test using $O(2^r)$ queries that rejects a received word at distance 2^{-r} form the code with constant probability. For $r = 2$, Samorodnitsky gave a tight analysis of their test and showed that it can detect the presence of a codeword within a radius of $\frac{1}{2} - \varepsilon$ [24]. The Gowers inverse conjecture asserted that a similar result holds true for all degrees r, but this was recently disproved.

Recent Developments. In this paper, we use a random subspace $\mathbf{A}$ of size $O(m/\varepsilon^2)$ in the decoding algorithm, but in the combinatorial arguments, we use $|\mathbf{A}|$ of size $O(\varepsilon^{-2})$. This was necessary since at the time of this work, we were unaware of interpolating sets for polynomials of degree 2 and higher that could handle errors. This problem has recently been solved by Dvir and Shpilka, who give an elegant construction of such sets [8]. Their result lets us pick $|\mathbf{A}| = O(\varepsilon^{-2})$ even in the algorithm. However, the combinatorial arguments stay the same. Subsequently, we discovered an alternate algorithmic approach which lets one choose $|\mathbf{A}| = O(\varepsilon^{-2})$, but bypasses the need for noise-tolerant interpolating sets, by using a local version of Majority Logic Decoding. We defer the details to the full version.

We present our local list-decoder for RM codes in Section 2, the global list-decoder in Section 3 and the combinatorial bounds on list size in Section 4. We present the $\mathbb{F}_q$ results in Section 5.

Preliminaries. Let $\mathbf{x}, \mathbf{y}$ denote vectors in $\mathbb{F}_2^m$ and $d(\mathbf{x}, \mathbf{y})$ the Hamming distance between them. Let $\mathbf{0}$ be the all-zeroes vector. Let $n = 2^m$. We identify vectors in $\mathbb{F}_2^n$ with functions mapping $\mathbb{F}_2^m \to \mathbb{F}_2$. Codewords are polynomials of degree at most r, and the received word is a function $R : \mathbb{F}_2^m \to \mathbb{F}_2$. Define the normalized distance between functions as $\Delta(P, R) = \Pr_{\mathbf{x} \in \mathbb{F}_2^m}[P(\mathbf{x}) \neq R(\mathbf{x})]$. Let $\text{wt}(P) = \Delta(P, \mathbf{0})$ be the normalized Hamming weight of P. Let $\ell(r, m, \eta)$ denote the maximum number of $\text{RM}(r, m)$ codewords in an open ball of radius η around a received word.

For a set $S \subseteq \mathbb{F}_2^m$ and $\mathbf{x} \in \mathbb{F}_2^m$, let $\mathbf{x} + S = \{\mathbf{x} + \mathbf{y} \mid \mathbf{y} \in S\}$. A set $S \in \{0, 1\}^m$ is an interpolating set for degree-r polynomials if every polynomial P with $\deg(P) \leq r$ is uniquely determined by its evaluations at points in S. If S is an interpolating set for degree-r polynomials, then every affine translate $\mathbf{a} + S$ of S is also an interpolating set. Let $\mathcal{B}(\mathbf{c}, r)$ denote the Hamming ball of radius r centered at $\mathbf{c}$: it is an interpolating set for degree-r polynomials over $\mathbb{F}_2$.

Any $k + 1$ vectors $\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_k \in \mathbb{F}_2^m$ define an affine space $\mathbf{A} = \{\mathbf{a}_0 + \sum_{i=1}^{k} \lambda_i \mathbf{a}_i | \lambda_i \in \mathbb{F}_2\}$. We will sample a random k-dimensional affine subspace by choosing $\mathbf{a}_1, \ldots, \mathbf{a}_k$ to be linearly independent, and then choosing $\mathbf{a}_0$ randomly. We define the normalized Hamming distance between two functions P and R over $\mathbf{A}$ as $\Delta_{\mathbf{A}}(P, R) = \Pr_{\mathbf{x} \in \mathbf{A}}[P(\mathbf{x}) \neq R(\mathbf{x})]$. For a polynomial P, let $P_{\mathbf{A}}$ denote its restriction to

the subspace $\mathbf{A}$, so that $\deg(P_{\mathbf{A}}) \leq \deg(P)$. We use the following Subspace Sampling Lemma repeatedly, and defer the (standard) proof.

LEMMA 1. (Subspace Sampling Lemma) *Fix $f : \mathbb{F}_2^m \to \{0,1\}$. For a random k-dimensional affine subspace $\mathbf{A}$ of $\mathbb{F}_2^m$, with probability at least $1 - \frac{1}{\varepsilon^2 2^k}$,*

$$\mathrm{E}_{\mathbf{x}\in\mathbf{A}}[f(\mathbf{x})] < \mathrm{E}_{\mathbf{x}\in\mathbb{F}_2^{\mathbf{m}}}[f(\mathbf{x})] + \varepsilon.$$

2. A LOCAL LIST-DECODER

PROBLEM 1. **(Local List-Decoding)** *Given oracle access to $R : \mathbb{F}_2^m \to \mathbb{F}_2$, find a list $\mathcal{L} = \{P_1, \cdots, P_\ell\}$ of all polynomials P_i s.t $\deg(P_i) \leq r$ and $\Delta(P_i, R) \leq 2^{-r} - \varepsilon$ in time $\mathrm{poly}(m^r, \ell(r, m, 2^r - \varepsilon))$.*

In Section 4, we will show that for any r, $\ell(r, m, 2^{-r} - \varepsilon) \leq 2(2^{r+5}\varepsilon^{-2})^{4r}$. Set $L = 2(2^{r+5}\varepsilon^{-2})^{4r}$ and $B = |\mathcal{B}(\mathbf{a}, r)|$ so that $B = O(m^r)$.

ALGORITHM 1. LOCAL-DECODE$(R, r, 2^{-r} - \varepsilon)$

1. Pick a random k-dimensional affine space $\mathbf{A}$ where $2^k > cLB\varepsilon^{-2}$.
2. Run Global-Decode$(R_{\mathbf{A}}, r, 2^{-r} - \frac{\varepsilon}{2})$ on $\mathbf{A}$ to get list $\mathcal{L}_{\mathbf{A}}$.
3. For each $Q \in \mathcal{L}_{\mathbf{A}}$,
   a. Set $R(\mathbf{x}) = Q(\mathbf{x})$ for $\mathbf{x} \in \mathbf{A}$.
   b. For $\mathbf{b} \in \mathcal{B}(\mathbf{0}, r)$,
      Unique-decode R on $\mathbf{A}' = \mathbf{A} \cup (\mathbf{b} + \mathbf{A})$ to get Q'.
      Set $P(\mathbf{b} + \mathbf{a}_0) = Q'(\mathbf{b} + \mathbf{a}_0)$.
   c. Interpolate P from the values $P(\mathbf{b}+\mathbf{a}_0)$.
   d. If $\Delta(P, R) < 2^{-r} - \frac{\varepsilon}{2}$, add P to $\mathcal{L}^\star$.
4. Output the List $\mathcal{L}^\star$.

We present the Global List-Decoder in Section 3. We solve the Unique-Decoding problem using Majority Logic Decoding. In Step 3b, if unique-decoding fails, we set $P(\mathbf{b}+\mathbf{a}_0)$ arbitrarily.

