Efficiently decoding Reed-Muller codes from random errors

Ramprasad Saptharishi* Amir Shpilka† Ben Lee Volk*

Abstract

Reed-Muller codes encode an m-variate polynomial of degree r by evaluating it on all points in $\{0,1\}^m$. We denote this code by $RM(m,r)$. The minimal distance of $RM(m,r)$ is 2^{m-r} and so it cannot correct more than half that number of errors in the worst case. For random errors one may hope for a better result.

In this work we give an efficient algorithm (in the block length $n = 2^m$) for decoding random errors in Reed-Muller codes far beyond the minimal distance. Specifically, for low rate codes (of degree $r = o(\sqrt{m})$) we can correct a random set of $(1/2 - o(1))n$ errors with high probability. For high rate codes (of degree $m - r$ for $r = o(\sqrt{m/\log m})$), we can correct roughly $m^{r/2}$ errors.

More generally, for any integer r, our algorithm can correct any error pattern in $RM(m, m - (2r+2))$ for which the same erasure pattern can be corrected in $RM(m, m-(r+1))$. The results above are obtained by applying recent results of Abbe, Shpilka and Wigderson (STOC, 2015), Kumar and Pfister (2015) and Kudekar et al. (2015) regarding the ability of Reed-Muller codes to correct random erasures.

The algorithm is based on solving a carefully defined set of linear equations and thus it is significantly different than other algorithms for decoding Reed-Muller codes that are based on the recursive structure of the code. It can be seen as a more explicit proof of a result of Abbe et al. that shows a reduction from correcting erasures to correcting errors, and it also bares some similarities with the famous Berlekamp-Welch algorithm for decoding Reed-Solomon codes.

*Department of Computer Science, Tel Aviv University, Tel Aviv, Israel, E-mails: ramprasad@cmi.ac.in, benleevolk@gmail.com. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 257575.

†Department of Computer Science, Tel Aviv University, Tel Aviv, Israel, shpilka@post.tau.ac.il. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 257575, and from the Israel Science Foundation (grant number 339/10).

1 Introduction

Consider the following challenge:

> Given the truth table of a polynomial $f(\mathbf{x}) \in \mathbb{F}_2[x_1, \ldots, x_m]$ of degree at most r, in which $1/2 - o(1)$ fraction of the locations were flipped (that is, given the evaluations of f over $\mathbb{F}_2^m$ with nearly half the entries corrupted), recover f efficiently.

If the errors are adversarial, then clearly this task is impossible for any degree bound $r \geq 1$: any two different linear polynomials disagree on half of the domain, so f cannot be recovered even if an adversary flips a quarter of the bits. Hence, we turn to considering *random* sets of errors of size $(1/2 - o(1))2^m$, and we hope to recover f with high probability (in this case, one may also consider the setting where each bit is independently flipped with probability $1/2 - o(1)$. By standard Chernoff bounds, both settings are almost equivalent).

Even in the random model, if every bit was flipped with probability exactly $1/2$, the situation is again hopeless: in this case the input is completely random and carries no information whatsoever about the original polynomial.

It turns out, however, that even a very small relaxation leads to a dramatic improvement in our ability to recover the hidden polynomial: in this paper we prove, among other results, that even at corruption rate $1/2 - o(1)$ and degree bound as large as $o(\sqrt{m})$, we can *efficiently* recover the *unique* polynomial f whose evaluations were corrupted. Note that in the worst case, given a polynomial of such a high degree, an adversary can flip a tiny fraction of the bits — just slightly more than $1/2^{\sqrt{m}}$ — and prevent unique recovery of f, even if we do not require an efficient solution; and yet, in the average case, we can deal with flipping almost half the bits.

Recasting the playful scenario above in a more traditional terminology, this paper deals with similar questions related to recovery of low-degree multivariate polynomials from their *randomly* corrupted evaluations on $\mathbb{F}_2^m$, or in the language of coding theory, we study the problem of decoding *Reed-Muller* codes under random errors in the *binary symmetric channel (BSC)*. We turn to some background and motivation.

1.1 Reed-Muller Codes

Reed-Muller (RM) codes were introduced in 1954, first by Muller [Mul54] and shortly after by Reed [Ree54] who also provided a decoding algorithm. They are among the oldest and simplest codes to construct — the codewords are multivariate polynomials of a given degree, and the encoding function is just their evaluation vectors. In this work we mainly focus on the most basic case where the underlying field is $\mathbb{F} = \mathbb{F}_2$, the field of two elements, although our techniques do generalize to larger finite fields. Over $\mathbb{F}_2$, the Reed-Muller code of degree r in m variables, denoted by $RM(m, r)$, has block length $n = 2^m$, rate $\binom{m}{\leq r}/2^m$ and its minimal distance is 2^{m-r}.

RM codes have been extensively studied with respect to decoding errors in both the worst case and random setting. We begin by giving a review of Reed-Muller codes and their use in theoretical computer science and then discuss our results.

Background

Error-correcting codes (over both large and small finite fields) have been extremely influential in the theory of computation, playing a central role in some important developments in several

areas such as cryptography (e.g. [Sha79] and [BF90]), theory of pseudorandomness (e.g. [BV10]), probabilistic proof systems (e.g. [BFL91, Sha92] and [ALM+98]) and many more.

An important aspect of error correcting codes that received a lot of attention is designing efficient decoding algorithms. The objective is to come up with an algorithm that can correct a certain amounts of errors in a received word. There are two settings in which this problem is studied:

Worst case errors: This is also referred to as errors in the *Hamming model* [Ham50]. Here, the algorithm should recover the original message regardless of the error pattern, as long as there are not too many errors. The number of errors such a decoding algorithm can tolerate is upper bounded in terms of the distance of the code. The *distance* of the code C is the minimum Hamming distance of any two codewords in C. If the distance is d, then one can *uniquely* recover from at most $d-1$ erasures and from $\lfloor (d-1)/2 \rfloor$ errors. For this model of worst-case errors it is easy to prove that Reed-Muller codes perform badly. They have relatively small distance compared to what random codes of the same rate can achieve (and also compared to explicit families of codes).

Another line of work in Hamming's worst case setting concerns designing algorithms that can correct beyond the unique-decoding bound. Here there is no unique answer and so the algorithm returns a list of candidate codewords. In this case the number of errors that the algorithm can tolerate is a parameter of the distance of the code. This question received a lot of attention and among the works in this area we mention the seminal works of Goldreich and Levin on Hadamard Codes [GL89] and of Sudan [Sud97] and Guruswami and Sudan [GS99] on list decoding Reed-Solomon codes. Recently, the list-decoding question for Reed-Muller codes was studied by Gopalan, Klivans and Zuckerman [GKZ08] and by Bhowmick and Lovett [BL15], who proved that the list decoding radius[1] of Reed-Muller codes, over $\mathbb{F}_2$, is at least twice the minimum distance (recall that the unique decoding radius is half that quantity) and is smaller than four times the minimal distance, when the degree of the code is constant.

Random errors: A different setting in which decoding algorithms are studied is Shannon's model of random errors [Sha48]. In Shannon's average-case setting (which we study here), a codeword is subjected to a random corruption, from which recovery should be possible *with high probability*. This random corruption model is called a *channel*. The two most basic ones, the Binary Erasure Channel (BEC) and the Binary Symmetric Channel (BSC), have a parameter p (which may depend on n), and corrupt a message by independently replacing, with probability p, the symbol in each coordinate, with a "lost" symbol in the BEC(p) channel, and with the complementary symbol in the BSC(p) case. In his paper Shannon studied the optimal trade-off achievable for these channels (and many other channels) between the distance and rate. For *every* p, the capacity of BEC(p) is $1-p$, and the capacity of BSC(p) is $1-h(p)$, where h is the binary entropy function.[2] Shannon also proved that random codes achieve this optimal behavior. That is, for every $0 < \varepsilon$ there exist codes of rate $1-h(p)-\varepsilon$ for the BSC (and rate $1-p-\varepsilon$ for the BEC), that can decode from a fraction p of errors (erasures) with high probability.

For our purposes, it is more convenient to assume that the codeword is subjected to a fixed number s of random errors. Note that by the Chernoff-Hoeffding bound, (see e.g., [AS92]), the probability that more than $pn + \omega(\sqrt{pn})$ errors occur in BSC(p) (or BEC(p)) is $o(1)$, and so we can restrict ourselves to the case of a fixed number s of random errors, by setting the corruption probability to be $p = s/n$. We refer to [ASW15] for further discussion on this subject.

[1] The maximum distance η for which the number of code words within distance η is only polynomially large (in n).

[2] $h(p) = -p\log_2(p) - (1-p)\log_2(1-p)$, for $p \in (0,1)$, and $h(0) = h(1) = 0$.

