2022

Results on the Generalized Covering Radius of Error Correcting Codes

Benjamin Langton

Results on the Generalized Covering Radius of Error Correcting Codes

Ben Langton

Mohamed Omar , Advisor

Michael Orrison, Reader

Department of Mathematics

May, 2022

Abstract

The generalized covering radius is an interesting property of error correcting codes that was recently proposed by Elimelech et al. (3). Because the work is very recent, little is known about the generalized covering radius, particularly on its bounds for most error correcting codes. This thesis summarizes the requisite background necessary to understand the literature surrounding the generalized covering radius. We then go on to establish a new, highly general bound on the covering radius of all codes which satisfy the chain condition, and compare it asymptotically to previously known bounds on Reed-Muller codes. This new bound produces an efficient algorithm which we show can be used with Reed-Muller codes. The new algorithm can also be used to modify the covering algorithm in (4), vastly improving the running time while having an unknown effect on performance.

Contents

List of Figures

List of Tables

Acknowledgments

Thank you to my advisor, Prof. Omar, for introducing me to this topic, guiding me throughout the year, and providing much detailed and helpful feedback on my drafts. Thank you to my second reader, Prof. Orrison, for further feedback on my draft, and also for showing me how exciting math could be in my first few years at Mudd. I would also like to thank Prof. Raviv, for taking a risk on me and introducing me to the world of math research, and all of my class mates for making this such a great experience.

Chapter 1

Introduction

Richard Hamming pioneered the field of error correcting codes in 1950s and today they are used in almost every piece of digital technology. The genius of Hamming's codes is that they allow errors in messages to be detected and also corrected—a novel concept at the time of their conception.

To briefly illustrate how error correcting codes work, suppose we are given an ordered triple of three numbers, (a, b, c) and we want to encode them in such a way that someone else can deduce the original numbers from the encoded version even if they are altered slightly. How might we do this?

Perhaps the simplest way is to repeat the numbers over and over again. For instance, we can simply repeat each number three times to get the encoding $(a, a, a, b, b, b, c, c, c)$. If any one element of this encoding is altered by noise we can still extract the original message.

$$abc \rightarrow aaabbbccc \rightarrow baabbbcac \rightarrow abc$$

Figure 1.1 Example of encoding and decoding a message using an error correcting code

Our "decoding algorithm" for this code would be to find the most frequently occurring character every three letters. The code would be considered 1-error correcting since it is guaranteed to correct any one error. This code *can* correct up to three errors, but unfortunate errors can lead to our decoding algorithm decoding the message incorrectly. For instance, if our encoding gets corrupted to $(a, z, z, b, b, b, c, c, c)$, then we will decode it as (z, b, c), despite there only being two errors introduced.

Furthermore, while the construction of this code is quite simple, we had to lengthen our message from three characters to nine to guarantee that any one error is correctable. This is quite inefficient, and in fact we can do much better.

One way is to consider our numbers as coefficients of a polynomial, $p(x) = ax^2 + bx + c$. Then to encode our message we can evaluate the polynomial at some predetermined points—one example is the encoding $(p(0), p(1), p(2), p(3), p(4), p(5), p(6))$. Because any quadratic polynomial is uniquely determined by three points with different x-values, we've actually evaluated the polynomial four more times than necessary to be able to recover a, b, and c. The added redundancy is what allows us to correct errors that are introduced. In fact, this scheme is significantly better than the previous one. How so? Well, with our new encoding we can find any two errors that are introduced. Let's say two of our coordinates change, so that the altered message is $(p(0) + s, p(1) + t, p(2), p(3), p(4), p(5), p(6))$, for $s, t \neq 0$. Our original polynomial still passes through five of the points, while any other polynomial can pass through at most four of the points. Thus, anyone can still recover the original polynomial $ax^2 + bx + c$ (and thus a, b, c) by finding the polynomial which passes through five or more points in the message. This encoding scheme is able to correct any two errors that occur while only using seven characters to encode the message—much better than the first construction which could only correct one error while using nine characters in the length of the encoding.

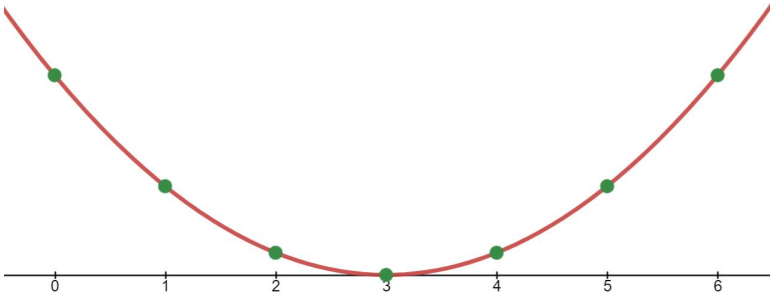

Figure 1.2 Original polynomial is evaluated at green points to encode it

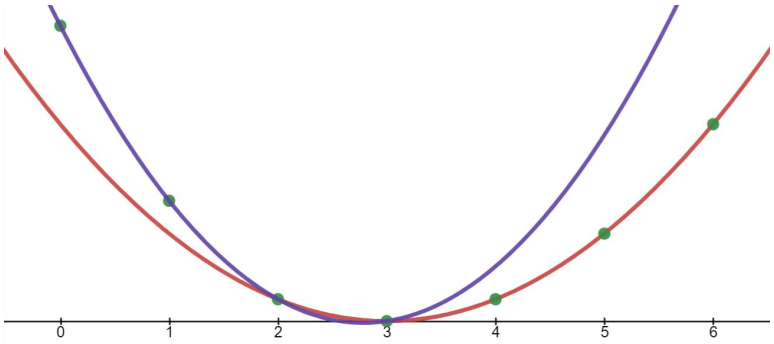

Figure 1.3 Even with two errors introduced, only the original polynomial can pass through at least five points

These two examples give a small window into the world of error correcting codes—a world that we will formalize and explore in depth in the following chapters.

Once we have defined and explored the fundamentals of error correcting codes, we will dive into the core of the thesis—the generalized covering radius and its properties.

Chapter 2

Preliminaries

2.1 What is a code?

In this chapter we will go through the basic definitions surrounding error correcting codes and afterwards the theory of generalized Hamming weights. To start, let us define a code, the basic object we will be working with throughout the entire document. The definitions and theorems in this section can be found in MacWilliams and Sloane (9).