THEOREM 2. *Algorithm 1 returns a list of size $\mathrm{poly}(\varepsilon^{-1})$ containing all polynomials such that $\Delta(P, R) < 2^{-r} - \varepsilon$ with probability $1 - O(1/c)$.*

Proof: Let $\mathcal{L}$ denote the true list of all polynomials P such that $\Delta(P, R) < 2^{-r} - \varepsilon$, so that $|\mathcal{L}| \leq L$. Fix $P \in \mathcal{L}$. Over a random choice of A, for every $\mathbf{b} \in \mathcal{B}(\mathbf{0}, r)$, the affine space $\mathbf{b} + \mathbf{A}$ is random. Hence

$$\Pr_{\mathbf{A}}\left[\Delta_{\mathbf{b}+\mathbf{A}}(P, R) \geq 2^{-r} - \frac{\varepsilon}{2}\right] \leq \frac{4}{\varepsilon^2 2^k}.$$

By the union bound, the condition $\Delta_{\mathbf{b}+\mathbf{A}}(P, R) < 2^{-r} - \frac{\varepsilon}{2}$ holds over the choice $\mathbf{A}$ for every polynomial $P \in \mathcal{L}$ and $\mathbf{b} \in \mathcal{B}(\mathbf{0}, r)$ with probability

$$1 - \frac{4LB}{\varepsilon^2 2^k} \geq 1 - \frac{4}{c}$$

We call such a subspace $\mathbf{A}$ good. Assuming $\mathbf{A}$ is good, fix a polynomial $P \in \mathcal{L}$. The restriction $P_{\mathbf{A}}$ will belong to the list $L_{\mathbf{A}}$ recovered in Step 2 (assuming we can do global list-decoding).

In Step 3, when the algorithm sets $R(\mathbf{x}) = P_{\mathbf{A}}(\mathbf{x})$ for $\mathbf{x} \in \mathbf{A}$, we have the correct values of P for the subspace $\mathbf{A}$, so $\Delta_{\mathbf{A}}(P, R) = 0$. Now our aim is to self-correct the values at $\mathbf{b} + \mathbf{A}$ for $\mathbf{b} \in \mathcal{B}(\mathbf{0}, r)$. It might happen that $\mathbf{b} + \mathbf{A} = \mathbf{A}$, in which case we are done. Otherwise $\mathbf{A}$ and $\mathbf{b} + \mathbf{A}$ are disjoint. Since $\mathbf{A}' = \mathbf{A} \cup (\mathbf{b} + \mathbf{A})$, we get

$$\Delta_{\mathbf{A}'}(P, R) = \frac{\Delta_{\mathbf{A}}(P, R) + \Delta_{\mathbf{b}+\mathbf{A}}(P, R)}{2} = \frac{\Delta_{\mathbf{b}+\mathbf{A}}(P, R)}{2}.$$

Since $\mathbf{A}$ is good, $\Delta_{\mathbf{b}+\mathbf{A}}(P, R) < 2^{-r}$, hence $\Delta_{\mathbf{A}'}(P, R) < 2^{-(r+1)}$, which is within the unique decoding radius, we will recover $Q' = P_{\mathbf{A}'}$ in Step 3a. We have self-corrected P over all the subspaces $\mathbf{b} + \mathbf{A}$ for $\mathbf{b} \in \mathcal{B}(\mathbf{0}, r)$, which includes the interpolating set $\mathcal{B}(\mathbf{a}_0, r)$. Therefore we can recover the polynomial P in Steps 3b, 3c.

This shows that if $\mathbf{A}$ is good, then $\mathcal{L} \subseteq \mathcal{L}^\star$. We need to show that $\mathcal{L}^\star$ does not contain too many other polynomials. For this, observe that $|\mathcal{L}^\star| \leq |\mathcal{L}_{\mathbf{A}}|$ since $\mathcal{L}^\star$ contains at most one polynomial for each $Q \in \mathcal{L}_{\mathbf{A}}$. Also $|\mathcal{L}_{\mathbf{A}}| \leq \ell(r, k, 2^{-r} - \varepsilon/2)$. By random sampling, one can estimate $\Delta(P, R)$ and discard any polynomials any $P \in \mathcal{L}^\star$ where $\Delta(P, R) > 2^{-r} - \varepsilon/2$. ∎

To analyze the running time, we need to give an efficient global list-decoder, and prove the combinatorial list size bounds that were used in the analysis. We address these issues in Sections 3 and 4 respectively, and then analyze the time complexity in Theorem 13.

3. A GLOBAL LIST-DECODER

We present a global list-decoder with a strong guarantee: for any error rate η, its running time is polynomial in both the block-length and maximum list size possible for error η.

PROBLEM 2. **(Global List-Decoding)** *Given a received word $R : \mathbb{F}_2^m \to \mathbb{F}_2$, produce a list $\mathcal{L} = \{P_1, \cdots, P_\ell\}$ of all degree r polynomials such that $\Delta(P_i, R) \leq \eta$ in time $\mathrm{poly}(2^m, \ell(r, m, \eta))$.*

We think of the input space as an affine space $\mathbf{A}$ of dimension m, and $R : \mathbf{A} \to \mathbb{F}_2$ as a function on it. Parametrize $\mathbf{A}$ by $X_1, \ldots, X_m$, we can partition it into $\mathbf{A}^0 = \{\mathbf{x} \in \mathbb{F}_2^m | x_m = 0\}$ and $\mathbf{A}^1 = \{\mathbf{x} \in \mathbb{F}_2^m | x_m = 1\}$. For any received word R and polynomial P in the list, we have $\Delta(P, R) = \frac{1}{2}(\Delta_{\mathbf{A}^0}(P, R) + \Delta_{\mathbf{A}^1}(P, R)) < \eta$. If we fix b so that $\Delta_{\mathbf{A}^b}(P, R) \leq \Delta_{\mathbf{A}^{1-b}}(P, R)$ then $\Delta_{\mathbf{A}^b}(P, R) \leq \eta$ and $\Delta_{\mathbf{A}^{1-b}}(P, R) \leq 2\eta$. Since the error-rate over $\mathbf{A}^b$ does not increase but the dimension reduces to $m - 1$, we can use recursion. By list-decoding R over $\mathbf{A}^b$ we find a list $\mathcal{L}^b$ of polynomials which includes $P_{\mathbf{A}^b}$. By enumerating over all polynomials in the list, we can assume that we know $P_{\mathbf{A}^b}$. We can write $P_{\mathbf{A}}(X_1, \ldots, X_m) = P_{\mathbf{A}^b}(X_1, \ldots, X_{m-1}) + (X_m + b)Q'(X_1, \ldots, X_{m-1})$. We will try to recover Q' from $\mathbf{A}^{1-b}$ using $R + P_{\mathbf{A}^b}$ as the received word. While the error rate doubles to 2η, $\deg(Q') \leq r - 1$, hence the minimum distance also doubles, allowing us to use recursion.

ALGORITHM 2. GLOBAL-DECODE($\mathbf{A}, R, r, \eta$)

For $b \in \mathbb{F}_2$,
  Set $\mathcal{L}^b$ = Global-Decode($\mathbf{A}^b, R_{\mathbf{A}^b}, r, \eta$).
  For each $Q \in \mathcal{L}^b$,
    Set $\mathcal{L}^{1-b}$ = Global-Decode($\mathbf{A}^{1-b}, R_{\mathbf{A}^{1-b}} + Q, r - 1, 2\eta$).
    For each $Q' \in \mathcal{L}^{1-b}$,
      Set $P(X_1, \ldots, X_m) = Q(X_1, \ldots, X_{m-1}) + (X_m + b_m)Q'(X_1, \ldots, X_{m-1})$
      If $\Delta_{\mathbf{A}}(P, R) < \eta$, add P to $\mathcal{L}$.
Return $\mathcal{L}$.

The base case of the algorithm is reached when $2^m \cdot \eta < 1$ or when $r = 1$. In the former case, the error-rate is 0, so the polynomial can be recovered by interpolation. In the latter case we can use the Goldreich-Levin algorithm or even brute force search. The following lemma shows that the list size does not increase during the recursive calls.