Decoding erasures to decoding errors

Recently, there has been a considerable progress in our understanding of the behavior of Reed-Muller codes under random erasures. In [ASW15], Abbe, Shpilka and Wigderson showed that Reed-Muller codes achieve capacity for the BEC for both sufficiently low and sufficiently high rates. Specifically, they showed that $RM(m,r)$ achieves capacity for the BEC for $r = o(m)$ or $r > m - o(\sqrt{m/\log m})$. More recently, Kumar and Pfister [KP15] and Kudekar, Mondelli, Şaşoğlu and Urbanke [KMŞU15] independently showed that Reed-Muller codes achieve capacity for the BEC in the entire constant rate regime, that is $r \in [m/2 - O(\sqrt{m}), m/2 + O(\sqrt{m})]$. These regimes are pictorially represented in Figure 1.

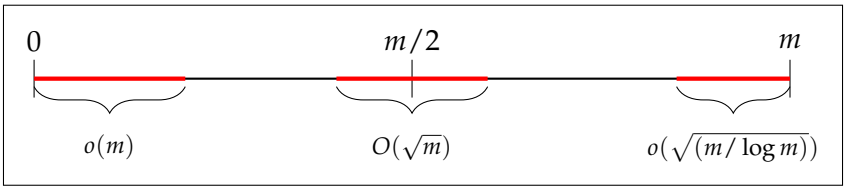

Figure 1: Regime of r for which $RM(m,r)$ is known to achieve capacity for the BEC

Another result proved by Abbe et al. [ASW15] is that Reed-Muller codes $RM(m, m-2r-2)$ can correct any *error pattern* if the same *erasure pattern* can be decoded in $RM(m, m-r-1)$. This reduction is appealing on its own, since it connects decoding from erasures — which is easier in both an intuitive and an algorithmic manner — with decoding from errors; but its importance is further emphasized by the progress made later by Kumar and Pfister and Kudekar et al., who showed that Reed-Muller codes can correct many erasures in the constant rate regime, right up to the channel capacity.

This result show that $RM(m, m-(2r+2))$ can cope with most error patterns of weight $(1-o(1))\binom{m}{\leq r}$, which is the capacity of $RM(m, m-(r+1))$ for the BEC. While this is polynomially smaller than what can be achieved in the Shannon model of errors for random codes of the same rate, this number is still much larger (super-polynomial) than the distance (and the list-decoding radius) of the code, which is 2^{2r+2}. Also, since $RM\left(m, \frac{m}{2} + o(\sqrt{m})\right)$ can cope with $\left(\frac{1}{2} - o(1)\right)$-fraction of erasures, this translation implies that $RM(m, o(\sqrt{m}))$ can handle that many random errors.

However, a shortcoming of the proof of Abbe et al. for the BSC is that it is existential. In particular it does not provide an efficient decoding algorithm. Thus, Abbe et al. left open the question of coming up with a decoding algorithm for Reed-Muller codes from random errors.

1.2 Our contributions

In this work we give an efficient decoding algorithm for Reed-Muller codes that matches the parameters given by Abbe et al. Following the aforementioned results about the erasure correcting ability of Reed-Muller codes, the results can be partitioned into the low-rate and the high-rate regimes. We begin with the result for the low rate case.

Theorem 1 (Low rate, informal)**.** *Let $r < \delta\sqrt{m}$ for a small enough δ. Then, there is an efficient algorithm that can decode $RM(m,r)$ from a random set of $(1-o(1)) \cdot \binom{m}{\leq (m-r)/2}$ errors. In particular, if $r = o(\sqrt{m})$, the algorithm can decode from $\left(\frac{1}{2} - o(1)\right) \cdot 2^m$ errors. The running time of the algorithm is $O(n^4)$ and it can be simulated in* NC.

For high rate Reed-Muller codes, we cannot hope to achieve such a high error correction capability as in the low rate case, even information theoretically. We do give, however, an algorithm that corrects many more errors (a super-polynomially larger number) than what the minimal distance of the code suggests, and its running time is also nearly linear in the block length of the code.

Theorem 2 (High rate, informal)**.** *Let $r = o(\sqrt{m/\log m})$. Then, there is an efficient algorithm that can decode $RM(m, m-(2r+2))$ from a random set of $(1-o(1))\binom{m}{\leq r}$ errors. Moreover, the running time of the algorithm is $2^m \cdot \mathrm{poly}(\binom{m}{\leq r})$ and it can be simulated in* NC*.*

Recall that the block length of the code is $n = 2^m$, and thus the running time is near linear in n when $r = o(m)$.

A general property of our algorithm is that it corrects any error pattern in $RM(m, m-2r-2)$ for which the same *erasure* pattern in $RM(m, m-r-1)$ can be corrected. Stated differently, if an erasure pattern can be corrected in $RM(m, m-r-1)$ then the same pattern, where the "lost" symbol is replaced with arbitrary $0/1$ values, can be corrected in $RM(m, m-(2r+2))$. This property is useful when we know $RM(m, m-r-1)$ can correct a large set of erasures with high probability, that is, when $m-r-1$ falls in the *red region* in Figure 1. Thus, our result has implications also beyond the above two instances. In particular, it may be the case that our algorithm performs well for other rates as well. For example, consider the following question and the theorem it implies.

Question 3. *Does $RM(m, m-r-1)$ achieve capacity for the BEC?*

Theorem 4 (informal)**.** *For any value r for which the answer to Question 3 is positive, there exists an efficient algorithm that decodes $RM(m, m-2r-2)$ from a random set of $(1-o(1))\binom{m}{\leq r}$ errors with probability $(1-o(1))$ (over the random errors). Moreover, the running time of the algorithm is $2^m \cdot \mathrm{poly}\left(\binom{m}{\leq r}\right)$.*

Recall that Abbe et al. [ASW15] also proved that the answer to Question 3 is positive for $r = m - o(m)$ (that is, for $RM(m, o(m))$) but this case does not help us as we need to consider $RM(m, m-(2r+2))$ and $m-(2r+2) < 0$ in this case. The coding theory community seems to believe the answer to Question 3 is positive, for all values of r, and conjectures to that effect were made[3] in [CF07, Arı08, MHU14]. Recent simulations have also suggested that the answer to the question is positive [Arı08, MHU14]. Thus, it seems natural to believe that the answer is positive for most values of r, even for $r = \Theta(m)$. As a conclusion, the belief in the coding theory community suggests that our algorithm can decode a random set of roughly $\binom{m}{\leq r}$ errors in $RM(m, m-(2r+2))$. For example, for $r = \rho \cdot m$, where $\rho < 1/2$, the minimal distance of $RM(m, m-(2r+2))$ is roughly $2^{2\rho m}$ whereas our algorithm can decode from roughly $2^{h(\rho)m}$ random errors (assuming the answer to Question 3 is positive), which is a much larger quantity for every $\rho < 1/2$.

In Section 3, we also present an abstraction of our decoding procedure that may be applicable to other linear codes. This is a generalization of the abstract Berlekamp-Welch decoder or "error-locating pairs" method of Duursma and Kötter [DK94] that connects decodable erasure patterns on a larger code to decodable error patterns. A specific instantiation of this was observed by

[3]The belief that RM codes achieve capacity is much older, but we did not trace back where it appears first.

Abbe et al. [ASW15] by connecting decodable error patterns of any linear code C to decodable erasure patterns of an appropriate "tensor" C' of C (by essentially embedding these codes in a large enough RM code). Although Abbe et al. did not provide an efficient decoding algorithm, the algorithm we present directly applies here (Section 3.2). The abstraction of the "error-locating pairs" method presented in Section 3 should hopefully be applicable in other contexts too, especially considering the generality of the results of [KP15, KMŞU15].

1.3 Related literature

In Section 1.1 we surveyed the known results regarding the ability of Reed-Muller codes to correct random erasures. In this section we summarize the results known about recovering RM codes from random errors.

Once again, it is useful to distinguish between the low rate and the high rate regime of Reed-Muller codes. We shall use d to denote the distance of the code in context. For $RM(m,r)$ codes, $d = 2^{m-r}$.

In [Kri70], the majority logic algorithm of [Ree54] is shown to succeed in recovering all but a vanishing fraction of error patterns of weight up to $d \log d/4$ for all RM codes of positive rate. In [Dum06], Dumer showed for all r such that $\min(r, m-r) = \omega(\log m)$ that most error patterns of weight at most $(d \log d/2) \cdot (1 - \frac{\log m}{\log d})$ can be recovered in $RM(m,r)$. To make sense of the parameters, we note that when $r = m - \omega(\log m)$ the weight is roughly $(d \log d/2)$. To compare this result to ours, we first consider the case when $r = m - o(\sqrt{m/\log m})$. Here the algorithm of [Dum06] can correct roughly $2^{o(\sqrt{m/\log m})}$ random errors in $RM(m,r)$ whereas Theorem 2 gives an algorithm for correcting roughly $m^{o(\sqrt{m/\log m})} \approx (d \log d)^{O(\log m)}$ random errors.