Definition 1. *Let Q be a finite set with q elements. A nonempty subset C of Q^n is a q-ary code of length n.*

We call Q the alphabet, elements of C codewords, and elements of Q^n words or vectors. Although Q need not be a field, in this document it nearly always will be.

Fundamental to the study of codes is the concept of distance between words.

Definition 2. *The Hamming distance between two vectors u and v, denoted $d(u, v)$, is defined to be the number of coordinates in which they differ.*

This distance is conveniently a metric on the space Q^n—that is, for all $u, v, w \in Q^n$:

$$d(u, v) \geq 0 \text{ and } d(x, y) = 0 \text{ iff } x = y \tag{2.1}$$

$$d(u, v) = d(v, u) \tag{2.2}$$

$$d(u, v) \leq d(u, w) + d(w, v) \tag{2.3}$$

(2.1) and (2.2) come straight from the definition of the Hamming distance, and the proof of (2.3), while nontrivial, is also fairly intuitive and left to the reader.

If Q is a group, then we can further define the (Hamming) weight of a word which is closely related to the distance between words:

Definition 3. *The weight of a vector $v \in Q^n$, denoted $h(v)$ is the Hamming distance between v and 0 , i.e $h(v) = d(v, 0)$. In other words, it is the number of nonzero coordinates of v.*

An interesting and useful property of the Hamming distance is that it is translation invariant, that is, $d(u, v) = d(u - w, v - w)\ \forall\ w \in Q^n$. This is because if we translate both u and v by the vector w, they will still differ in the same coordinates, and remain equal in the same coordinates as well. Since $h(v) = d(v, 0)$, then we see that $d(u, v) = d(u - v, v - v) = h(u - v)$—that is, the distance between two vectors is also the weight of their difference. This will also be useful when working with Hamming distances in later sections.

We can further define the minimum distance of a code:

Definition 4. *The minimum distance of a code C is the smallest distance between any two codewords, that is,*

$$d(C) = \min\{d(u, v)) \mid u, v \in C, u \neq v\}.$$

We can also define the weight of a code, $h(C)$:

Definition 5. *The weight of a code C, $h(C)$, is the smallest weight of any nonzero codeword, that is,*

$$h(C) = \min\{h(c) \mid u \in C, u \neq 0\}.$$

2.1.1 The Covering Radius

The covering radius is a concept that lies at the core of this thesis. It has been extensively studied and there is a large body of work on finding bounds on the covering radius of various codes (2; 5; 7; 8; 9). In this section we spend some time discussing the covering radius and some of it's most general bounds. To understand this idea, we first turn to the concept of a Hamming ball.

Definition 6. *The Hamming ball of radius r centered at a vector $x \in Q^n$, is defined to be $B_r(x) = \{u \in Q^n : d(u, x) \leq r\}$.*

We can then ask, for fixed r, whether radius r balls centered at codewords of a code C cover the entirety of Q^n. The covering radius of the code C is the smallest number r such that this is the case. Alternatively:

Definition 7. *The covering radius of C is*

$$R(C) = \max_{u \in Q^n} d(u, C) = \max_{u \in Q^n} \min_{v \in C} d(u, v).$$

We can see from this definition that every vector must be less than distance $R(C)$ away from some codeword in C, and so every vector is contained in an $R(C)$-radius Hamming ball centered at a codeword.

These Hamming balls allow us to compute a general upper bound on the size of codes with a set covering radius and length. We derive the bound by seeing that if the codewords of a code are spaced "optimally" so that the space of all words could be covered by disjoint balls, then each vector would be contained in exactly one Hamming ball of radius $R(C)$. The volume of each Hamming ball is

$$V_q(n, r) = \sum_{i=0}^{r} \binom{n}{i} (q-1)^i.$$

This is a combinatorial formula. At each distance i, each vector is different in i places ($\binom{n}{i}$ different total combinations), and there are $q-1$ ways each position can be different.

Hence, since every vector in Q^n must be contained in at least one Hamming ball of radius $R(C)$ centered at a codeword, we achieve a lower bound on the covering radius of the code based on the volume of the balls and the size of the alphabet q. Equality in this lower bound is achieved when every word is contained in *exactly* one ball of radius $R(C)$.

Theorem 1. *Let C be a code with covering radius R and length n. Then $|C| \geq q^n / V_q(n, R)$.*

Proof. Since R is the covering radius, we know every vector in Q^n lies in some Hamming Ball of radius R centered at a codeword. There are q^n such vectors, $|C|$ codewords, and $V_q(n, R)$ vectors in each Hamming ball. The total volume of all Hamming balls is then at most $|C| \cdot V_q(n, R)$, which is achieved only when all of the balls are disjoint. Otherwise the balls must intersect and the true volume is smaller. We know there are exactly q^n distinct vectors in the balls (since they cover Q^n), so $|C| \cdot V_q(n, R) \geq q^n$. Thus, $|C| \geq q^n / V_q(n, R)$. □

Codes for which this code satisfies equality are called perfect codes. Notice that while it is not so easy to express in a closed form, this also puts a lower bound on covering radius for any code of a given size.

2.2 Linear Codes

While the field of error correcting codes is broad and contains many subfields, the most commonly studied codes are linear codes. The key feature of linear codes is that their codewords live in a linear subspace of some vector space. Put more formally:

Definition 8. *Let Q be an n-dimensional vector space over some finite field. Then any k-dimensional subspace of $\mathbb{F}_q^n$ is an $[n, k]$ linear code.*

Since any arbitrary linear code C is a vector subspace of $\mathbb{F}_q^n$, we can find a basis of k linearly independent codewords $c_1, c_2, ..., c_k$ which span C. The matrix $k \times n$ matrix $G = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{bmatrix}$ is the generator matrix of C. G can be thought of as a linear transformation that maps vectors from $\mathbb{F}_q^k$ to C.

Additionally, the subspace C has an orthogonal complement $C^\perp$ which is called the dual code of C. The generator matrix H of $C^\perp$ is also called the parity-check matrix of C. Notice that for any codeword $c \in C$, $Hc = 0$ which is an important property in many decoding algorithms. Additionally, notice that because the rows of G and H are bases for C and $C^\perp$ (by definition), then G and H are both full rank, and $\begin{bmatrix} G \\ H \end{bmatrix}$ is invertible.