LEMMA 3. *For any m, r and η, $\ell(r-1, m-1, 2\eta) \leq \ell(r, m, \eta)$.*

Proof: Let $R \in \mathbb{F}_2^{m-1}$ be a received word and $\{P_1, \ldots, P_\ell\}$ be a list of polynomials in $X_1, \ldots, X_{m-1}$ of degree $r-1$ such that $\Delta(P, R) < 2\eta$. Define the polynomials $\{P'_1, \ldots, P'_\ell\}$ in $X_1, \ldots, X_m$ and the received word $R' \in \mathbb{F}_2^m$ as

$$P'_i(X_1, \ldots, X_m) = X_m P_i(X_1, \ldots, X_{m-1}),$$
$$R'(X_1, \ldots, X_m) = X_m R(X_1, \ldots, X_{m-1}).$$

It is easy to see that $\Delta(R', P'_i) < \eta$ which proves the claim. ∎

THEOREM 4. *Algorithm 2 runs in time $2^{3m} \cdot \ell(r, m, \eta)^r$ and returns a list of all polynomials P such that $\deg(P) \leq r$ and $\Delta(P, R) < \eta$.*

Proof: The running time $T(r, m, \eta)$ satisfies

$$T(r, m, \eta) \leq 2(T(r, m-1, \eta) + \ell(r, m, \eta)T(r-1, m-1, 2\eta) + \ell(r, m, \eta)\ell(r-1, m-1, 2\eta)2^m)$$

Let $\ell = \ell(r, m, \eta)$. Then by Lemma 3, we can write

$$T(r, m, \eta) \leq 2(T(r, m-1, \eta) + \ell T(r-1, m-1, 2\eta) + \ell^2 2^m)$$

We will prove that $T(r, m, \eta) < 2^{3m}\ell^r$ by double induction of m and r. The base case $r = 1$ is easy to verify. For the inductive case $r \geq 2$, we have

$$\begin{aligned} T(r, m, \eta) &\leq 2(2^{3(m-1)}\ell^r + \ell 2^{3(m-1)}\ell^{r-1} + 2^m\ell^2) \\ &= 2(2^{3m-2}\ell^r + 2^m\ell^2) < 2^{3m}\ell^r. \end{aligned}$$

It is easy to verify the correctness of the algorithm by induction. ∎

4. COMBINATORIAL BOUNDS

Since we have shown that the running times of our decoders depend polynomially on the list size, we now bound the list size. We prove our bound in two steps. In the first dimension-reduction step, we show that to bound the list size for error-rate $2^{-r} - \varepsilon$ for m-dimensional spaces, it suffices to prove a bound for subspaces of dimension $O(\log(\varepsilon^{-1}))$ and error-rate 2^{-r}. In the second step, we use properties of the weight-distribution of RM codes to show that it is possible to nearly double the minimum distance of RM codes by deleting relatively few codewords. We apply the Johnson bound to this code and infer bounds for the original RM code.

4.1 Dimension Reduction

PROBLEM 3. **(Bounded Distance Decoding)** *Given a received word $R : \mathbb{F}_2^m \to \mathbb{F}_2$, produce a polynomial P of degree r such that $\Delta(P, R) \leq \eta$.*

For any polynomial P s.t. $\Delta(P, R) \leq 2^{-r} - \varepsilon$, we show that our algorithm returns that specific polynomial P with probability at least p, which implies that $\ell(r, m, 2^{-r} - \varepsilon) \leq \frac{1}{p}$. We use Algorithm 1 with some modifications. We pick a random subspace $\mathbf{A}$ with $|\mathbf{A}| = O(\varepsilon^{-2})$, and guess the restriction of P to $\mathbf{A}$. If this succeeds, we can only guarantee that self-correction works for a random point with constant probability. But running self-correction at every point in $\mathbb{F}_2^m$, followed by unique decoding will recover R with reasonable probability. The algorithm is inefficient, but suffices for a combinatorial bound.

Fix a polynomial P so that $\Delta(P, R) < (2^{-r} - \varepsilon)$. We say that a k-dimensional affine subspace $\mathbf{A}$ is good for P if the following conditions hold:
1. The subspace $\mathbf{A}$ has low error rate with respect to P: $\Delta_{\mathbf{A}}(P, R) < 2^{-r}$.
2. Most affine shifts of $\mathbf{A}$ have low error with respect to P: $\Pr_{\mathbf{b} \in \mathbb{F}_2^m}[\Delta_{\mathbf{b}+\mathbf{A}}(P, R) \geq 2^{-r}] < 2^{-(r+1)}$.

One can use the Subspace Sampling Lemma to prove that if $2^k > 2^{r+3}\varepsilon^{-2}$, a random k-dimensional affine space $\mathbf{A}$ is good for P with probability at least $\frac{1}{2}$.

ALGORITHM 3. RECOVER($R, r, 2^{-r} - \varepsilon$)

1. Pick a random k-dimensional affine subspace $\mathbf{A}$ where $2^k > 2^{r+3}\varepsilon^{-2}$.
2. Find list $\mathcal{L}$ of all Q s.t. $\deg(Q) \leq r$ and $\Delta_{\mathbf{A}}(Q, R) < 2^{-r}$.
   Choose $P_{\mathbf{A}}$ randomly from $\mathcal{L}$.
   Set $R(\mathbf{x}) = P_{\mathbf{A}}(\mathbf{x})$ for $\mathbf{x} \in \mathbf{A}$.
3. For each $\mathbf{b} \in \mathbb{F}_2^m$,
   a. Let $\mathbf{A}' = \mathbf{A} \cup (\mathbf{b} + \mathbf{A})$. Run unique decoding on $R_{\mathbf{A}'}$ to get Q'.
   b. Set $P(\mathbf{b} + \mathbf{a}_0) = Q'(\mathbf{b} + \mathbf{a}_0)$.
4. Run unique decoding on P and output the result.

THEOREM 5. *Let $2^k > 2^{r+3}\varepsilon^{-2}$ and fix a polynomial P so that $\Delta(P, R) < 2^{-r} - \varepsilon$. Algorithm 3 recovers P with probability at least $(2\ell(r, k, 2^{-r}))^{-1}$.*

Proof: In Step 1, we pick a good subspace for P with probability at least $\frac{1}{2}$. Assume that this happens. In Step 2, we aim to correct the values of R on $\mathbf{A}$. We construct a list of at most $\ell = \ell(r, k, 2^{-r})$ polynomials Q of degree r so that $\Delta_{\mathbf{A}}(Q, R) < 2^{-r}$ using brute force. Since $\mathbf{A}$ is good for P,

this list contains the polynomial $P_{\mathbf{A}}$. We guess the correct polynomial with probability $\frac{1}{\ell}$. Assuming this happens, we correct the values of R on $\mathbf{A}$, and so $\Delta_{\mathbf{A}}(P,R)=0$.

In Step 3, we aim to correct the values of R on most points $\mathbf{b}\in\mathbb{F}_2^m$. If $\mathbf{b}$ is such that $\Delta_{\mathbf{b}+\mathbf{A}}(P,R)<2^{-r}$,

$$\begin{aligned}\Delta_{\mathbf{A}'}(P,R) &= \frac{1}{2}(\Delta_{\mathbf{A}}(P,R)+\Delta_{\mathbf{b}+\mathbf{A}}(P,R))\\ &= \frac{1}{2}\Delta_{\mathbf{b}+\mathbf{A}}(P,R) < 2^{-(r+1)}\end{aligned}$$

Since this is less than the unique decoding radius, we will recover $Q'=P_{\mathbf{A}'}$ in Step 3a. and assign the correct value to $\mathbf{b}+\mathbf{a}_0$ in Step 3b. Since $\mathbf{A}$ is good for P, self-correction fails for no more than $2^{-(r+1)}$ fraction of $\mathbf{b}\in\mathbb{F}_2^m$ which is within the unique decoding radius. So we can recover P using Unique Decoding in Step 4. Overall the algorithm succeeds with probability $\frac{1}{2\ell}$. ∎

COROLLARY 6. *For any $\varepsilon>0$ and k s.t. $2^k>2^{r+3}\varepsilon^{-2}$, $\ell(r,m,2^{-r}-\varepsilon)\le 2\ell(r,k,2^{-r})$.*

The total number of degree r polynomials on a k-dimensional subspace is bounded by 2^{k^r}. This shows that $\ell(r,m,2^{-r}-\varepsilon)\le O(2^{\log(\varepsilon^{-1})^r})$. While this suffices to prove that the list size is constant for ε constant, it does not imply a poly(m) bound when $\varepsilon=1/\operatorname{poly}(m)$. In the next section, we address this by showing that the list size is bounded by poly(ε^{-1}).