Let us now consider the case $r = \rho m$ for some constant $0 < \rho < 1$. Here, the minimal distance is $2^{(1-\rho)m}$. In this regime, the bound in the above result of [Dum06] is equal to $O(d \log d)$. On the other hand, assuming a positive answer to Question 3, Theorem 4 implies an efficient decoding algorithm for $RM(m, \rho m)$ that can decode from, roughly, $\binom{m}{\leq \frac{1-\rho}{2} m} \approx 2^{h\left(\frac{1-\rho}{2}\right)m}$ random errors, where again h is the binary entropy function.

When ρ is a small constant, we have that $h\left(\frac{1}{2} - \frac{\rho}{2}\right) \approx 1 - \Theta(\rho^2)$, so the number of errors we can correct is roughly $2^{(1-\Theta(\rho^2))m}$, which is a larger quantity than the minimal distance $d = 2^{(1-\rho)m}$, and also than the above $O(d \log d)$ result of [Dum06].

When $\rho = 1 - \varepsilon$ for some small constant $\varepsilon > 0$, $h(\varepsilon/2) = \Omega(\varepsilon \log(1/\varepsilon))$, and the number of errors we can correct is around $2^{\varepsilon \log(1/\varepsilon) m} \approx d^{\log(1/\varepsilon)}$.

As a last example, suppose $\rho = 1/2 - \delta$ for some small constant $\delta > 0$. In this case,

$$h\left(\frac{1-\rho}{2}\right) = h(1/4 + \delta/2) = h(1/4) + \Theta(\delta) = \frac{1}{2} + \frac{3}{4}\log(4/3) + \Theta(\delta),$$

which gives a bound of roughly $2^{(1/2+\eta+\delta)m}$ on the number of errors, for an absolute constant $\eta > 0$, compared to $2^{(1/2+\delta)m}$, which is the minimal distance.

In order to exhibit the fact that $h\left(\frac{1}{2} - \frac{\rho}{2}\right)$ is always larger than $1 - \rho$, we plot the graphs of the two functions side by side in Figure 2. Recall that the former function is the logarithm of the number of errors we can correct, normalized by m, assuming a positive answer to Question 3; and the latter is the logarithm of the minimal distance, normalized by m.

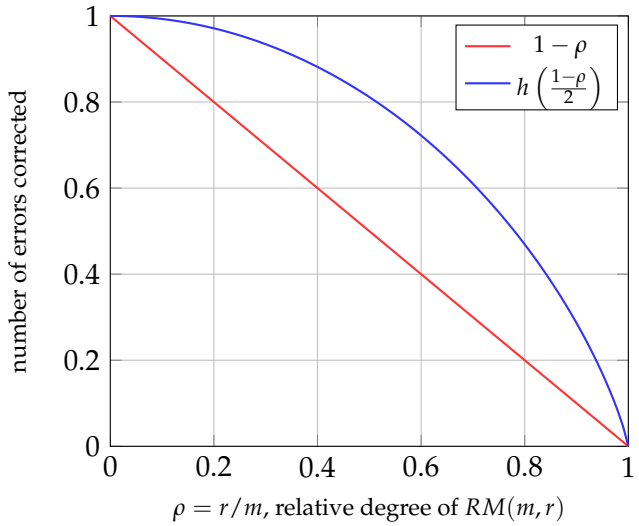

Figure 2: The number of correctable errors as a function of the degree, in logarithmic scale, normalized by m. The red line corresponds to the algorithm given in [Dum06], which corrects $\approx d \log d \approx 2^{(1-\rho)m} \cdot (1-\rho)m$ errors. The blue curve corresponds to the algorithm given in this work, which corrects $\approx 2^{h(\frac{1-\rho}{2})m}$ errors.

We now turn to considering RM codes of low rate. For the special case of $r = 1, 2$, [HKL05] shows that $RM(m,r)$ codes are capacity-achieving. In [SP92], it is shown that RM codes of fixed order (i.e., $r = O(1)$) can decode most error patterns of weight up to $\frac{1}{2}n(1 - \sqrt{c(2^r-1)m^r/nr!})$, where $c > \ln(4)$. In [ASW15], Abbe et al. settled the question for low order Reed-Muller codes proving that $RM(m,r)$ codes achieve capacity for the BSC when $r = o(m)$ [ASW15]. We note however that all the results mentioned here are existential in nature and do not provide an efficient decoding algorithm.

A line of work by Dumer [Dum04, DS06] based on recursive algorithms (that exploit the recursive structure of Reed-Muller codes), obtains algorithmic results mainly for low-rate regimes. In [Dum04], it is shown that for a fixed degree, i.e., $r = O(1)$, an algorithm of complexity $O(n \log n)$ can correct most error patterns of weight up to $n(1/2 - \varepsilon)$ given that ε exceeds $n^{-1/2^r}$. In [Dum06], this is improved to errors of weight up to $\frac{1}{2}n(1 - (4m/d)^{1/2^r})$ for all $r = o(\log m)$. The case $r = \omega(\log m)$ is also covered in [Dum06], as described above.

We note that all the efficient algorithms mentioned above (both for high- and low-rate) rely on the so called Plotkin construction of the code, that is, on its recursive structure (expanding an m-variate polynomial according to the m-th variable $f(x_1, \ldots, x_m) = x_m g(x_1, \ldots, x_{m-1}) + h(x_1, \ldots, x_{m-1})$), whereas our approach is very different.

We summarize and compare our results with [Dum04, DS06, Dum06] for various range of parameters in Figure 3 (degree is r and distance is $d = 2^{m-r}$). The dotted region in Figure 3 corresponds to the uncovered region in Figure 1 beyond $m/2$, via the connection given in Theorem 4. Figure 2 compares our results, for the linear degree regime ($r = \rho m$), with those of [Dum06].

1.4 Notation and terminology

Before explaining the idea behind the proofs of our results we need to introduce some notation and parameters. We shall use the same notation as [ASW15].

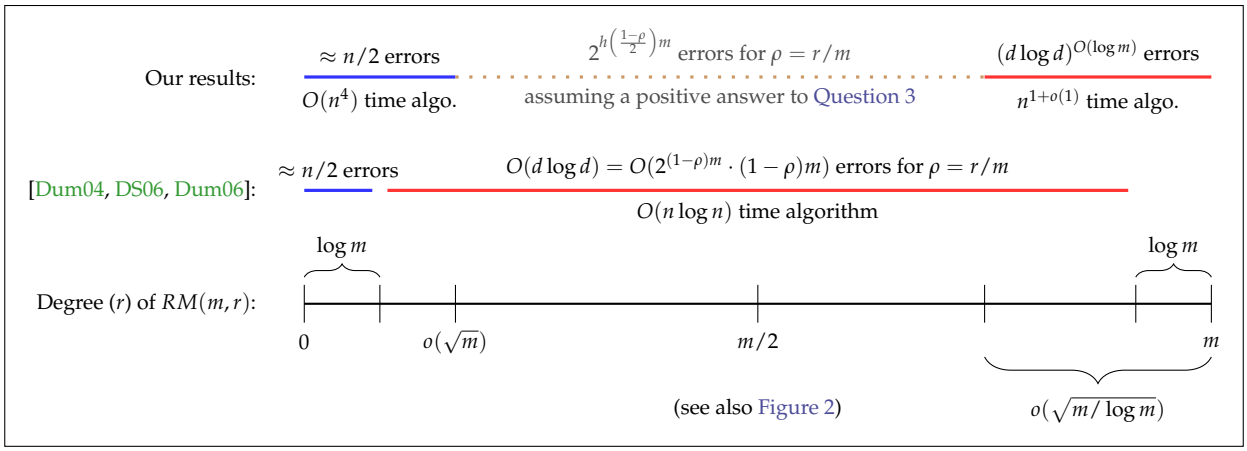

Figure 3: Comparison with [Dum04, DS06, Dum06]

- We denote by $\mathbb{M}(m,r)$ the set of m-variate monomials over $\mathbb{F}_2$ of degree at most r.
- For non-negative integers $r \leq m$, $RM(m,r)$ denotes the Reed-Muller code whose codewords are the evaluation vectors of all multivariate polynomials of degree at most r on m boolean variables. The maximal degree r is sometimes called the order of the code. The block length of the code is $n = 2^m$, the dimension $k = k(m,r) = \sum_{i=0}^{r} \binom{m}{i} \stackrel{\text{def}}{=} \binom{m}{\leq r}$, and the distance $d = d(m,r) = 2^{m-r}$. The code rate is given by $R = k(m,r)/n$.
- We use $E(m,r)$ to denote the "evaluation matrix" of parameters m, r, whose rows are indexed by all monomials in $\mathbb{M}(m,r)$, and whose columns are indexed by all vectors in $\mathbb{F}_2^m$. The value at entry $(M, \mathbf{u})$ is equal to $M(\mathbf{u})$. For $\mathbf{u} \in \mathbb{F}_2^m$, we denote by $\mathbf{u}^r$ the column of $E(m,r)$ indexed by $\mathbf{u}$, which is a k-dimensional vector, consisting of all evaluations of degree $\leq r$ monomials at $\mathbf{u}$. For a subset of columns $U \subseteq \mathbb{F}_2^m$ we denote by U^r the corresponding submatrix of $E(m,r)$.
- $E(m,r)$ is a generator matrix for $RM(m,r)$. The duality property of Reed-Muller codes (see, for example, [MS77]) states that $E(m, m-r-1)$ is a parity-check matrix for $RM(m,r)$, or equivalently, $E(m,r)$ is a parity-check matrix for $RM(m, m-r-1)$.
- We associate with a subset $U \subseteq \mathbb{F}_2^m$ its characteristic vector $\mathbb{1}_U \in \mathbb{F}_2^n$. We often think of the vector $\mathbb{1}_U$ as denoting either an *erasure pattern* or an *error pattern*.
- For a positive integer n, we use the standard notation $[n]$ for the set $\{1, 2, \ldots, n\}$.