Computing the minimum distance between all codewords of linear codes is important for this thesis, and computationally it's not feasible to do so by brute-force computing all pairwise distances between codewords. However, with the following theorem we don't have to:

Theorem 2. *Let C be a linear code. Then the minimum weight and minimum distance of C are equal.*

Proof. Let C be a linear code, $h(C)$ its minimum weight, and $d(C)$ its minimum distance. We suppose for the sake of contradiction that $h(C) \neq d(C)$. Clearly since $h(C) = d(u, 0)$ for some $u \in C$, then $d(C) < h(C)$. Now, we choose $u, v \in C$ such that $d(u, v) = d(C)$. We see that $d(u, v) = d(u - v, 0) =$

$h(u-v)$. Since C is linear, $u-v \in C$, so $h(u-v) < h(C)$, a contradiction. Thus $h(C) = d(C)$. □

Let's quickly run through an example: Suppose our vector space is $\mathbb{F}_2^6$, and our code C will be the subspace $\langle [111000], [000111] \rangle$, which makes C a $[6, 2]$ linear code. There are only four codewords of C:

$$[000000], [111000], [000111], [111111]$$

and it has generator matrix $G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$. We see that the four codewords can also be written as $[00]G$, $[10]G$, $[01]G$, and $[11]G$. The parity check matrix is the 4×6 matrix

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

We can easily see that the minimum weight/distance is 3. But what's the covering radius? In this code we see that we can choose the first three and last three coordinates independently and still get a codeword. The only constraint is that all three must be the same in each half of the vector. Thus, since we're in $\mathbb{F}_2$, any arbitrary vector must have either two 0s or two 1s in the first three coordinates, and two 0s or 1s in the second three coordinates, so we can always choose a codeword that will differ by 1 in the first three coordinates and 1 in the second three coordinates. Thus the covering radius is 2. However, notice that if we were in a field with more than two elements (but keeping the generator matrix the same), then the covering radius would be 4, because we could only match one coordinate from each of the first/second three. So while minimum distance isn't affected by q, the covering radius is!

2.3 Error Correcting Codes

Codes have the nice property that it is possible to correct a certain number of errors that can be introduced to codewords in a variety of applications. Formally, if a code has an error correcting capability e, then if we change any codeword in e places, we can still recover the original codeword. In the language of linear codes, this allows us to recover a codeword c from a given word $c + v$, provided $h(v) \leq e$.

In reality, c is not known when the message is received, so we perform what is called nearest neighbor decoding, which means that we simply find the codeword that is closest to the received word to decode the message. This means that if an error is introduced that pushes the message closer to a different codeword, we can no longer correctly decode the message. This establishes a relationship between the error correcting capability and minimum distance of a code.

Theorem 3. *If a code C can correct t errors, then $2t + 1 \leq d(C)$.*

Proof. Let c be the sent codeword, u be that codeword with t errors introduced, and c' be another arbitrary codeword. We want to prove that u is closer to c than c'. We see that by the triangle inequality $2t + 1 \leq d(C) \leq d(c, c') \leq d(c, u) + d(u, c') = t + d(u, c')$. So $t + 1 \leq d(u, c')$. Thus $d(u, c) \leq d(u, c')$, and so c is still the closest codeword to u. Thus we can decode any t errors that are introduced. □

The error correcting capability of a code is the maximum number of errors that can be corrected, or $\lfloor d(C)/2 \rfloor$.

To actually decode something, we are given a message u which we know can be written as $c + v$. We can find the error v by finding the vector of least weight in the coset $C - u$, which is the set of all possible error vectors. This vector is called the elected coset leader. This can be done somewhat efficiently by computing the syndrome of u, Hu. We see that $Hu = H(c + v) = Hv$. The parity check matrix is not invertible, and in reality it maps all elements of cosets of C to the same syndrome. This is ideal since we are trying to find the error vector, so we can decode any linear code by creating a bijection between syndromes and error vectors. This of course can get very difficult as the size of the code and hence number of possible error vectors grows.

2.4 Generalized Hamming Weights

Generalized Hamming weights (GHWs) were introduced in the namesake paper by Wei (11) to analyze the algebraic and cryptographic properties of codes. They have proven to be extremely useful in a number of applications, and we will see that one of them is in the analysis of the generalized covering radius.

First, the support of a vector is the set of coordinates where it is nonzero. Accordingly, the support of a code C is the set of coordinates where codewords of C are nonzero. More formally:

Definition 9. $\text{supp}(C) = \{i : \exists(x_1, ..., x_n) \in C, x_i \neq 0\}$.

With this, we can define the r-th generalized Hamming weight of a code, $d_r(C)$.

Definition 10. (11) $d_r(C) = \{\min |\text{supp}(D)| : D \textit{ is an r-dimensional subcode of } C\}$.

We see that $d_1(C)$ is just the weight of a code, hence the name generalized Hamming Weights. Another way we can define the generalized Hamming weight comes from the idea of the support of a matrix. Similar to the support of a code or a vector, we define the support of a matrix to be the set of nonzero columns of the matrix. Then the generalized Hamming weight can also be defined as the following:

Definition 11. $d_r(C) = \{\min |\text{supp}(X)| : X \textit{ is a rank r matrix with codewords as rows } \}$.

The generalized Hamming weights allow us to define a property that certain codes have called the chain condition.

Definition 12. (12) *Let C be an $[n, k]$ linear code with GHWs $d_1(C), d_2(C), ..., d_k(C)$. Then C satisfies the chain condition if there are k linearly independent vectors $c_1, c_2, ...c_k$ such that $d_r(C) = |\bigcup_{i=1}^{r} \text{supp}(c_i)|$.*

A large number of codes satisfy the chain condition, such as Reed-Muller codes, Hamming codes, MDS codes, and the extended Golay code.

Generalized Hamming weights can be related to the covering radius of any code that satisfies the chain condition using the following theorem:

Theorem 4. (7) *Let C be an $[n, k]$ code that satisfies the chain condition with GHWs $d_1, d_2, ..., d_k$. Then the covering radius of C, $R(C)$ satisfies the following bound*:

$$R(C) \leq n - \sum_{r=1}^{k} \left\lceil \frac{d_r - d_{r-1}}{q} \right\rceil$$

Chapter 3

Constructions of Linear Error Correcting Codes

In this chapter we will go through a number of common error correcting codes that are frequently studied in the literature. Many of these codes are used in real world applications and knowing their properties can be important. The constructions can be found in MacWilliams and Sloane (9).