4.2 Combinatorial Bounds via Deletion

We prove a lemma which implies better bounds on the list-decoding radius of any linear code which has few low-weight codewords. We state it over $\mathbb{F}_2$ for simplicity, but it can be applied to linear codes any finite field. We will use the Johnson bound [17, 18], which applies to arbitrary codes, including non-linear ones. Define $J(\alpha)=\frac{1}{2}(1-\sqrt{1-2\alpha})$.

LEMMA 7. **(Johnson Bound)** *Let $\mathcal{C}\subseteq\{0,1\}^n$ be a code with distance δn. For any $R\in\{0,1\}^n$,*
1. The number of $C\in\mathcal{C}$ such that $\Delta(R,C)<J(\delta)-\gamma$ is bounded by $c(\gamma)=O(\gamma^{-2})$.
2. The number of $C\in\mathcal{C}$ such that $\Delta(R,C)<J(\delta)$ is bounded by $2n$.

For a linear code $\mathcal{C}\subset\{0,1\}^n$ and $\alpha\in[0,1]$, Let $A(\alpha)$ denote the number of codewords in $\mathcal{C}$ of normalized weight less than α.

LEMMA 8. *Let $\mathcal{C}\subseteq\{0,1\}^n$ be a* linear *code. For any $\alpha\in[0,1]$ and any $R\in\{0,1\}^n$,*
1. The number of $C\in\mathcal{C}$ such that $\Delta(R,C)<J(\alpha)-\gamma$ is bounded by $A(\alpha)\cdot c(\gamma)$.
2. The number of $C\in\mathcal{C}$ such that $\Delta(R,C)<J(\alpha)$ is bounded by $2A(\alpha)n$.

Proof: Let $\mathcal{L}=\{C_1,\ldots,C_L\}$ be the list of codewords such that $\Delta(R,C_i)<J(\alpha)-\varepsilon$. We greedily prune $\mathcal{L}$ to get a new list $\mathcal{L}^\star$ so that any two codewords on $\mathcal{L}^\star$ satisfy $\Delta(P,Q)\ge\alpha$. Pick a codeword $C\in\mathcal{L}$, add it to $\mathcal{L}^\star$ and delete all codewords $D\in\mathcal{L}$ such that $\Delta(C,D)<\alpha$. We charge the deleted codewords to C. Repeat until there are no codewords left in $\mathcal{L}$. Every pair of codewords in $\mathcal{L}^\star$ is at distance at least αn. The linearity of $\mathcal{C}$ implies that at most $A(\alpha)$ deleted codewords are charged to any $C\in\mathcal{L}^\star$, so $|\mathcal{L}|\le|\mathcal{L}^\star|\cdot A(\alpha)$. We can think of $\mathcal{L}^\star$ as a (non-linear) code with minimum distance αn. Applying the Johnson bound with received word R and list $\mathcal{L}^\star$ completes the proof. ∎

Linearity is only needed to bound the number of codeword within distance α from an arbitrary codeword. One can recover the Johnson bound from Lemma 8 by setting $\alpha=\delta$ and $A(\delta)=1$. To apply this lemma we need bounds on the weight distribution of Reed-Muller codes.

4.3 Weight Distribution of Reed-Muller codes

When $r=2$, we can completely describe the weight-distribution of Reed-Muller codes using the notion of rank of a quadratic form [21]. For $r\ge 3$, the weight distribution of RM codes is not well understood (to the best of our knowledge) , see [21, Research Problem 15.1]. However, an elegant result of Kasami and Tokura which characterizes RM codewords with weight less than twice the minimum distance suffices for us.

THEOREM 9. *[19]* **(Kasami-Tokura)** *Let $r\ge 2$. Let P be a polynomial with $\deg(P)\le r$ such that $\mathrm{wt}(P)<2^{1-r}$. Then under an invertible affine transformation, P can be written as either*

$$P(Y_1,\ldots,Y_{r+t}) = Y_1\cdots Y_{r-t}(Y_{r-t+1}\cdots Y_r + Y_{r+1}\cdots Y_{r+t}) \quad (1)$$

where $3\le t\le r$ and $t+r\le m$, or

$$P(Y_1,\ldots,Y_{r+2t-2}) = Y_1\cdots Y_{r-2}(Y_{r-1}Y_r + Y_{r+1}Y_{r+2} + \cdots + Y_{r+2t-3}Y_{r+2t-2}) \quad (2)$$

where $2\le 2t\le m-r+2$.

COROLLARY 10. *For $RM(r,m)$, $A(2^{1-r}-\varepsilon)\le\varepsilon^{-2(m+1)}$.*

Proof: Consider P of the form of Equation 1. One can show that $\mathrm{wt}(P)=2^{1-r}(1-2^{-t})$. If $\mathrm{wt}(P)<2^{1-r}-\varepsilon$, then $t\le\log(\varepsilon^{-1})-r$. Such P is specified by the choices of the affine forms $Y_1,\ldots,Y_{r+t}$. There are at most 2^{m+1} choices for each Y_i. Hence the total number of polynomials of this form is bounded by

$$N_1=\sum_{t\le\log(\varepsilon^{-1})-r} 2^{(m+1)(r+t)} < 2\varepsilon^{-(m+1)}.$$

Consider P of the form of Equation 2. One can again show that $\mathrm{wt}(P)=2^{1-r}(1-2^{-t})$, hence $t\le\log(\varepsilon^{-1})-r$. The polynomial is specified by the choice of at most $r+2t$ affine forms. Hence the number of such polynomials is bounded by

$$N_2=\sum_{t\le\log(\varepsilon^{-1})-r} 2^{(m+1)(r+2t)} < 2^{1-r(m+1)}\varepsilon^{-2(m+1)}.$$

Thus $A(2^{1-r}-\varepsilon)\le N_1+N_2\le\varepsilon^{-2(m+1)}$. ∎

While r does not appear explicitly on the RHS, the bound is interesting when $\varepsilon<2^{-r}$. We now derive bounds on list size. In what follows, we think of r as a constant, and we want asymptotic bounds in terms of m and ϵ.

THEOREM 11. *For* RM(r,m) *codes, for any $\varepsilon>0$:*

$$\ell(r,m,2^{-r}-\varepsilon)\le\ 2(2^{r+5}\varepsilon^{-2})^{4r}=O(\varepsilon^{-8r}) \quad (3)$$

$$\ell(r,m,J(2^{1-r})-\varepsilon)\le\ (2\varepsilon^2)^{-2m-O(1)} \quad (4)$$

Proof: To prove Equation 3, pick k so that $2^{r+4}\varepsilon^{-2}>2^k\ge 2^{r+3}\varepsilon^{-2}$. By Corollary 6, $\ell(r,m,2^{-r}-\varepsilon)\le 2\cdot\ell(r,k,2^{-r})$, so we focus on bounding $\ell(r,k,2^{-r})$. Set $\alpha=2^{1-r}-2^{-2r+1}$,

so that $J(\alpha) = 2^{-r}$. Applying Corollary 10 with $\varepsilon = 2^{-2r+1}$ gives $A(\alpha) \leq 2 \cdot 2^{(4r-2)(k+1)}$. Applying part 2 of Lemma 8,

$$\ell(r, k, 2^{-r}) \leq A(\alpha)2^{k+1} = 2 \cdot 2^{(4r-1)(k+1)} < 2(2^{r+5}\varepsilon^{-2})^{4r}.$$

To prove Equation 4, we apply part 1 of Lemma 8 with $\alpha = 2^{1-r} - \varepsilon'$, to get

$$\ell(r, m, J(2^{1-r} - \varepsilon')) \;\leq\; A(\alpha)c(\varepsilon') \;=\; \varepsilon'^{-2m-O(1)}.$$

A simple calculation shows that $J(2^{1-r} - \varepsilon') \geq J(2^{1-r}) - \sqrt{\varepsilon'/2}$. Setting $\varepsilon' = 2\varepsilon^2$ gives the desired result. ∎

We show that Equation 3 is tight: at error-rate 2^{-r} the list size is exponential in m, so it is impossible to have a local-decoder running in time $\text{poly}(m)$. Further, the list size at radius $2^{-r} - \varepsilon$ grows as ε^{-r}. In contrast, it is unclear at what distance, the list size goes from $2^{O(m)}$ to $2^{\omega(m)}$.