We next define what we call the degree-r syndrome of a set.

Definition 5 (Syndrome). *Let $r \leq m$ be two positive integers. The* degree-r syndrome*, or simply r-syndrome of a set $U = \{\mathbf{u}_1, \ldots, \mathbf{u}_t\} \subseteq \mathbb{F}_2^m$ is the $\binom{m}{\leq r}$-dimensional vector α whose entries are indexed by all monomials $M \in \mathbb{M}(m,r)$, such that*

$$\alpha_M \stackrel{\text{def}}{=} \sum_{i=1}^{t} M(\mathbf{u}_i).$$

Note that this is nothing but the syndrome of the error pattern $\mathbb{1}_U \in \mathbb{F}_2^n$ in the code $RM(m, m-r-1)$ (whose parity check matrix is the generator matrix of $RM(m,r)$).

1.5 Proof techniques

In this section we describe our approach for constructing a decoding algorithm. Recall that the algorithm has the property that is decodes in $RM(m, m-2r-2)$ any error pattern U which is correctable from erasures in $RM(m, m-r-1)$. Such patterns are characterized by the property that the columns of $E(m,r)$ corresponding to the elements of U are linearly independent vectors. Thus, it suffices to give an algorithm that succeeds whenever the error pattern $\mathbb{1}_U$ gives rise to such linearly independent columns, which happens with probability $1-o(1)$ for the regime of parameters mentioned in Theorem 1 and Theorem 2.

So let us assume from now on that the error pattern $\mathbb{1}_U$ corresponds to a set of linearly independent columns in $E(m,r)$. Notice that by the choice of our parameters, our task is to recover U from the degree $(2r+1)$-syndrome of U. Furthermore, we want to do so efficiently. For convenience, let $t = |U| = (1-o(1))\binom{m}{\leq r}$.

Recall that the degree-$(2r+1)$ syndrome of U is the $\binom{m}{\leq 2r+1}$-long vector α such that for every monomial $M \in \mathbb{M}(m, 2r+1)$, $\alpha_M = \sum_{i=1}^{t} M(\mathbf{u}_i)$. Imagine now that we could somehow find degree-r polynomials $f_i(x_1, \ldots, x_m)$ satisfying $f_i(u_j) = \delta_{i,j}$. Then, from knowledge of α and, say, f_1, we could compute the following sums:

$$\sigma_\ell = \sum_{i=1}^{t} (f_1 \cdot x_\ell)(\mathbf{u}_i), \quad \ell \in [m].$$

Indeed, if we know α and f_1 then we can compute each σ_ℓ, as it just involves summing several coordinates of α (since $\deg(f_1 \cdot x_\ell) \leq r+1$). We now observe that

$$\sigma_\ell = \sum_{i=1}^{t} (f_1 \cdot x_\ell)(\mathbf{u}_i) = (f_1 \cdot x_\ell)(\mathbf{u}_1) = (\mathbf{u}_1)_\ell.$$

In other words, knowledge of such an f_1 would allow us to discover all coordinates of $\mathbf{u}_1$ and in particular, we will be able to deduce $\mathbf{u}_1$, and similarly all other $\mathbf{u}_i$ using f_i.

Our approach is thus to find such polynomials f_i. What we will do is set up a system of linear equations in the coefficients of an unknown degree r polynomial f and show that f_1 is the unique solution to the system. Indeed, showing that f_1 is a solution is easy and the hard part is proving that it is the unique solution.

To explain how we set the system of equations, let us assume for the time being that we actually know $\mathbf{u}_1$. Let $f = \sum_{M \in \mathbb{M}(m,r)} c_M \cdot M$, where we think of $\{c_M\}$ as unknowns. Consider the following linear system:

1. $\sum_{i=1}^{t} f(\mathbf{u}_i) = f(\mathbf{u}_1) = 1,$
2. $\sum_{i=1}^{t} (f \cdot M)(\mathbf{u}_i) = M(\mathbf{u}_1)$, for all $M \in \mathbb{M}(m,r)$.

In words, we have a system of $2 + \binom{m}{\leq r} + m \cdot \binom{m}{\leq r}$ equations in $\binom{m}{\leq r}$ variables (the coefficients of f). Observe that $f = f_1$ is indeed a solution to the system. Furthermore, a solution to the equations in item 2 gives (the coefficients of) a linear combination of the columns of U_r that equals $\mathbf{u}_1^r$. Since those columns are linearly independent, there is a unique such linear combination, which implies f_1 is the unique solution to the system.

Now we explain what to do when we do not know $\mathbf{u}_1$. Let $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{F}_2^m$. We modify the linear system above to:

1. $\sum_{i=1}^{t} f(\mathbf{u}_i) = f(\mathbf{v}) = 1$,
2. $\sum_{i=1}^{t} (f \cdot M)(\mathbf{u}_i) = M(\mathbf{v})$ for all $M \in \mathbb{M}(m, r)$.
3. $\sum_{i=1}^{t} (f \cdot M \cdot (x_\ell + v_\ell + 1))(\mathbf{u}_i) = M(\mathbf{v})$ for all $\ell \in [m]$ and $M \in \mathbb{M}(m, r)$.

Now the point is that one can prove that if a solution exists then it must be the case that $\mathbf{v}$ is an element of U. Indeed, the set of equations in item 2 implies that $\mathbf{v}^r$ is in the linear span of the columns of U^r. The linear equations in item 3 then imply that $\mathbf{v}$ must actually be in the set U.

Notice that what we actually do amounts to setting, for every $\mathbf{v} \in \mathbb{F}_2^m$, a system of linear equations of size roughly $\binom{m}{\leq r}$. Such a system can be solved in time poly $\left(\binom{m}{\leq r}\right)$. Thus, when we go over all $\mathbf{v} \in \mathbb{F}_2^m$ we get a running time of $2^m \cdot$ poly $\left(\binom{m}{\leq r}\right)$, as claimed.

Our proof can be viewed as an algorithmic version of the proof of Theorem 1.8 of Abbe et al. [ASW15]. That theorem asserts that when the columns of U^r are linearly independent, the $(2r+1)$-syndrome of U is unique. In their proof of the theorem they first use the $(2r)$-syndrome to claim that if V is another set with the same $(2r)$-syndrome then the column span of U^r is the same as that of V^r. Then, using the degree $(2r+1)$ monomials they deduce that $U = V$. This is similar to what our linear system does, but, in contrast, [ASW15] did not have an efficient algorithmic version of this statement.

2 Decoding Algorithm For Reed-Muller Codes

We begin with the following basic linear algebraic fact.

Lemma 6. *Let $\mathbf{u}_1, \ldots, \mathbf{u}_t \in \mathbb{F}_2^m$ such that $\{\mathbf{u}_1^r, \ldots, \mathbf{u}_t^r\}$ are linearly independent. Then, for every $i \in [t]$, there exists a polynomial f_i so that for every $j \in [t]$,*

$$f_i(\mathbf{u}_j) = \delta_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

For completeness, we give the short proof.

Proof. Consider the matrix $U^r \in \mathbb{F}_2^{t \times \binom{m}{\leq r}}$ whose i-th row is $\mathbf{u}_i^r$. A polynomial f_i which satisfies the properties of the lemma is a solution to the linear system $U^r \mathbf{x} = \mathbf{e}_i$, where $\mathbf{e}_i \in \mathbb{F}_2^t$ is the i-th elementary basis vector (that is, $(\mathbf{e}_i)_j = \delta_{i,j}$), and the $\binom{m}{\leq r}$ unknowns are the coefficients of f_i. By the assumption that U is of full rank, indeed there exists a solution. □

The algorithm would proceed by making a guess $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{F}_2^m$ for one of the error locations. If we could come up with an efficient way to *verify* that the guess is correct, this would immediately yield a decoding algorithm. We shall verify our guess by using the dual polynomials

$f_1, \ldots, f_t$ described above. We shall find them by solving a system of linear equations that can be constructed from the $(2r+1)$-syndrome of $\{\mathbf{u}_1, \ldots, \mathbf{u}_m\}$. We will need the following crucial, yet simple, observation.