3.1 Hamming Codes

One of simplest yet most elegant codes, the Hamming code, was invented by Richard Hamming in his seminal work on error correcting codes during the 50s (6). While they can only correct a single error, which we will see, they are highly efficient and are used in a variety of applications including computer memory to compensate for hardware failure. They are defined as follows:

Definition 13. *Let $r \geq 2$. An (r, q) Hamming code (denoted $Ham(r, q)$) is a code over $\mathbb{F}_q$ whose parity check matrix has r rows and n columns, where n is the maximum number of possible columns such that any two are linearly independent.*

While this definition does not specify the exact structure of the code, a true fact that will not be proven here is that all Hamming codes with the same parameters are equivalent, up to a permutation of their coordinates.

In $\mathbb{F}_2^n$, two vectors are linearly independent if and only if they are distinct. Thus since the parity check matrix is $(n-k) \times n$, we get that $n = 2^r - 1$, which is the number of nonzero vectors in $\mathbb{F}_2^r$. It follows from the dimension of the

parity check matrix that $k = n - r$.

In $\mathbb{F}_q^n$, each vector has $q - 1$ nonzero multiples, and as such every 1-dimensional subspace of $\mathbb{F}_q^n$ contains $q - 1$ vectors. There are $q^r - 1$ different possible columns of the parity check matrix, and we know that know two of the columns can be in the same 1-dimensional subspace. Thus the maximum number of pairwise independent vectors is

$$n = (q^r - 1)/(q - 1).$$

One of the main properties of Hamming codes is the following:

Theorem 5. *Hamming codes have minimum distance* 3 *and can correct any* 1 *error.*

Proof. Let C be a Hamming code and H its parity check matrix. If we can prove that the minimum distance is 3, it immediately follows that it can correct any 1 error from Theorem 3. Since no two columns of H are linearly independent, there is no vector v of weight 2 or less such that $Hv^T = 0$. Since H has the greatest number of columns such that no two are independent, then there must be a vector u of weight 3 such that $Hu^T = 0$. Thus the minimum distance of C is 3. □

Hamming codes also have another property which makes them highly efficient:

Theorem 6. *Hamming codes are perfect and thus have covering radius* 1.

Proof. To prove this we must show that Hamming codes satisfy the sphere-packing bound. Let's consider the total volume of all spheres of radius 1 centered at codewords:

$$|C| \cdot V_q(1) = q^{n-1} \cdot \sum_{i=0}^{1} \binom{n}{i} (q-1)^i = q^{n-1} \cdot (1 + q^1 - 1) = q^n.$$

Thus Hamming balls of radius 1 are disjoint and cover the entire space, proving the desired result. □

3.2 Reed Muller Codes

Another ubiquitous class of codes are the Reed-Muller codes, which will be discussed at length in future chapters. A Reed-Muller code has two

parameters, r and m, and is denoted $RM(r, m)$, with $0 \leq r \leq m$. Reed-Muller codes can be thought of subspaces of the space of multivariate polynomials. In particular, an $RM(r, m)$ code can be thought of as the subspace of m-variate polynomials with total degree r or less. Because polynomials over finite fields satisfy the relation that $x^q = x$, the linear space of all m-variate polynomials over $\mathbb{F}_q$ is in fact finite.

For simplicity we will only consider the Reed-Muller codes over $\mathbb{F}_2$, although they can easily be extended to any finite field. In the binary case, there are 2^m possible monomials in m variables, and there are $\sum_0^r \binom{m}{r}$ possible monomials of total degree r or less, so an $RM(r, m)$ code is an $[n, k]$ linear code where $n = 2^m$ and $k = \sum_0^r \binom{m}{r}$.

The "message" we are sending in a Reed-Muller code is really the coefficients of a polynomial, and we encode it by evaluating the polynomial at every possible input. As an example, consider the $RM(2, 4)$ code, which is a $[16, 11]$ linear code. If we want to send a message, say, 11010010101, we encode it as a polynomial P_x of total degree 2 or less in four variables, with coefficients coming from the message:

$$P_x[X_1, X_2, X_3, X_4] = 1 + 1 \cdot X_1 + 0 \cdot X_2 + X_3 + 0 \cdot X_4 + 0 \cdot X_1X_2 + ... + 1 \cdot X_3X_4.$$

We then encode P_x by evaluating it at all 2^4 possible input points,

$$P_x(0, 0, 0, 0), P_x(0, 0, 0, 1), ..., P_x(1, 1, 1, 1)$$

which gives us the encoded vector 1111101001010000. The details of decoding the encoded message are unimportant for this thesis, but it can be done very quickly and efficiently which is part of why Reed-Muller codes are so popular.

Reed-Muller codes can be explicitly constructed in a variety of ways, but they have a nice inductive definition which involves the use of the following construction, called the $(u, u + v)$ construction:

Definition 14. *If C_1 and C_2 are codes of the same length n, then there is a code $C_3 = \{(u, u + v) | u \in C_1, v \in C_2\}$.*

Using this, we can define Reed-Muller codes inductively. We first let $RM(0, M)$ be the code containing only the vector of all 0s and the vector of all 1s. This follows because polynomials in 0 variables are constant no matter where they are evaluated. Then

$$RM(r, m) = \{(u, u + v) : u \in RM(r, m - 1), v \in RM(r - 1, m - 1)\}.$$

The $(u, u+v)$ construction also gives us a nice way of inductively finding the generator matrix of arbitrary Reed-Muller codes.

Theorem 7. *If $G_{r,m}$ is the generator matrix of an $RM(r, m)$ code, then*

$$G_{r,m} = \begin{bmatrix} G_{r-1,m-1} & 0 \\ G_{r,m-1} & G_{r,m-1} \end{bmatrix}.$$

Proof. Follows directly from the $(u, u+v)$ construction of Reed-Muller codes. □

3.2.1 The Generalized Hamming Weights of Reed-Muller Codes

In this section we will briefly discuss the Generalized Hamming Weights of Reed-Muller codes, as we will need to compute them in later chapters. While the Hamming Weight hierarchy of q-ary Reed-Muller codes is known, we only need to compute them in the binary case, so only the binary case will be discussed here.

To understand the GHWs, we must first understand what is called the (r, m) decomposition of a nonnegative integer, which is related to its Macaulay representation.

In the following we let $\rho(r, m) = \text{Rank}(RM(r, m)) = \sum_{i=0}^{r} \binom{m}{i}$.