THEOREM 12. *For* $\text{RM}(r, m)$ *codes:*

$$\ell(r, m, 2^{-r}) \geq 2^{r(m-r)} \tag{5}$$

$$\ell(r, m, 2^{-r} - \varepsilon) \geq \varepsilon^{-r}2^{-(r+r^2)} \tag{6}$$

Proof: Equation 5 is folklore ([24]); take R to be the all 0s vector. The codewords will be the indicator vectors for vector spaces of dimension $m - r$. Fix a vector space $\mathbf{V}_i$ where $\dim(\mathbf{V}_i) = m - r$. Define $P_i(\mathbf{x})$ to be 1 iff $\mathbf{x} \in \mathbf{V}_i$. It is clear that $\Delta(P_i, R) = 2^{-r}$ and that the various P_is are distinct. To show that $\deg(P_i) = r$, take $L_1, \cdots, L_r$ to be the rows of a parity check matrix for $\mathbf{V}_i$. Then

$$P_i(\mathbf{x}) = \prod_{j=1}^{r}(1 + L_j(\mathbf{x})).$$

The list size equals the number of subspaces of $\mathbb{F}_2^m$ of dimension $m - r$, which is at least $2^{r(m-r)}$.

To prove Equation 6, pick the k so that $2\varepsilon \geq 2^{-k} > \varepsilon$. Our codewords and received word will be functions on $\mathbb{F}_2^k$. Take the received word to be 1 at 0^k and 0 elsewhere. As before, we take the polynomials to be the indicators of subspaces of dimension $k - r$. It follows that $\Delta(P, R) = 2^{-r} - 2^{-k} \leq 2^{-r} - \varepsilon$. The list size is lower bounded by $2^{r(k-r)} \geq (2\varepsilon)^{-r}2^{-r^2} = \Omega(\varepsilon^{-r})$. ∎

4.4 Putting It Together

THEOREM 13. *Algorithm 1 solves the Local List-Decoding problem for* $\text{RM}(r, m)$ *codes up to error rate* $\eta = 2^{-r} - \varepsilon$ *in time* $\text{poly}(m^r, \varepsilon^{-r})$.

Proof: We have already proved correctness in Theorem 2, so we only need to bound the runtime. Since $L = O(\varepsilon^{-8r})$ (Equation 3) and $B = O(m^r)$, so the subspace $\mathbf{A}$ chosen in Step 1 is of size $2^k = O((m/\varepsilon)^r)$. In Step 2, we run Algorithm 2 on $\mathbf{A}$, with error-rate $\eta = 2^{-r} - \varepsilon/2$. We can bound the list size as $\mathcal{L}_{\mathbf{A}} \leq O(\varepsilon^{-8r})$ by Equation 3, thus by Theorem 4, this step takes time $\text{poly}(m^r, \varepsilon^{-r})$. In step 3, we run unique decoding $O(m^r)$ times for each polynomial in $\mathcal{L}_{\mathbf{A}}$. Overall the running time is bounded by $\text{poly}(m^r, \varepsilon^{-r})$. ∎

In the Global List-Decoding scenario, Algorithm 2 can receiver codewords up to error rate $J(2^{1-r}) - \varepsilon$ in time polynomial in the block-length (the polynomial depends on ε).

THEOREM 14. *For any* $\varepsilon > 0$, *Algorithm 1 solves the Global List-Decoding problem for* $\text{RM}(r, m)$ *codes up to error rate* $\eta = J(2^{1-r}) - \varepsilon$ *in time* $2^{O(m \log(\varepsilon^{-1}))}$.

Proof: By Equation 4, the list size is bounded by $\ell(r, m, \eta) \leq 2^{O(m \log(\varepsilon^{-1}))}$. By Theorem 4, the running time of Algorithm 2 is polynomial in 2^m and ℓ, which implies the claim. ∎

5. EXTENSION TO SMALL FIELDS

In this section, we consider Reed-Muller codes over arbitrary fields $\mathbb{F}_q$. The field size q is considered a constant. The message space of the code $\text{RM}_q(r, m)$ consists of all polynomials $P : \mathbb{F}_q^m \to \mathbb{F}_q$ of degree at most r. The degree in each variable X_i denoted $\deg_i(P)$ is at most $q - 1$. Thus $\text{RM}_q(r, m)$ is an $[n, k, d]_q$ error-correcting code where $n = q^m$ and k equals the number of monomials in $\mathbb{F}_q[X_1, \ldots, X_m]$ with $\deg(M) \leq r$. Let $d(r, m)$ denote its minimum distance. The following bound (famous as the Schwartz-Zippel Lemma when $r < q$) is proved using induction.

LEMMA 15. *If* $r = a(q - 1) + b$ *where* $a \geq 0$ *and* $1 \leq b \leq q - 1$, *then*

$$\delta_q(r) = \min_m \frac{d(r, m)}{q^m} = \frac{1}{q^a}\left(1 - \frac{b}{q}\right).$$

One can think of $\delta_q(r)$ as the fractional minimum distance for $m \geq \frac{r}{q-1}$. If $m < \frac{r}{q-1}$, there are no polynomials P of degree r with $\deg_i(P) \leq q - 1$.

5.1 On the List-Decoding Radius of $\text{RM}_q(r, m)$

We generalize our arguments for the $\mathbb{F}_2$ case to show the that the list-decoding radius of $\text{RM}_q(r, m)$ codes is at least the unique-decoding radius for $\text{RM}_q(r - 1, m)$ codes. Define $\ell_q(r, m, \eta)$ to be the maximum list size possible for $\text{RM}_q(r, m)$ codes with error-rate η.

THEOREM 16. *For any degree* r *and* $\varepsilon > 0$, *there exists a constant* $c(\varepsilon, q, r)$ *such that*

$$\ell_q(r, m, \frac{\delta_q(r-1)}{2} - \varepsilon) < c(\varepsilon, q, r).$$

As in Theorem 5, we prove this bound by giving an algorithm. The idea behind the algorithm is to reconstruct the polynomial from its values on a random subspace, plus its derivatives along random directions. Similar ideas are used in the recent work of Bogdanov and Viola [5]. For any $\mathbf{b} \in \mathbb{F}_q^m$, define the derivative $P^{(\mathbf{b})}$ of P along $\mathbf{b}$ as

$$P^{(\mathbf{b})}(\mathbf{x}) = P(\mathbf{x} + \mathbf{b}) - P(\mathbf{x}) \tag{7}$$

It is easy to verify that $\deg(P^{(\mathbf{b})}) \leq r - 1$.

ALGORITHM 4. RECOVER$(R, r, \delta_q(r-1)/2 - \varepsilon)$

1. Pick a random k-dimensional affine subspace $\mathbf{A}$ where $q^k > C\varepsilon^{-2}$.
2. Guess the polynomial $P_{\mathbf{A}}$.
3. For each $\mathbf{b} \in \mathbb{F}_q^m$,
   a. Define $S : \mathbf{A} \to \mathbb{F}_q$ as $S(\mathbf{a}) = R(\mathbf{b}+\mathbf{a}) - P_{\mathbf{A}}(\mathbf{a})$.
   b. Run unique decoding on S to get $P^{(\mathbf{b})}$.
   c. Set $P(\mathbf{b}+\mathbf{a}_0) = P^{(\mathbf{b})}(\mathbf{a}_0) + P_{\mathbf{A}}(\mathbf{a}_0)$.
4. Run unique decoding on P and output the result.