Observation 7. *Let f be any m-variate polynomial of degree at most $2r+1$, and $\mathbf{u}_1, \ldots, \mathbf{u}_t \in \mathbb{F}_2^m$. Then, the sum $\sum_{i=1}^t f(\mathbf{u}_i)$ can be computed given the $(2r+1)$-syndrome of $\{\mathbf{u}_1, \ldots, \mathbf{u}_t\}$, in time $O\left(\binom{m}{2r+1}\right)$.*

Proof. For any $M \in \mathbb{M}(m, 2r+1)$, denote $\alpha_M = \sum_{i=1}^t M(\mathbf{u}_i)$ (so that $\alpha = (\alpha_M)_{M \in \mathbb{M}(M,2r+1)}$ is precisely the syndrome of $\{\mathbf{u}_1, \ldots, \mathbf{u}_t\}$). Write $f = \sum_{M \in \mathbb{M}(m,2r+1)} c_M \cdot M$, where $c_M \in \mathbb{F}_2$, then

$$\begin{aligned}\sum_{i=1}^t f(\mathbf{u}_i) &= \sum_{i=1}^t \sum_{M \in \mathbb{M}(m,2r+1)} c_M \cdot M(\mathbf{u}_i) \\ &= \sum_{M \in \mathbb{M}(m,2r+1)} c_M \left(\sum_{i=1}^t M(\mathbf{u}_i)\right) = \sum_{M \in \mathbb{M}(m,2r+1)} c_M \alpha_M. \qquad \square\end{aligned}$$

The following lemma shows how to verify a guess for an error location. It is the main ingredient in the analysis of our algorithm and the reason why it works. Basically, the lemma gives a system of linear equations whose solution enables us to decide whether a given $\mathbf{v} \in \mathbb{F}_2^m$ is a corrupted coordinate or not, without knowledge of the set of errors U but only of its syndrome. In a sense, this lemma is analogous to the Berlekamp-Welch algorithm, which also gives a system of linear equations whose solution reveals the set of erroneous locations ([WB86], and see also the exposition in Chapter 13 of [GRS14]).

Lemma 8 (Main Lemma)**.** *Let $\mathbf{u}_1, \ldots, \mathbf{u}_t \in \mathbb{F}_2^m$ such that $\{\mathbf{u}_1^r, \ldots, \mathbf{u}_t^r\}$ are linearly independent, and $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{F}_2^m$. Suppose there exists a multilinear polynomial $f \in \mathbb{F}_2[x_1, \ldots, x_m]$ with $\deg(f) \leq r$ such that for every monomial $M \in \mathbb{M}(m, r)$,*

1. $\sum_{i=1}^t f(\mathbf{u}_i) = f(\mathbf{v}) = 1$,
2. $\sum_{i=1}^t (f \cdot M)(\mathbf{u}_i) = M(\mathbf{v})$, *and*
3. $\sum_{i=1}^t (f \cdot M \cdot (x_\ell + v_\ell + 1))(\mathbf{u}_i) = M(\mathbf{v})$ *for every $\ell \in [m]$.*

Then there exists $i \in [t]$ such that $\mathbf{v} = \mathbf{u}_i$.

Observe that if indeed $\mathbf{v} = \mathbf{u}_i$ for some $i \in [t]$, then the polynomial f_i guaranteed by Lemma 6 satisfies those equations. Hence, the lemma should be interpreted as saying the converse: that if there exists such a solution, then $\mathbf{v} = \mathbf{u}_i$ for some i. Further, given the $(2r+1)$-syndrome of $\{\mathbf{u}_1, \ldots, \mathbf{u}_t\}$ as input, Observation 7 shows that each of the above constraints are linear constraints in the coefficients of f. Thus, finding such an f is merely solving a system of $O\left(\binom{m}{\leq r}\right)$ linear equations in $\binom{m}{\leq r}$ unknowns and can be done in poly $\left(\binom{m}{\leq r}\right)$ time.

Proof of Lemma 8. Let $J = \{j \mid f(\mathbf{u}_j) = 1\}$. Note that by item 1 it holds that $J \neq \emptyset$.

Subclaim 9. $\sum_{i \in J} \mathbf{u}_i^r = \mathbf{v}^r$.

Proof. Let $M \in \mathbb{M}(m, r)$. We show that $\sum_{i \in J} M(\mathbf{u}_i) = M(\mathbf{v})$, i.e., that the M'th coordinate of $\sum_{i \in J} \mathbf{u}_i^r$ is equal to that of $\mathbf{v}^r$. Indeed, as f satisfies the constraints in item 2,

$$M(\mathbf{v}) = \sum_{i=1}^{t} (f \cdot M)(\mathbf{u}_i) = \sum_{i \in J} (f \cdot M)(\mathbf{u}_i) + \sum_{i \notin J} (f \cdot M)(\mathbf{u}_i) = \sum_{i \in J} M(\mathbf{u}_i). \qquad \square \text{(Subclaim)}$$

For any $\ell \in [m]$, let $J_\ell = \{j \mid f(\mathbf{u}_j) = 1 \text{ and } (\mathbf{u}_j)_\ell = v_\ell\} \subseteq J$. Observe that this definition implies that for every $j \in [t]$, the index j is in J_ℓ if and only if $(f \cdot (x_\ell + v_\ell + 1))(\mathbf{u}_j) = 1$. Using a similar argument, we can show the following.

Subclaim 10. *For every* $\ell \in [m]$,

$$\sum_{i \in J_\ell} \mathbf{u}_i^r = \mathbf{v}^r. \tag{11}$$

Proof. Again, for any $M \in \mathbb{M}(m, r)$ the constraints in item 3 imply that

$$M(v) = \sum_{i=1}^{t} (f \cdot M \cdot (x_\ell + v_\ell + 1))(\mathbf{u}_i) = \sum_{i \in J_\ell} M(\mathbf{u}_i). \qquad \square \text{(Subclaim)}$$

From the above claims,

$$\mathbf{v}^r = \sum_{i \in J} \mathbf{u}_i^r = \sum_{i \in J_1} \mathbf{u}_i^r = \cdots = \sum_{i \in J_m} \mathbf{u}_i^r.$$

By the linear independence of $\{\mathbf{u}_1^r, \ldots, \mathbf{u}_t^r\}$, it follows that $J = J_1 = J_2 = \cdots = J_m$. Indeed, there is a unique linear combination of $\{\mathbf{u}_1^r, \ldots, \mathbf{u}_t^r\}$ that gives $\mathbf{v}^r$. The only vector which can be in the (non-empty) intersection $\bigcap_{k=1}^{m} J_k$ is $\mathbf{v}$, and so there exists $i \in [t]$ so that $\mathbf{u}_i = \mathbf{v}$. $\square$

Lemma 8 implies a natural algorithm for decoding from t errors indexed by vectors $\{\mathbf{u}_1, \ldots, \mathbf{u}_t\}$, assuming $\{\mathbf{u}_1^r, \ldots, \mathbf{u}_t^r\}$ are linearly independent, that we write down explicitly in Algorithm 1.

Theorem 12. *Given the* $(2r+1)$*-syndrome of* t *unknown vectors* $\{\mathbf{u}_1, \ldots, \mathbf{u}_t\} \subseteq \mathbb{F}_2^m$ *such that* $\{\mathbf{u}_1^r, \ldots, \mathbf{u}_t^r\}$ *are linearly independent, Algorithm 1 outputs* $\{\mathbf{u}_1, \ldots, \mathbf{u}_t\}$*, runs in time* $2^m \cdot \mathrm{poly}(\binom{m}{\leq r})$ *and can be realized using a circuit of depth* $\mathrm{poly}(m) = \mathrm{poly}(\log n)$.

Proof. The algorithm enumerates all vectors in $\mathbb{F}_2^m$, and for each candidate $\mathbf{v}$ checks whether there exists a solution to the linear system of $\mathrm{poly}(\binom{m}{\leq r})$ equations in $\mathrm{poly}(\binom{m}{\leq r})$ unknowns given in Lemma 8. Observation 7 shows that this system of linear equations can be constructed from the $(2r+1)$-syndrome in $\mathrm{poly}(\binom{m}{\leq r})$ time.