Theorem 8. (11) *Given r, m, and z, $0 \leq t \leq \rho(r, m)$, we can write t as the sum*

$$t = \sum_{i=1}^{k} \rho(r_i, m_i)$$

where the r_i are decreasing, and $m_i - r_i = m - r - i + 1$

As a quick example to illustrate this, we see that the (u, m) canonical representation of 7 is $7 = \rho(4, 1) + \rho(2, 0) + \rho(1, 0)$.

Now that we have established this representation, we can write out the GHW Hierarchy for Reed-Muller Codes:

Theorem 9. (11) $d_t(C) = \sum_{i=1}^{k} 2^{m_i}$, *where* $t = \sum_{i=1}^{k} \rho(r_i, m_i)$.

Chapter 4

The Generalized Covering Radius

4.1 The Generalized Covering Radii of Error Correcting Codes

4.1.1 Definitions

In this section we will state each of the definitions of the t-th generalized covering radius introduced in (3) and prove they are equivalent. We will start with the following definition:

Definition 15. *Let C be an $[n,k]$ code over $\mathbb{F}_q$ with parity check matrix H. Then for every $t \in \mathbb{N}$ we define the t-th generalized covering radius, $R_t(C)$, to be the minimal integer r such that for any set of vectors $S = \{s_1, s_2, ..., s_t | s_i \in \mathbb{F}_q^{n-k}\}$, there exist some r columns of H, indexed by $I \in \binom{[n]}{r}$, such that $S \subseteq \langle H_I \rangle$.*

This brings us to our second definition:

Definition 16. *Let C be an $[n,k]$ code over $\mathbb{F}_q$ with parity check matrix H. Then for every $t \in \mathbb{N}$ we define the t-th generalized covering radius, $R_t(C)$, to be the minimal integer r such that for every $v_1, v_2, .., v_t \in \mathbb{F}_q^n$, there exist codewords $c_1, c_2, ..., c_t$ and $I \in \binom{[n]}{r}$ such that $\text{supp}(v_i - c_i) \subseteq I$ for all $i \in [t]$.*

We now prove these definitions equivalent:

Proof. We begin from Definition 15. Let $v_1, ..., v_t$ be arbitrary column vectors of a matrix $V \in \mathbb{F}_q^{t \times n}$, and let $S = HV^T$. Since H is full rank, S is also arbitrary. Let $r = R(S)$ be the minimal number such that $I \in \binom{[n]}{r}$ is a

minimal set of columns of H such that $S \subseteq \langle H_I \rangle$. Then for each column $s_i \in S$ there is some vector e_i such that $\text{supp}(e_i) \subset I$ and $He_i = s_i$. Then since $He_i = Hv_i$, we can write that $e_i = v_i - c_i$, where c_i is a codeword of the code C. Let $R'(V) = |\bigcup_i \text{supp}(e_i)|$ Notice that $R'(V) = r = R(S) = R(HV)$. Furthermore, from Definition 15, $R_t(C) = \max_S R(S)$. Since the linear map $H : \mathbb{F}_q^{n\times t} \to \mathbb{F}_q^{(n-k)\times t}$ (mapping V to S) is onto, then $\max_V R'(V) = \max_S R(HV) = \max_S R(S) = R_t(C)$, so since $\max_V R'(V)$ is the number described in Definition 16, then this is equivalent to $R_t(C)$. □

We now introduce a more formal version of this definition, for which we first define a new metric on the space of matrices $\mathbb{F}_q^{t\times n}$. The t-weight of a matrix $V \in \mathbb{F}_q^{t\times n}$ is defined to be $wt^{(t)}(V) = |\bigcup_i \text{supp}(v_i)|$. Notice that for $t = 1$ this is just the Hamming weight. Then the t-distance between two matrices V, V' is just $d^{(t)}(V, V') = wt^{(t)}(V - V')$. We can then define a t-ball of radius r centered at a matrix V, $B^{(t)}(V)$, as the set of all matrices V' such that $d^{(t)}(V, V') \leq r$. We can now introduce a new definition:

Definition 17. *Let C be an $[n, k]$ linear code over $\mathbb{F}_q$. Then we define the t-th generalized covering radius of C, $R_t(C)$, to be the minimal integer r such that t-balls of radius r centered at matrices $C^t = [c_1, ..., c_t]^T \in \mathbb{F}_q^{t\times n}$, $c_i \in C$, that is, matrices where all row vectors are codewords in C, covers $\mathbb{F}_q^{t\times n}$. We now prove that this definition is equivalent to the previous one.*

Proof. Let C be a code with t-th generalized covering radius r. Consider an arbitrary matrix $V \in \mathbb{F}_q^{t\times n}$. Then we know from Definition 15 that we can find a matrix with codewords as rows C^t such that for each row $e_i, 0 \leq i < t$ in $V - C^t$, for some $I = \binom{[n]}{r}$, $\text{supp}(e_i) \subseteq I$. Then it follows that $|\bigcup_i \text{supp}(e_i)| \leq r$, and by the minimality of r there is some matrix V such that $|\bigcup_i \text{supp}(e_i)| = r$, and there is no matrix C^t that makes $|\bigcup_i \text{supp}(e_i)|$ smaller. By the definition of our t-th distance metric, for all V, $d(V, C^t) \leq r$ for some C^t, and for some V, $d(V, C^t) \geq r$ for all C^t. Thus, r is the minimal integer such that radius r t-balls centered at the C^t cover $\mathbb{F}_q^{n\times t}$. □

We now move to our final definition:

Definition 18. *Let C be an $[n, k]$ linear code over $\mathbb{F}_q$ with generator matrix G. Then we define the t-th generalized covering radius of C, $R_t(C)$, as simply the standard covering radius of C', where C' is the code with generator matrix G over $\mathbb{F}_{q^t}$*

Proof. We know there is an isomorphism Φ from $\mathbb{F}_{q^t} \rightarrow \mathbb{F}_q^t$, and so there is also an isomorphism $\Phi : \mathbb{F}_q^{t\times n} \rightarrow \mathbb{F}_{q^t}^n$, defined by appling ϕ to each column of V element-wise. So we can think of $V \in \mathbb{F}_q^{n\times t}$ as a vector $\Phi(V) = (v_1, v_2, ...v_n) \in F_{q^t}^n$. Then we see that the t-distance between two matrices $d^{(t)}(V, V')$ is equivalent to the Hamming distance (or 1-distance) between their images in $\mathbb{F}_{q^t}^n$. That is, $d^{(t)}(V, V') = d^{(1)}(\Phi(V), \Phi(V'))$. Furthermore, we know that matrices with codewords from the code C are just codewords in C', so it follows directly from the previous definition that the covering radiius of C', $R_1(C')$, is equivalent to $R_t(C)$. □