In Step 3c, the point $\mathbf{a}_0$ can be an arbitrary point in $\mathbf{A}$. The algorithm tries to guess P on a random subspace $\mathbf{A}$ of constant dimension; assume that this succeeds. Step 3 is a self-correction step where for every $\mathbf{b} \in \mathbb{F}_q^m$, we try to recover $P^{(\mathbf{b})}$ restricted to the subspace $\mathbf{A}$. Note that for $\mathbf{a} \in \mathbf{A}$, $P^{(\mathbf{b})}(\mathbf{a}) = P(\mathbf{a}+\mathbf{b}) - P(\mathbf{a})$ whereas the received word we have is $S(\mathbf{a}) = R(\mathbf{a}+\mathbf{b}) - P_{\mathbf{A}}(\mathbf{a})$. Since we guessed $P_{\mathbf{A}}$ correctly, we have the correct value of P for all $\mathbf{a} \in \mathbf{A}$. The Subspace Sampling Lemma guarantees that (for most $\mathbf{b}$), R and P disagree on less than $\delta_q(r-1)/2$ fraction of points in $\mathbf{b}+\mathbf{A}$. Thus $\Delta_{\mathbf{A}}(S, P^{(\mathbf{b})}) < \delta_q(r-1)/2$. But since $\deg(P^{(\mathbf{b})}) \le r-1$, this is good enough for unique-decoding. Finally by Equation 7, knowing $P(\mathbf{a})$ and $P^{(\mathbf{b})}(\mathbf{a})$ suffices to recover $P(\mathbf{b}+\mathbf{a})$.

LEMMA 17. *Fix P so that $\Delta(P,R) < \delta_q(r-1)/2 - \varepsilon$. Algorithm 4 recovers P with probability at least $\frac{1}{c(\varepsilon,q,r)}$.*

Proof: The total number of degree r polynomials over $\mathbf{A}$ is a constant (depending on ε, q, r). So there is a constant probability of guessing $P_{\mathbf{A}}$ correctly in Step 2. Assume this happens.

By the Subspace Sampling Lemma, for large enough C, the condition $\Delta_{\mathbf{b}+\mathbf{A}}(R,P) < \delta_q(r-1)/2$ holds for more than $1 - \delta_q(r)/2$ fraction of points $\mathbf{b} \in \mathbb{F}_q^m$. For such $\mathbf{b}$, the received word S defined in Step 3a satisfies $\Delta(S, P^{(\mathbf{b})}) < \delta_q(r-1)/2$. Since $\deg(P^{(\mathbf{b})}) \le r-1$, we recover $P^{(\mathbf{b})}$ in Step 3b and self-correction succeeds at $\mathbf{b}$ in Step 3c. Since the fraction of bad $\mathbf{b}$s is bounded by $\delta_q(r)/2$, we recover P in Step 4. ∎

Theorem 16 follows immediately from Lemma 17. By Lemma 15, how this bound compares to the minimum distance of the code depends on the congruence class of r modulo $q-1$.

COROLLARY 18. *Let $r \equiv b \bmod q-1$ where $1 \le b \le q-1$.*

$$\ell_q(r, m, \frac{q-b+1}{2(q-b)}\delta_q(r) - \varepsilon) < c(\varepsilon, q, r) \qquad (8)$$

In particular, if $(q-1)|r$, then $\ell_q(r, m, \delta_q(r) - \varepsilon) < c(\varepsilon, q, r)$.

Proof: From Lemma 15, it follows that $\delta_q(r-1) = \frac{q-b+1}{q-b}\delta(r)$. Plugging this into Theorem 16 gives Equation 8. In particular, when $(q-1)|r$, then $b = q-1$ so $\delta_q(r-1) = 2\delta_q(r)$. ∎

Equation 8 gives a lower bound on the list-decoding radius lying between $\frac{q}{2(q-1)}\delta_q(r)$ and $\delta_q(r)$. It is easy to show that $\delta_q(r)$ is an upper bound on the list-decoding radius (in the local setting). But unlike in the $\mathbb{F}_2$ case, over $\mathbb{F}_q$, the bound given by Equation 8 is not always tight. The Johnson bound implies that the list size can be bounded by a constant as long as $\eta < J_q(\delta_q(r)) - \varepsilon$ where

$$J_q(\delta) \triangleq \frac{q-1}{q}\left(1 - \sqrt{1 - \frac{q}{q-1}\delta}\right). \qquad (9)$$

It is easy to see that $\frac{\delta}{2} < J_q(\delta) < \delta$ for $0 < \delta < \frac{q-1}{q}$. These two bounds are incomparable; the Johnson bound is better for large q and small r, whereas for every $\mathbb{F}_q$, Equation 8 is better for sufficiently large degree r. We conjecture that the true list-decoding radius for $\mathbb{F}_q$ lies close to the minimum distance for all degrees.

CONJECTURE 1. *For the field $\mathbb{F}_q$, and $\varepsilon > 0$, there exists $c(q, \varepsilon, r)$ such that for all m and r,*

$$\ell_q(r, m, \delta_q(r) - \varepsilon) \le c(q, \varepsilon, r).$$

Conjecture 1 holds when $r = 1$ by the Johnson bound and when $(q-1)|r$ by Corollary 18.

5.2 Algorithms for List-Decoding over $\mathbb{F}_q$

We now consider the algorithmic problem in both local and global settings. Like in the $\mathbb{F}_2$ case, our global list-decoder over $\mathbb{F}_q$ runs in time polynomial in the block-length and worst case list size for any error-rate η.

It is easy to convert Algorithm 4 to an efficient list-decoder that works up to radius $\delta_q(r-1)/2$, by taking $|\mathbf{A}| = \text{poly}(m)$ and replacing the guessing step by the global list-decoder. This is not entirely satisfactory since this bound is not tight for all r; however, we do not know the correct bound. Rather, we give a local list-decoder which works at any error rate η where we can upper bound the list size by a constant (independent of m). To make this precise, for any constant ℓ, we define $\eta_q(r, \ell)$ to be the largest error-rate η so that $\ell_q(r, m, \eta) \le \ell$ for all m. It is easy to see that for any $\ell \ge 1$, $\eta_q(r, \ell)$ lies between $\delta_q(r)/2$ and $\delta_q(r)$. Our local list-decoder takes as input ℓ and a lower bound $\eta < \eta_q(r, \ell)$. It solves the problem up to error rate $\eta - \varepsilon$ in time $\text{poly}(\varepsilon^{-1}, \ell, m)$.

5.2.1 A Local List-Decoder for $\text{RM}_q(r, m)$

ALGORITHM 5. LOCAL-DECODE$(R, r, \ell, \eta - \varepsilon)$

1. Pick a hitting set $\mathcal{B}$ by sampling $B = c_1\binom{m}{\le r}\log q$ points from $\mathbb{F}_q^m$.
2. Pick a random k-dimensional affine space $\mathbf{A}$ where $q^k > c_2\frac{\ell^2 B}{\varepsilon^2}$.
3. Run Global-Decode$(\mathbf{A}, R, r, \eta)$ to get list $\mathcal{L}(\mathbf{A})$.
4. For each $Q \in \mathcal{L}(\mathbf{A})$,
   a. For $\mathbf{b} \in \mathcal{B}$,
      Run Global-Decode$(\mathbf{A} \oplus \mathbf{b}, R, r, \eta)$ to get $\mathcal{L}(\mathbf{b})$.
      Choose $Q' \in \mathcal{L}(\mathbf{b})$ such that $Q'_{\mathbf{A}} = Q$.
      Set $P(\mathbf{b}+\mathbf{a}_0) = Q'(\mathbf{b}+\mathbf{a}_0)$.
   b. Interpolate P from the values at $\mathbf{a}_0 + \mathcal{B}$.
   c. If $\Delta(P, R) < \eta - \frac{\varepsilon}{2}$, add P to $\mathcal{L}$.
5. Output the List $\mathcal{L}$.