By Lemma 6 and Lemma 8, a solution to this system exists if and only if there is $i \in [t]$ so that $\mathbf{v} = \mathbf{u}_i$. The bound on the running time follows from the description of the algorithm. Furthermore, all $2^m = n$ linear systems can be solved in parallel, and each linear system can be solved with an NC^2 circuit (see, e.g., [MV97]). $\square$

Algorithm 1 : Reed-Muller Decoding

Input: A $(2r+1)$-syndrome of $\{\mathbf{u}_1, \ldots, \mathbf{u}_t\}$

$\mathcal{E} = \emptyset$

for all $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{F}_2^m$ **do**

  Solve for a polynomial $f \in \mathbb{F}_2[x_1, \ldots, x_m]$ of degree at most r:

- $\sum_{i=1}^{t} f(\mathbf{u}_i) = f(\mathbf{v}) = 1$,
- $\sum_{i=1}^{t} (f \cdot M)(\mathbf{u}_i) = M(\mathbf{v})$ for all $M \in \mathbb{M}(m, r)$.
- $\sum_{i=1}^{t} (f \cdot M \cdot (x_\ell + v_\ell + 1))(\mathbf{u}_i) = M(\mathbf{v})$ for all $\ell \in [m]$ and $M \in \mathbb{M}(m, r)$.

  if there is a polynomial f that satisfies the above system of equations **then**

    Add $\mathbf{v}$ to the set $\mathcal{E}$.

return the set $\mathcal{E}$ as the error locations.

Observe that the the proof of correctness for Algorithm 1 is valid, for any value of r, whenever the set of error locations $\{\mathbf{u}_1, \ldots, \mathbf{u}_t\}$ satisfies the property that $\{\mathbf{u}_1^r, \ldots, \mathbf{u}_t^r\}$ are linearly independent. Therefore, we would like to apply Theorem 12 in settings where $\{\mathbf{u}_1^r, \ldots, \mathbf{u}_t^r\}$ are linearly independent with high probability.

For the constant rate regime, Kumar and Pfister [KP15] and Kudekar, Mondelli, Şaşoğlu and Urbanke [KMŞU15] proved that $RM(m, m-r-1)$ achieves capacity for $r = m/2 \pm O(\sqrt{m})$.

Theorem 13 ([KP15], Theorem 23)**.** *Let $r \leq m$ be integers such that $r = m/2 \pm O(\sqrt{m})$. Then, for $t = (1 - o(1))\binom{m}{\leq r}$, with probability $1 - o(1)$, for a set of vectors $\{\mathbf{u}_1, \ldots, \mathbf{u}_t\} \subseteq \mathbb{F}_2^m$ chosen uniformly at random, it holds that $\{\mathbf{u}_1^r, \ldots, \mathbf{u}_t^r\}$ are linearly independent over $\mathbb{F}_2^{\binom{m}{\leq r}}$.*

Letting $r = m/2 - o(\sqrt{m})$ and looking at the code $RM(m, m-2r-2) = RM(m, o(\sqrt{m}))$ so that $\binom{m}{\leq r} = (1/2 - o(1))2^m$, we get the following statement, stated earlier as Theorem 1.

Corollary 14. *There exists a (deterministic) algorithm that is able to correct $t = (1/2 - o(1))2^m$ random errors in $RM(m, o(\sqrt{m}))$ with probability $1 - o(1)$. The algorithm runs in time $2^m \cdot \left(\binom{m}{m/2 - o(\sqrt{m})}\right)^3 \leq n^4$.*

Alternatively, we can pick $r = m/2 - O(\sqrt{m})$ and correct $c \cdot 2^m$ random errors in the code $RM(m, O(\sqrt{m}))$, where c is some positive constant that goes to zero as the constant hidden under the big O increases.

For the high-rate regime, recall the following capacity achieving result proved in [ASW15]:

Theorem 15 ([ASW15], Theorem 4.5)**.** *Let $\varepsilon > 0$, $r \leq m$ be two positive integers and $t < \binom{m - \log(\binom{m}{\leq r}) - \log(1/\varepsilon)}{\leq r}$. Then, with probability at least $1 - \varepsilon$, for a set of vectors $\{\mathbf{u}_1, \ldots, \mathbf{u}_t\} \subseteq \mathbb{F}_2^m$ chosen uniformly at random, it holds that $\{\mathbf{u}_1^r, \ldots, \mathbf{u}_t^r\}$ are linearly independent over $\mathbb{F}_2^{\binom{m}{\leq r}}$.*

Using Theorem 15, we apply Theorem 12 to obtain the following corollary, which was stated informally as Theorem 2.

Corollary 16. *Let $\varepsilon > 0$, and $r \leq m$ be two positive integers. Then there exists a (deterministic) algorithm that is able to correct $t = \left\lfloor \binom{m-\log(\binom{m}{\leq r})-\log(1/\varepsilon)}{\leq r} \right\rfloor - 1$ random errors in $RM(m, m-(2r+2))$ with probability at least $1-\varepsilon$. The algorithm runs in time $2^m \cdot \text{poly}\left(\binom{m}{\leq r}\right)$.*

If $r = o(\sqrt{m/\log m})$, the bound on t is $(1-o(1))\binom{m}{\leq r}$, as promised.

More generally, a positive answer to Question 3 is equivalent to $\{\mathbf{u}_1^r, \ldots, \mathbf{u}_t^r\}$ for $t = (1-o(1))\binom{m}{\leq r}$ being linearly independent with probability $1-o(1)$ (see Corollary 2.9 in [ASW15]), and thus we also obtain the following corollary, which was stated informally as Theorem 4.

Corollary 17. *Let $r \leq m$ be two positive integers. Suppose that $RM(m, m-r-1)$ achieves capacity for the BEC. Then there exists a (deterministic) algorithm that is able to correct $(1-o(1))\binom{m}{\leq r}$ random errors in $RM(m, m-(2r+2))$ with probability $1-o(1)$. The algorithm runs in time $2^m \cdot \text{poly}\left(\binom{m}{\leq r}\right)$.*

We note that for all values of r, $2^m \cdot \text{poly}\left(\binom{m}{\leq r}\right)$ is polynomial in the block length $n = 2^m$, and when $r = o(m)$ this is equal to $n^{1+o(1)}$.

3 Abstractions and Generalizations

3.1 An abstract view of the decoding algorithm

In this section we present a more abstract view of Algorithm 1, in the spirit of the works by Pellikaan, Duursma and Kötter ([Pel92, DK94]) which abstract the Berlekamp-Welch algorithm (see also the exposition in [Sud01]). Stated in this way, it is also clear that the algorithm works also over larger alphabets, so we no longer limit ourselves to dealing with binary alphabets. As shown in [KP15], Reed-Muller codes over $\mathbb{F}_q$ (sometimes referred to as *Generalized Reed-Muller codes)* also achieve capacity in the constant rate regime.

We begin by giving the definition of a (pointwise) product of two vectors, and of two codes.

Definition 18. *Let $\mathbf{u}, \mathbf{v} \in \mathbb{F}_q^n$. Denote by $\mathbf{u} * \mathbf{v} \in \mathbb{F}_q^n$ the vector $(\mathbf{u}_1\mathbf{v}_1, \ldots, \mathbf{u}_n\mathbf{v}_n)$. For $A, B \subseteq \mathbb{F}_q^n$ we similarly define $A * B = \{\mathbf{u} * \mathbf{v} \mid \mathbf{u} \in A, \mathbf{v} \in B\}$.*

Following the footsteps of Algorithm 1, we wish to decode, in a code C, error patterns which are correctable from erasures in a related code N, through the use of an *error-locating code* E. Under some assumptions on C, N and E, we can use a similar proof in order to do this.

Theorem 19. *Let $E, C, N \subseteq \mathbb{F}_q^n$ be codes with the following properties.*

1. *$E * C \subseteq N$*
2. *For any pattern $\mathbb{1}_U$ that is correctable from erasures in N, and for any coordinate $i \notin U$ there exists a codeword $\mathbf{e} \in E$ such that $\mathbf{e}_j = 0$ for all $j \in U$ and $\mathbf{e}_i = 1$.*

Then there exists an efficient algorithm that corrects in C any pattern $\mathbb{1}_U$, which is correctable from erasures in N.

To put things in perspective, earlier we set $C = RM(m, m-2r-2)$, $N = RM(m, m-r-1)$ and $E = RM(m, r+1)$. It is immediate to observe that item 1 holds in this case, and item 2 is guaranteed by Lemma 6: Indeed, consider the error pattern $U = \{\mathbf{u}_1, \ldots, \mathbf{u}_t\}$ and the dual polynomials $\{f_i\}_{i=1}^t$, and let $\mathbf{v} \notin U$ be any other coordinate of the code. If there exists $j \in [t]$ such that $f_j(\mathbf{v}) = 1$, we can pick the codeword $g = f_j \cdot (1 + x_\ell + \mathbf{v}_\ell)$, where ℓ is some coordinate such that $\mathbf{v}_\ell \neq (\mathbf{u}_j)_\ell$. g has degree at most $r+1$ and so it is a codeword in E, and it can be directly verified that it satisfies the conditions of item 2. If $f_j(\mathbf{v}) = 0$ for all j, we can pick $g = 1 - \sum_{i=1}^t f_i$.