We now briefly turn our attention to the effect of basic code operations on the t-th generalized covering radius of the code. It is easy to show that for the operations of code puncturing, code extension, $(u, u + v)$ construction, and direct sum, the t-th generalized covering radius has the same properties as the generalized covering radius. It is known that if C is a code, with C it's code puncturing and $\overline{C}$ it's code extension, $R_1(C*) = R_1(C)$ or $R_1(C) - 1$, and $R_1(\overline{C}) = R_1(C)$ or $R_1(C) + 1$. The same is true of the t-th generalized covering radius, that is, $R_t(C*) = R_t(C)$ or $R_t(C) - 1$, and $R_1(\overline{C}) = R_t(C)$ or $R_t(C) + 1$. Furthermore, if C_1, C_2 are codes and $C = \{(u, u + v), u \in C_1, v \in C_2\}$, then $R_t(C) \leq R_t(C_1) + R_t(C_2)$. Lastly, if $C = C_1 \bigoplus C_2$, the direct sum of C_1 and C_2, then $R_t(C) = R_t(C_1) + R_t(C_2)$ is The proof is nearly identical for all theorems, and will be presented only for the code puncturing:

Proof. Let C be an $[n, k]$ linear code with generator matrix G over $\mathbb{F}_q$ and C_t be the code with the same generator matrix but over $\mathbb{F}_{q^t}$. Then the generator matrices for tbe code puncturings of $C*$ and C_t* are identical. It follows that $R_t(C*) = R_1(C_t*) = R_1(C_t)$ or $R_1(C_t) - 1$. □

The theorems for the other three operations can be proven with a near-identical proof.

We will now prove that the t-th generalized covering radius of any Hamming code is t.

Definition 19. *A Hamming code is a* $[2^r - 1, 2^r - r - 1]$ *linear code. It is a perfect code with covering radius* 1.

Theorem 10. (3) *Let H be an $[n, k]$ Hamming code. Then $R_t(C) = \min(t, n - k)$.*

Proof. Let C be an $[n, k]$ Hamming code and H its parity check matrix. Recall that the parity check matrix H, which is of size $(n - k) \times 2^{(n-k)} - 1$ contains

every possible nonzero column vector. For any $t \leq (n-k)$, we can then choose t linearly independent columns of H as our $S = s_1, s_2, ...s_t$ as defined in 15. Since $dim(S) = t$, then clearly for any H_I, $I \in \binom{[n]}{r}$, where $S\langle H_I \rangle$, $r \geq t$, meaning t is a lower bound on $R_t(C)$. Since t is an upper bound on the generalized covering radius (due to subadditivity and the fact that $R_1(C) = 1$, it follows that $R_t(C) = t$. □

4.2 A new bound on the generalized covering radius

One of the primary findings of this thesis is that Theorem 4 can be extended to a new bound for the generalized covering radius, which we denote $U_t(C)$:

Theorem 11. *Let C be a code satisfying the chain condition with t-th generalized covering radius $R_t(C)$ and generalized Hamming weights $d_1, d_2, ..., d_k$. Then*

$$R_t(C) \leq n - \sum_{r=1}^{k} \left\lceil \frac{d_r - d_{r-1}}{q^t} \right\rceil = U_t(C).$$

Proof. To prove this theorem, we will use the fact that $R_t(C)$ is the covering radius of code generated by the generator matrix of C over $\mathbb{F}_q^t$, which we denote C^t. Therefore, Theorem 4 will extend to $R_t(C)$ if C^t both satisfies the chain condition and has the same generalized Hamming weights as C.
We first prove that C_t has the same generalized Hamming weights as C. To prove this, let d_r be the r-th generalized Hamming weight of C and d_r^t be the r-th generalized Hamming weight of C^t. We first note that because $\mathbb{F}_q$ is a subfield of $\mathbb{F}_q^t$, every codeword in C naturally embeds into a codeword of C^t. Remember that $d_r(C) = \{\min |\text{supp}(D)| : D \textit{ is an } r\textit{-dimensional subcode of } C$. The basis for this r-dimensional subcode therefore also exists in C^t and has the same support, and so it follows that $d_r^t \leq d_r$.
Now to show that $d_r = d_r^t$, we must just show that $d_r^t \geq d_r$. To see this, consider that $\mathbb{F}_{q^t}$ is isomorphic to $\mathbb{F}_q^t$ since both are isomorphic to t-dimensional vector spaces over $\mathbb{F}_q$. Therefore, we can view codewords in C^t, which are vectors in $\mathbb{F}_{q^t}^n$ as elements of $\mathbb{F}_q^{t \times n}$, that is, $t \times n$ matrices over $\mathbb{F}_q$. If $c \in C^t$, then

$$\text{supp}(c) = \bigcup_{i=1}^{t} \text{supp}(c^i)$$

where the c^i are the rows of c when c is viewed as a matrix. Furthermore, the rows of c are codewords of C. It follows that $|\text{supp}(c)| \geq |\text{supp}(c^i)|$.

It follows immediately that for any basis of an r-dimensional subcode of C^t, $\{u_1, ..., u_r\}$, we can always choose rows of the u_i which are linearly independent and nonzero. This is because the generalized Hamming weight hierarchy is strictly increasing as we see in Theorem 1 of (11), so every u_i must have a row with a new nonzero coordinate. Our chosen rows form a subcode of C with support of magnitude less than or equal to the original subcode. Since d_r and d_r^t are the magnitudes of the supports of such subcodes, it follows that $d_r \leq d_r^t$. Therefore $d_r = d_r^t$.

It immediately follows that since $d_r = d_r^t$, C^t also satisfies the chain condition. Why? Because we can use the exact same codewords in C^t that were used to satisfy the chain condition in C. We can do this because the embedding of a vector in F_q^n in $F_{q^t}^n$ has the same support as the original vector, and since the emebeddings of the codewords of C are codewords in C^t, the result follows. □

Upon some exploration, this bound has proven to be better, both asymptotically and in many specific cases to the bounds derived in (4). Since this bound is highly general and can be applied to any code that satisfies the chain condition, there is a huge amount of further analysis that can be done in this regard. Furthermore, there are potentially more results in (7) that can be applied to improve the above bound, such as for codes which partially satisfy the chain condition.