Our algorithm combines ideas from Algorithm 1 and the List-decoder for Reed-Muller codes over large fields due to

Sudan *et al.* (STV) [27]. Given an affine space $\mathbf{A}$ and a vector $\mathbf{b}$, let $\mathbf{A} \oplus \mathbf{b}$ be the affine space given by $\{\mathbf{a} + \lambda\mathbf{b} | \mathbf{a} \in \mathbf{A}, \lambda \in \mathbb{F}_q\}$. We will use a randomly sampled interpolating set for degree r polynomials. A random subset of $B = c\binom{m}{\leq r} \log q$ points in $\mathbb{F}_q^m$ is an interpolating set for degree-r polynomials over $\mathbb{F}_q$ with high probability.

Our algorithm takes the received word and parameters $r, \ell, \eta < \eta_q(r, \ell)$ and ε as inputs. It picks a random subspace $\mathbf{A}$ of size $O(\ell^2 m^r \varepsilon^{-2})$ as input and tries to recover P from its evaluations over $\mathbf{a}_0 + \mathcal{B}$, for some $\mathbf{a}_0 \in \mathbf{A}$. We run the global List-Decoder on the affine subspace $\mathbf{A} \oplus \mathbf{b}$ (containing $\mathbf{a}_0 + \mathbf{b}$) for every $\mathbf{b} \in \mathcal{B}$. This gives us a list of at most ℓ possibilities for the restriction of P to that subspace. The Subspace Sampling Lemma implies that this list is likely to contain the restriction of every polynomial P satisfying $\Delta(P, R) \leq \eta - \varepsilon$; it could also contain some other polynomials. Now the problem is to assign values to points in $\mathbf{a}_0 + \mathbf{B}$ in a *co-ordinated* manner, so that we choose the correct polynomial from among the ℓ choices at every point.

We solve this co-ordination problem using ideas from the STV algorithm. We run the global list-decoder on $\mathbf{A}$ and choose a polynomial Q from this list, and think of this as $P_{\mathbf{A}}$. We use this as *advice* to make co-ordinated choices for points in $\mathbf{a}_0 + \mathcal{B}$. From each list $\mathcal{L}(\mathbf{b})$, we choose the polynomial whose restriction to $\mathbf{A}$ is Q and use it to assign a value to $\mathbf{a}_0 + \mathbf{b}$. We show that if Q is actually the restriction of some polynomial P with $\Delta(P, R) < \eta - \varepsilon$, then this restriction exists for every $\mathbf{b} \in \mathcal{B}$ and it is unique. This is because the set $\mathcal{B}$ is chosen randomly, so $\mathbf{A}$ is a random k-dimensional affine subspace of $\mathbf{b} + \mathbf{A}$, and any two polynomials from the list of ℓ candidates are unlikely to agree on $\mathbf{A}$. This is similar to the STV algorithm where the value of the polynomial at a point is used as advice to solve a similar co-ordination problem. We defer the analysis to the full version.

5.2.2 A Global List-Decoder for $\mathrm{RM}_q(r, m)$

We give an algorithm that can solve the list-decoding problem up to any error rate η in time polynomial in the block-length and $\ell_q(r, m, \eta)$, extending Algorithm 2 which solves the $\mathbb{F}_2$ case. Along the way, we also solve the unique decoding problem.

Let $\mathbf{A}$ be an m-dimensional affine space $\mathbf{A}$ parametrized by $X_1, \ldots, X_m$. Fix a received word R, and a polynomial P so that $\Delta(P, R) \leq \eta$. We can partition $\mathbf{A}$ into q affine spaces of dimension $m-1$ each based on the value of X_m. Number these spaces $\mathbf{A}^0, \ldots, \mathbf{A}^{q-1}$ where

$$\Delta_{\mathbf{A}^0}(P, R) \leq \cdots \leq \Delta_{\mathbf{A}^{q-1}}(P, R) \tag{10}$$

LEMMA 19. *For each $i \in \{0, \ldots, q-1\}$,*

$$\Delta_{\mathbf{A}^i}(P, R) \leq \frac{q}{q-i} \Delta_{\mathbf{A}}(P, R) \tag{11}$$

Proof: Let $\Delta_{\mathbf{A}}(P, R) = \eta$. Note that

$$\eta = \frac{1}{q} \sum_k \Delta_{\mathbf{A}^k}(P, R).$$

Assume for contradiction that for some $i \in \{0, \ldots, q-1\}$, $\Delta_{\mathbf{A}^i}(P, R) > \frac{q}{q-i}\eta$. Then $\Delta_{\mathbf{A}^j}(P, R) > \frac{q}{q-i}\eta$ for all $j \geq i$, hence $\sum_{k=0}^{q-1} \Delta_{\mathbf{A}^k}(P, R) > q\eta$ which is a contradiction. ∎

We now describe the algorithm. Fix the permutation $a_0, \ldots, a_{q-1}$ of $\mathbb{F}_q$ so that $\mathbf{A}^i = \{\mathbf{x} \in \mathbf{A} \mid x_m = a_i\}$. We can write the polynomial P as

$$P(X_1, \ldots, X_m) = \sum_{j=0}^{q-1} P_j(X_1, \ldots, X_{m-1}) \cdot \prod_{k=0}^{j-1} (X_m - a_k). \tag{12}$$

Our algorithm tries to recover $P_0, \ldots, P_{q-1}$ in that order from the affine spaces $\mathbf{A}^0, \ldots, \mathbf{A}^{q-1}$. As the index i increases from 0 to $q-1$, the error-rate increases. But $\deg(P_i)$ decreases, resulting in an increase in the the list-decoding radius, which compensates for the increased error-rate.

ALGORITHM 6. GLOBAL-DECODE($\mathbf{A}, R, r, \eta$)

1. `For all orderings` $\mathbf{A}^0, \ldots, \mathbf{A}^{q-1}$`,`
2. `For` $i = 0, \ldots, q-1$`,`
3. `Let`
$$R_i = \frac{R_{\mathbf{A}^i} - \sum_{j=0}^{i-1} P_j \cdot \prod_{k=0}^{j-1}(a_i - a_k)}{\prod_{k=0}^{i-1}(a_i - a_k)}.$$
4. `Set` $\mathcal{L}^i =$ `Global-Decode(`$\mathbf{A}^i, R_i, r-i, \frac{q}{q-i}\eta$`).`
5. `Enumerate over all possible` $P_i \in \mathcal{L}^i$`.`
6. `Let` $P(X_1, \ldots, X_m) = \sum_{j=0}^{q-1} P_j(X_1, \ldots, X_{m-1}) \cdot \prod_{k=0}^{j-1}(X_m - a_k)$`.`
7. `If` $\Delta_{\mathbf{A}}(P, R) < \eta$`, add` P `to` $\mathcal{L}$`.`
8. `Return` $\mathcal{L}$`.`

In the base case when $m = 1$, we just use brute-force search. The analysis is in two steps: we first prove that the list $\mathcal{L}$ contains every polynomial close to R, then we bound the running time.

THEOREM 20. *Fix a polynomial P with $\deg(P) = r$ such that $\Delta(P, R) \leq \eta$. Then $P \in \mathcal{L}$.*

Proof: Fix an ordering of subspaces satisfying Equation 11. Write the polynomial P in the form of Equation 12. We will show using induction on m that $P_i \in \mathcal{L}^i$, which will imply that $P \in \mathcal{L}$. The base case when $m = 1$ is trivial.