It is also worth pointing out the differences between our approach and the abstract Berlekamp-Welch decoder of Duursma and Kötter: They similarly set up codes E, C and N such that $E * C \subseteq N$. However, instead of item 2, they require that for any $\mathbf{e} \in E$ and $\mathbf{c} \in C$, if $\mathbf{e} * \mathbf{c} = 0$ then $\mathbf{e} = 0$ or $\mathbf{c} = 0$ (or similar requirements regarding the distances of E and C that guarantee this property). This property, as well as the distance properties, do not hold in the case of Reed-Muller codes.

Turning back to the proof of Theorem 19, the algorithm and the proof of correctness turn out to be very short to describe in this level of generality. Given a word $\mathbf{y} \in \mathbb{F}_q^n$, the algorithm would solve the the linear system $\mathbf{a} * \mathbf{y} = \mathbf{b}$, in unknowns $\mathbf{a} \in E$ and $\mathbf{b} \in N$. Under the hypothesis of the theorem, we show that common zeros of the possible solutions for $\mathbf{a}$ determine exactly the error locations. Once the locations of the errors are identified, correcting them is easy: we can replace the error locations by the symbol '?' and use an algorithm which corrects erasures (this can always be done efficiently, when unique decoding is possible, as this merely amounts to solving a system of linear equations). The algorithm is given in Algorithm 2.

Algorithm 2 : Abstract Decoding Algorithm

Input: received word $\mathbf{y} \in \mathbb{F}_q^n$ such that $\mathbf{y} = \mathbf{c} + \mathbf{e}$, with $\mathbf{c} \in C$ and $\mathbf{e}$ is supported on a set U

1: Solve for $\mathbf{a} \in E, \mathbf{b} \in N$, the linear system $\mathbf{a} * \mathbf{y} = \mathbf{b}$.
2: Let $\{\mathbf{a}_1, \ldots, \mathbf{a}_k\}$ be a basis for the solution space of $\mathbf{a}$, and let $\mathcal{E}$ denote the common zeros of $\{\mathbf{a}_i \mid i \in [k]\}$.
3: For every $j \in \mathcal{E}$, replace $\mathbf{y}_j$ with '?', to get a new word $\mathbf{y}'$.
4: Correct $\mathbf{y}'$ from erasures in C.

Note that in Theorem 19 we assume that the error pattern U is correctable from erasures in N, whereas Algorithm 2 first computes a set of error locations $\mathcal{E}$ and then corrects $\mathbf{y}'$ from erasures in C. Thus, the proof of Theorem 19 can be divided into two steps. The first, and the main one, will be to show that $\mathcal{E} = U$. The second, which is merely an immediate observation, will be to show that U is also correctable from erasures in C. We begin with the second part:

Lemma 20. *Assume the setup of Theorem 19, and let U be any pattern which is correctable from erasures in N. Then U is also correctable from erasures in C.*

Proof. We may assume that $U \neq \emptyset$, as otherwise the statement is trivial. Suppose on the contrary that U is not correctable from erasures in C, that is, there exists a non-zero codeword $\mathbf{c} \in C$ supported on U. For any $\mathbf{a} \in E$, we have that $\mathbf{a} * \mathbf{c}$ is a codeword of N which is supported on a subset of U. In order to reach a contradiction, we want to pick $\mathbf{a} \in E$ so that $\mathbf{a} * \mathbf{c}$ is a non-zero codeword of N, which contradicts the assumption that U is correctable from erasures in N.

Pick $i \in U$ so that $\mathbf{c}_i \neq 0$. Observe that if U is correctable from erasures in N then so is $U \setminus \{i\}$. By item 2 in Theorem 19 with respect to the set $U \setminus \{i\}$ there exists $\mathbf{a} \in E$ with $\mathbf{a}_i = 1$. Thus, in particular $\mathbf{a} * \mathbf{c}$ is non-zero. □

We now prove that main part of Theorem 19, that is, that under the assumptions stated in the theorem, Algorithm 2 correctly decodes (in C) any error pattern that is correctable from erasures in N.

Proof of Theorem 19. Write $\mathbf{y} = \mathbf{c} + \mathbf{e}$, so that $\mathbf{c} \in C$ is the transmitted codeword and $\mathbf{e}$ is supported on the set of error locations U. As noted above, by Lemma 20 it is enough to show that under the assumptions of the theorem (in particular, that U is correctable from erasures in N), the set of error locations $\mathcal{E}$ computed by Algorithm 2 equals U.

In the following two lemmas, we argue that any solution $\mathbf{a}$ for the system vanishes on the error points, and then that for every other index i, there exists a solution whose i-th entry is non-zero (and so there must be a basis element for the solution space whose i-th entry is non-zero).

The following lemma states that every solution $\mathbf{a} \in E$ to the equation $\mathbf{a} * \mathbf{y} = \mathbf{b}$ vanishes on U, the support of $\mathbf{e}$. In the pointwise product notation, this is equivalent to showing that $\mathbf{a} * \mathbf{e} = 0$.

> **Subclaim 21.** *For every $\mathbf{a} \in E, \mathbf{b} \in N$ such that $\mathbf{a} * \mathbf{y} = \mathbf{b}$, it holds that $\mathbf{a} * \mathbf{e} = 0$.*
>
> *Proof.* Since $\mathbf{a} * \mathbf{y} = \mathbf{b} \in N$ (by the assumption) and $\mathbf{a} * \mathbf{c} \in N$ (by item 1), we get that $\mathbf{a} * \mathbf{e} = \mathbf{a} * \mathbf{y} - \mathbf{a} * \mathbf{c}$ is also a codeword in N. Furthermore, $\mathbf{a} * \mathbf{e}$ is also supported on U, and since U is an erasure-correctable pattern in N, the only codeword that is supported on U is the zero codeword. □ (Subclaim)

To finish the proof, we show that for any $i \notin U$, there is a solution $\mathbf{a}$ to the system of linear equations with $\mathbf{a}_i = 1$.

> **Subclaim 22.** *For every $i \notin U$ there exists $\mathbf{a} \in E, \mathbf{b} \in N$ such that $\mathbf{a}$ is 0 on U, $\mathbf{a}_i = 1$ and $\mathbf{a} * \mathbf{y} = \mathbf{b}$.*
>
> *Proof.* By item 2, since U is correctable from erasures in N, for every $i \notin U$ we can pick $\mathbf{a} \in E$ such that $\mathbf{a}$ is 0 on U and $\mathbf{a}_i = 1$. Set $\mathbf{b} = \mathbf{a} * \mathbf{y}$. It remains to be shown that $\mathbf{b}$ is a codeword of N. This follows from the fact that
>
> $$\mathbf{b} = \mathbf{a} * \mathbf{c} + \mathbf{a} * \mathbf{e} = \mathbf{a} * \mathbf{c},$$
>
> where the second equality follows from the fact that $\mathbf{a}$ is zero on U (the support of $\mathbf{e}$). Finally, $\mathbf{a} * \mathbf{c}$ is a codeword of N by item 1. □ (Subclaim)

These two claims complete the proof of the theorem. □

3.2 Decoding of Linear Codes over $\mathbb{F}_2$

In [ASW15], it is observed that their results for Reed-Muller codes imply that for *every* linear code N, every pattern which is correctable from erasures in N is correctable from errors in what they call the "degree-three tensoring" of N. One can in fact use our Algorithm 1 almost verbatim to obtain an efficient version of this statement. However, here we remark that this is nothing but a special case of Theorem 19 with an appropriate setting of the codes E, C, N. We begin by briefly describing their definitions and their argument.

The basic tool used by [ASW15] is embedding any parity check matrix in the matrix $E(m, 1)$ for an appropriate choice of m. Let N be any linear code of dimension k over $\mathbb{F}_2$ and H be its parity check matrix. For convenience, we first extend N by adding a parity bit. This increases the block

length by 1, does not decrease the distance and preserves the dimension. A parity check matrix for the extended code can by obtained from H by constructing the matrix

$$H_0 = \begin{pmatrix} 1 & 1 \cdots 1 \\ 0 & \\ \vdots & H \\ 0 & \end{pmatrix}.$$

The main observation now is that $E(m,1)$ is an $(m+1) \times 2^m$ matrix that contains all vectors of the form $(1, \mathbf{v})$ for $\mathbf{v} \in \mathbb{F}_2^m$, so if we set $m = n - k$ to be the number of rows of H, we can pick a subset S of the columns of $E(m,1)$ that correspond to the columns that appear in H_0.

[ASW15] then define the degree-three tensoring of N, which is a code C whose parity check matrix is $H_0^{\otimes 3}$: this is an $\binom{m}{\leq 3} \times n$ matrix with rows indexed by tuples $i_1 < i_2 < i_3$, with the corresponding row being the pointwise product (as in Definition 18) of rows i_1, i_2, i_3 of H_0. One can then verify that Algorithm 1 can be used in order to correct (in C) any error pattern which is correctable from erasures in N, by using the algorithm with $r = 1$ and having the error location guesses run only over the columns in S.