4.3 A comparison to other known bounds

It is difficult to compare this bound to other bounds on the generalized covering radius for a few reasons. One is that there aren't that many bounds out there, and the ones that exist all have restrictions on either t, q, or the type of code such as the bounds in (3), (4). In contrast, the only restriction on U_t is that the code needs to satisfy the chain condition, which is satisfied by almost all commonly discussed codes including Reed-Muller, MDS, BCH (partially), Dual Hamming, and extended Golay codes as discussed in (12), (7). Specifically in this section we will use computational tools to compare the bounds derived in (4) with our new bound. For ease of reference, table 4.1 summarizes the results from (3).

One challenge with comparing $U_t(C)$ with these bounds is that most of them are asymptotic approximations rather than exact bounds. However, we can still compare the bounds' performance asymptotically. In the following

$R_t(r,m)$	$\leq (1-\frac{1}{2^t})2^m - \frac{\sqrt{2^t-1}}{2^t}(1+\sqrt{(2)})^{r-1}\cdot 2^{m/2} + O(m^{r-2})$
$R_t(m-s,m)$	$\leq \frac{t}{(s-2)!}\cdot m^{s-2} + O(m^{s-3})$
$R_t(\alpha m,m)$, for $0<\alpha<1-\frac{1}{\sqrt{(2)}}$	$\leq (1-\frac{1}{2^t})2^m - \frac{\sqrt{2^t-1}}{2^t(2+\sqrt{(2)})}\cdot 2^{m\cdot(1/2+\alpha\cdot\log_2(1+\sqrt{(2)}))}\cdot(1+o(1))$
$R_t(\alpha m,m)$, for $1-\frac{1}{\sqrt{(2)}}\leq\alpha\leq 1/2$	$\leq (1-\frac{1}{2^t})2^m - \frac{\sqrt{2^t-1}}{2^t}\cdot\frac{1}{\sqrt{(8m\alpha(1-\alpha))}}\cdot 2^{mH_2(\alpha)}$
$R_t(\alpha m,m)$, for $1/2\leq\alpha\leq 1$	$\leq t\cdot 4^{H_2(\alpha)}\cdot 2^{mH_2\alpha}\cdot(1+o(1))$

Table 4.1 Table of upper bounds for Reed-Muller Codes

comparisons, the O terms are ignored, leading to underestimates of the first two bounds and overestimates of the third bound.

4.3.1 Comparison with $R_t(r,m)$

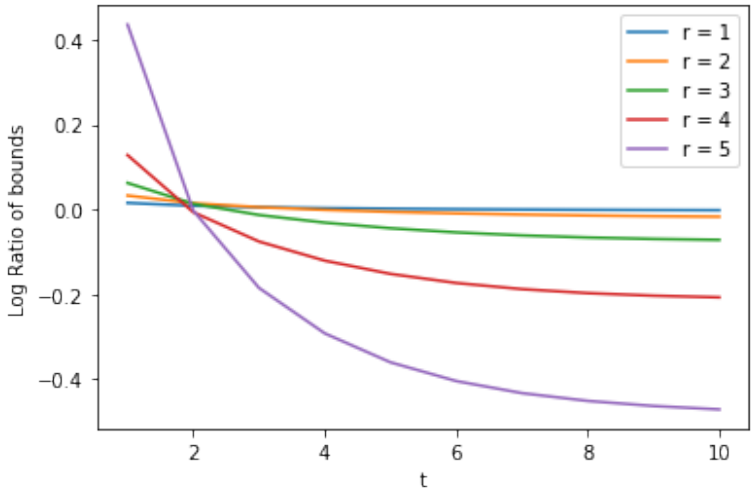

Figure 4.1 A plot of $\log(U_t(RM(r,12))/R_t(r,12))$ for various values of r and t.

We first compare $U(C)$ with $R_t(r,m)$. In all following plots, a negative value means that $U_t(C)$ takes on a lower value than the bound it is being compared to. For this chart it is especially true because we cannot account for the $O(m^{r-2})$ term which occurs in the bound $R_t(r,m)$. We see that in Figure 4.1 our bound appears to outperform $R_t(r,m)$ asymptotically as t

gets larger and also as r gets larger.

However, we can still determine asymptotic performance in m by fixing $r = 5$ and comparing the bounds for various values of m and t. We see that for fixed r and t, the bounds appear to approach each other as $m \to \infty$. Because this bound is exponential in m, the $O(m^r)$ term does not matter here. However, our new bound is still an improvement on the old one for a variety of values.

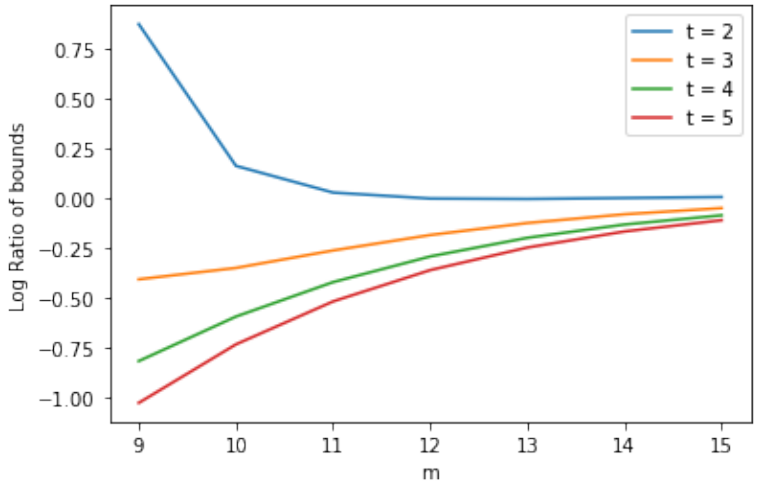

Figure 4.2 A plot of $\log(U_t(RM(5,m))/R_t(5,m))$ for various values of m and t.

Furthermore, in the next chapter we will discuss bounding the generalized covering radius of $RM(1,m)$ and finding vectors within the bound, so this comparison is highly relevant.

4.3.2 Comparison with $R_t(m-s,m)$

We now move on to comparing our new bound with the second bound of Table 4.1.