Note that P restricted to $\mathbf{A}^0$ is just P_0. Note that $\deg(P_0) = r, \dim(\mathbf{A}^0) = m - 1$ and $\Delta_{\mathbf{A}^0}(P_0, R_0) \leq \Delta_{\mathbf{A}}(P, R) \leq \eta$. Hence by the inductive assumption $P_0 \in \mathcal{L}^0$.

Now assume that we have correctly recovered $P_0, \ldots, P_{i-1}$. We compute P restricted to $\mathbf{A}^i$ as

$$\begin{aligned} P_{\mathbf{A}^i} &= P(X_1, \ldots, X_{m-1}, a_i) \\ &= \sum_{j=0}^{i} P_j(X_1, \ldots, X_{m-1}) \cdot \prod_{k=0}^{j-1} (a_i - a_k). \end{aligned}$$

Hence we have

$$P_i = \frac{P_{\mathbf{A}^i} - \sum_{j=0}^{i-1} P_j \cdot \prod_{k=0}^{j-1}(a_i - a_k)}{\prod_{k=0}^{i-1}(a_i - a_k)}$$

$$R_i = \frac{R_{\mathbf{A}^i} - \sum_{j=0}^{i-1} P_j \cdot \prod_{k=0}^{j-1}(a_i - a_k)}{\prod_{k=0}^{i-1}(a_i - a_k)}.$$

By Equation 11, $\Delta_{\mathbf{A}^i}(R_i, P_i) = \Delta_{\mathbf{A}^i}(R, P) \leq \frac{q}{q-i}\eta$. So the polynomial P_i will be one of the polynomials in the list $\mathcal{L}^i$ returned in Step 4. This completes the induction, so the claim is proved. ∎

THEOREM 21. *The running time of Algorithm 6 is bounded by* $(q!)^m q^{2m} \ell_q(r, m, \eta)^{r+q}$.

Proof: We claim that for any r, m and η,

$$\ell_q(r-i, m-1, \frac{q}{q-i}\eta) \leq \ell_q(r, m, \eta). \tag{13}$$

Let R be a received word and $\{P_1, \ldots, P_{\ell'}\}$ a list of $\ell' = \ell_q(r-1, m-1, \frac{q}{q-i}\eta)$ polynomials in $m-1$ variables such that $\Delta(P_i, R) \leq \frac{q}{q-i}\eta$. Let $S(X_m)$ be a univariate polynomial of degree i with i roots so that $\Pr_x[S(x) \neq 0] = \frac{q-i}{q}$. Define

$$R'(X_1, \ldots, X_m) = R(X_1, \cdots, X_{m-1})S(X_m)$$
$$P'_i(X_1, \ldots, X_m) = P_i(X_1, \cdots, X_{m-1})S(X_m).$$

Then it follows that $\Delta(R', P'_i) \leq \eta$ while $\deg(P'_i) = r$. This proves Equation 13. From here, it is easy to bound the running time using a simple recurrence. ∎

Acknowledgements

We would like to thank Anup Rao for his collaboration in the early stages of this work. Thanks to Venkatesan Guruswami and Salil Vadhan for several helpful discussions and pointers to literature. We thank Jaikumar Radhakrishnan for pointing out an error in the earlier proof of Theorem 12, and for suggesting a fix.

6. REFERENCES

[1] A. Akavia, S. Goldwasser, and S. Safra, *Proving hard-core predicates using list decoding*, in Proc. 44^{th} IEEE Symposium on Foundations of Computer Science (FOCS'03), 2003.

[2] N. Alon, T. Kaufman, M. Krivelevich, S. Litsyn, and D. Ron, *Testing Reed-Muller codes*, IEEE Transactions on Information Theory, 51(11) (2005), pp. 4032–4039.

[3] S. Arora and M. Sudan, *Improved low-degree testing and its applications.*, Combinatorica, 23 (2003), pp. 365–426.

[4] M. Blum, M. Luby, and R. Rubinfeld, *Self-testing/correcting with applications to numerical problems*, J. Comput. Syst. Sci., 47(3) (1993), pp. 549–595.

[5] A. Bogdanov and E. Viola, *Pseudorandom bits for polynomials*, in Proc. 48^{th} IEEE Symp. on Foundations of Computer Science (FOCS'07), 2007.

[6] R. Calderbank, A. Gilbert, and M. Strauss, *List decoding of noisy Reed-Muller-like codes*, arXiv:cs/0607098v2 [cs.DS], (2006).

[7] I. Dumer, *Decoding of Reed-Muller codes with polylogarithmic complexity*, in WISICT '04: Proceedings of the winter international synposium on Information and communication technologies, 2004, pp. 1–6.

[8] Z. Dvir and A. Shpilka, *Noisy interpolating sets for low degree polynomials*, in Proc. 23^{rd} IEEE Conference on Computational Complexity, to appear, 2008.

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

[10] O. Goldreich, *Foundations of Cryptography: Basic Tools*, Cambridge University Press, 2000.

[11] O. Goldreich and L. Levin, *A hard-core predicate for all one-way functions.*, in Proc. 21^{st} ACM Symposium on the Theory of Computing, 1989, pp. 25–32.

[12] O. Goldreich, R. Rubinfeld, and M. Sudan, *Learning polynomials with queries: The highly noisy case.*, SIAM J. Discrete Math., 13 (2000), pp. 535–570.

[13] V. Guruswami, *List Decoding of Error-Correcting Codes*, vol. 3282 of Lecture Notes in Computer Science, Springer, 2004.

[14] ———, *Algorithmic Results in List Decoding*, vol. 2 of Foundations and Trends in Theoretical Computer Science, Now Publishers, 2006.

[15] V. Guruswami and M. Sudan, *Improved decoding of Reed-Solomon and Algebraic-Geometric codes.*, IEEE Transactions on Information Theory, 45(6) (1999), pp. 1757–1767.

[16] J. Jackson, *An efficient membership-query algorithm for learning DNF with respect to the uniform distribution*, Journal of Computer and System Sciences, 55 (1997), pp. 414–440.

[17] S. M. Johnson, *A new upper bound for error-correcting codes*, IEEE Transactions on Information Theory, 8 (1962), pp. 203–207.

[18] ———, *Improved asymptotic bounds for error-correcting codes*, IEEE Transactions on Information Theory, 9 (1963), pp. 198–205.

[19] T. Kasami and N. Tokura, *On the weight structure of Reed-Muller codes*, IEEE Transactions on Information Theory, 16(6) (1970), pp. 752–759.

[20] E. Kushilevitz and Y. Mansour, *Learning decision trees using the Fourier spectrum*, SIAM Journal of Computing, 22(6) (1993), pp. 1331–1348.

[21] F. MacWilliams and N. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, 1977.

[22] Y. Mansour, *Learning Boolean functions via the Fourier transform*, Theoretical Advances in Neural Computation and Learning, (1994), pp. 391–424.

[23] R. Pellikaan and X. Wu, *List decoding of q-ary Reed-Muller codes*, IEEE Transactions on Information Theory, 50(4) (2004), pp. 679–682.

[24] A. Samorodnitsky, *Low-degree tests at large distances*, in Proc. 39^{th} ACM Symposium on the Theory of Computing (STOC'07), 2007, pp. 506–515.

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

[26] ———, *List decoding: Algorithms and applications*, SIGACT News, 31 (2000), pp. 16–27.

[27] M. Sudan, L. Trevisan, and S. P. Vadhan, *Pseudorandom generators without the XOR lemma.*, J. Comput. Syst. Sci., 62 (2001), pp. 236–266.

[28] L. Trevisan, *List-decoding using the XOR lemma*, in Proc. 44^{th} IEEE Symposium on Foundations of Computer Science (FOCS'03), 2003, p. 126.

[29] ———, *Some applications of coding theory in computational complexity.*, Quaderni di Matematica, 13 (2004), pp. 347–424.

[30] J. Wozencraft, *List decoding*, Tech. Report 48:90-95, Quarterly Progress Report, Research Laboratory of Electronics, MIT, 1958.