A closer look reveals that this construction is in fact a special case of Theorem 19. Given any linear binary code N with parity check matrix H, the main observation of [ASW15] can be interpreted as saying that when we add a parity bit to N, we can embed N in a puncturing of $RM(m, m-2)$ (whose parity check matrix is $E(m,1)$). We state it in the following claim:

Claim 23. *Let N' denote the subcode of $RM(m, m-2)$ of all words that are 0 outside S. Then N is precisely the restriction of N' to the S coordinates.*

Proof. Let $\mathbf{b} \in N$. Then $H_0\mathbf{b} = 0$, i.e. the columns of H_0 indexed by the non-zero elements in $\mathbf{b}$ add up to 0. Let $\mathbf{b}' \in \mathbb{F}_2^{2^m}$ denote that extension of $\mathbf{b}$ into a vector of length 2^m obtained by filling 0's in every coordinate not in S. Then $E(m,1)\mathbf{b}' = 0$, since the same columns that appeared in H_0 appear in $E(m,1)$. This implies that $\mathbf{b}' \in N'$.

Similarly, for every $\mathbf{b}' \in N'$, we can define $\mathbf{b}$ to be its restriction to S, and then $H_0\mathbf{b} = 0$, i.e. $\mathbf{b} \in N$. □

The degree-three tensoring of N, which we denote by C, can then be similarly embedded in a puncturing of $RM(m, m-4)$, where again, only the coordinates in S remain, and similarly C can be seen to be the restriction to S to the subcode C' of $RM(m, m-4)$ that contains the words that are 0 outside S.

Finally, we define the error locating code E to be the restriction of $RM(m,2)$ to the coordinates of S.

We now show that the conditions of Theorem 19 are satisfied in this case. We begin with item 2. If U is a correctable pattern in N, it means that the columns indexed by U in H_0 are linearly independent. It follows that they are also linearly independent as columns in $E(m,1)$. Hence, using the same arguments as before we can find, for any coordinate $\mathbf{v} \notin U$, a degree 2 polynomial g such that $g(\mathbf{v}) = 1$ and g restricted to U is 0. Restricting the evaluations of g to the subset of coordinates S, we get a codeword $\mathbf{e} \in E$ with the required property.

As for item 1: We first argue that $RM(m,2) * C' \subseteq N'$, since the degrees match and the property of vanishing outside S is preserved under multiplication. Projecting back to the coordinates in S, we get that $E * C \subseteq N$.

Acknowledgement

We would like thank Avi Wigderson, Emmanuel Abbe and Ilya Dumer for helpful discussions and for commenting on an earlier version of the paper. We thank Venkatesan Guruswami and anonymous reviewers for pointing out the abstraction of Algorithm 1 given in Section 3.

References

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof Verification and the Hardness of Approximation Problems. *J. ACM*, 45(3):501–555, 1998.

[Arı08] Erdal Arıkan. A Performance Comparison of Polar Codes and Reed-Muller Codes. *IEEE Communications Letters*, 12(6):447–449, 2008.

[AS92] Noga Alon and Joel H. Spencer. *The Probabilistic Method*. John Wiley, 1992.

[ASW15] Emmanuel Abbe, Amir Shpilka, and Avi Wigderson. Reed-Muller Codes for Random Erasures and Errors. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC 2015)*, pages 297–306, 2015. Pre-print available at `arXiv:1411.4590`.

[BF90] Donald Beaver and Joan Feigenbaum. Hiding instances in multioracle queries. In *Proceedings of the 7th Symposium on Theoretical Aspects of Computer Science (STACS 1990)*, pages 37–48, 1990.

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. Non-Deterministic Exponential Time has Two-Prover Interactive Protocols. *Computational Complexity*, 1:3–40, 1991. Preliminary version in the *31st Annual IEEE Symposium on Foundations of Computer Science (FOCS 1990)*.

[BL15] Abhishek Bhowmick and Shachar Lovett. The List Decoding Radius of Reed-Muller Codes over Small Fields. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC 2015)*, pages 277–285, 2015. Pre-print available at `eccc:TR14-087`.

[BV10] Andrej Bogdanov and Emanuele Viola. Pseudorandom Bits for Polynomials. *SIAM J. Comput.*, 39(6):2464–2486, 2010. Preliminary version in the *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007)*. Pre-print available at `eccc:TR14-081`.

[CF07] Daniel J. Costello, Jr. and G. David Forney, Jr. Channel coding: The road to channel capacity. *Proceedings of the IEEE*, 95(6):1150–1177, 2007.

[DK94] Iwan M. Duursma and Ralf Kötter. Error-locating pairs for cyclic codes. *IEEE Transactions on Information Theory*, 40(4):1108–1121, 1994.

[DS06] Ilya Dumer and Kirill Shabunov. Recursive error correction for general Reed-Muller codes. *Discrete Applied Mathematics*, 154(2):253–269, 2006.

[Dum04] Ilya Dumer. Recursive decoding and its performance for low-rate Reed-Muller codes. *IEEE Transactions on Information Theory*, 50(5):811–823, 2004.

[Dum06] Ilya Dumer. Soft-decision decoding of Reed-Muller codes: a simplified algorithm. *IEEE Transactions on Information Theory*, 52(3):954–963, 2006.

[GKZ08] Parikshit Gopalan, Adam R. Klivans, and David Zuckerman. List-decoding Reed-Muller codes over small fields. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC 2008)*, pages 265–274, 2008.

[GL89] Oded Goldreich and Leonid A. Levin. A Hard-Core Predicate for all One-Way Functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC 1989)*, pages 25–32, 1989.

[GRS14] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. *Essential Coding Theory*. 2014. Available at `http://www.cse.buffalo.edu/faculty/atri/courses/coding-theory/book/`.

[GS99] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

[Ham50] R. W. Hamming. Error Detecting and Error Correcting Codes. *Bell System Technical Journal*, 26(2):147–160, 1950.

[HKL05] Tor Helleseth, Torleiv Kløve, and Vladimir I. Levenshtein. Error-correction capability of binary linear codes. *IEEE Transactions on Information Theory*, 51(4):1408–1423, 2005.

[KMŞU15] Shrinivas Kudekar, Marco Mondelli, Eren Şaşoğlu, and Rüdiger L. Urbanke. Reed-Muller Codes Achieve Capacity on the Binary Erasure Channel under MAP Decoding. *CoRR*, abs/1505.05831, 2015.

[KP15] Santhosh Kumar and Henry D. Pfister. Reed-Muller Codes Achieve Capacity on Erasure Channels. *CoRR*, abs/1505.05123, 2015.

[Kri70] R. E. Krichevskiy. On the number of Reed-Muller code correctable errors. *Dokl. Sov. Acad. Sci.*, 191:541–547, 1970.

[MHU14] Marco Mondelli, Seyed Hamed Hassani, and Rüdiger L. Urbanke. From Polar to Reed-Muller Codes: A Technique to Improve the Finite-Length Performance. *IEEE Transactions on Communications*, 62(9):3084–3091, 2014.

[MS77] F. J. MacWilliams and N. J. A. Sloane. *The theory of error correcting codes*. Number v. 2 in North-Holland mathematical library. North-Holland Publishing Company, 1977.

[Mul54] D. E. Muller. Application of Boolean algebra to switching circuit design and to error detection. *Electronic Computers, Transactions of the I.R.E. Professional Group on*, EC-3(3):6–12, Sept 1954.

[MV97] Meena Mahajan and V. Vinay. Determinant: Combinatorics, Algorithms, and Complexity. *Chicago J. Theor. Comput. Sci.*, 1997.

[Pel92] Ruud Pellikaan. On decoding by error location and dependent sets of error positions. *Discrete Mathematics*, 106-107:369–381, 1992.

[Ree54] Irving S. Reed. A class of multiple-error-correcting codes and the decoding scheme. *Trans. of the IRE Professional Group on Information Theory (TIT)*, 4:38–49, 1954.

[Sha48] C. E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Sha92] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.

[SP92] V. M. Sidel'nikov and A. S. Pershakov. Decoding of Reed-Muller Codes with a Large Number of Errors. *Problems Inform. Transmission*, 28(3):80–94, 1992.

[Sud97] Madhu Sudan. Decoding of Reed Solomon Codes beyond the Error-Correction Bound. *J. Complexity*, 13(1):180–193, 1997.

[Sud01] Madhu Sudan. Algorithmic Introduction to Coding Theory, 2001. Lecture Notes, available at `http://people.csail.mit.edu/madhu/FT02/scribe/lect11.pdf`.

[WB86] Lloyd R. Welch and Elwyn R. Berlekamp. Error correction for algebraic block codes, 1986. US Patent 4,633,470.