Interestingly, for this bound we find that while U_t is an improvement for low values of m (so when s is large relative to m, for fixed s $R_t(m-s,m)$ is clearly better asymptotically as $m \to \infty$. However, we also see that for fixed s and m, U_t outperforms as t gets larger.

Perhaps unsurprisingly, we see that when m is fixed and s is increased, U_t performs better than $R_t(m-s,s)$.

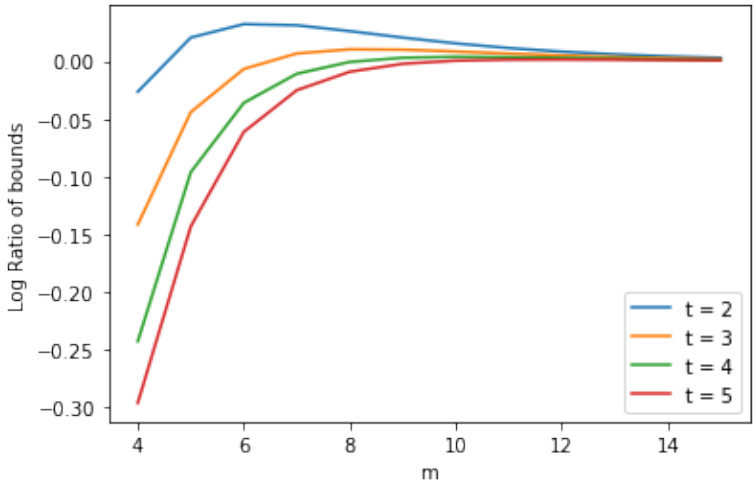

Figure 4.3 A plot of $\log(U_t(RM(1,m))/R_t(1,m))$ for various values of m and t.

4.3.3 Comparison with $R_t(\alpha m, m)$

In this section we will compare U_t with the last 3 upper bounds of table 4.1 for $R_t(\alpha m, m)$ that are given for various values of α. We will consider each as α changes and as m changes.

Interestingly, when s is small and fixed, R_t performs better significantly than U_t asymptotically. Furthermore, we see that when $\alpha \geq 1 - 1/\sqrt{(2)}$, U_t appears to be asymptotically better in α, and of course in all cases U_t appears to be asymptotically better in t. Interestingly, U_t appears to perform significantly better than R_t when $\alpha \geq .5$, taking on values 3-5 orders of magnitude smaller than R_t.

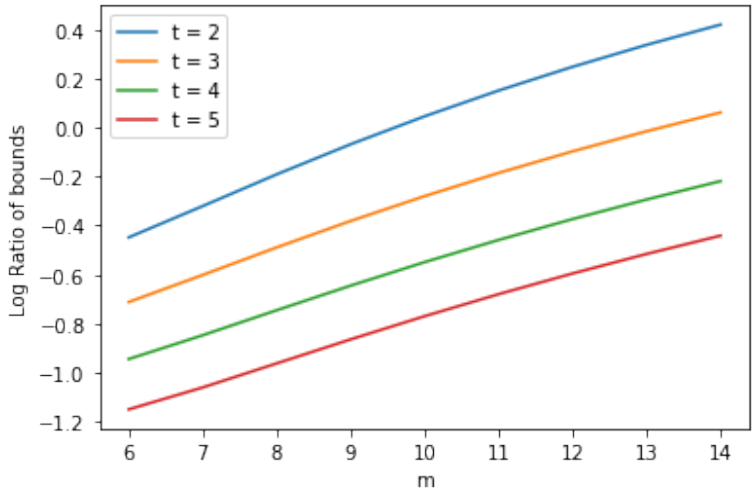

Figure 4.4 A plot of $\log(U_t(RM(m-5,m))/R_t(m-5,m))$ for various values of m and t.

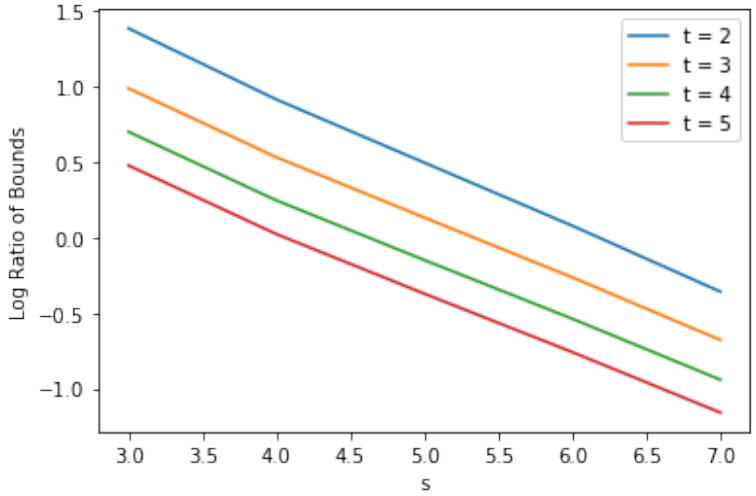

Figure 4.5 A plot of $\log(U_t(RM(15-s,15))/R_t(15-s,15))$ for various values of s and t.

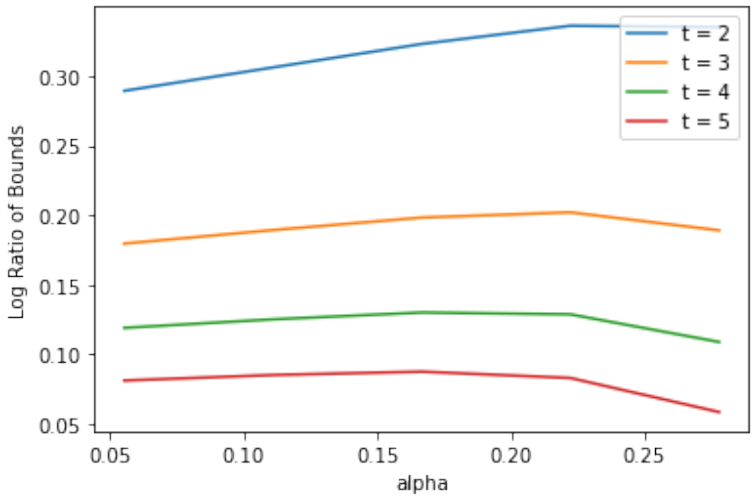

Figure 4.6 A plot of $\log(U_t(RM(\alpha \cdot 18, 18))/R_t(\alpha \cdot 18, 18))$ for various values of α and t.

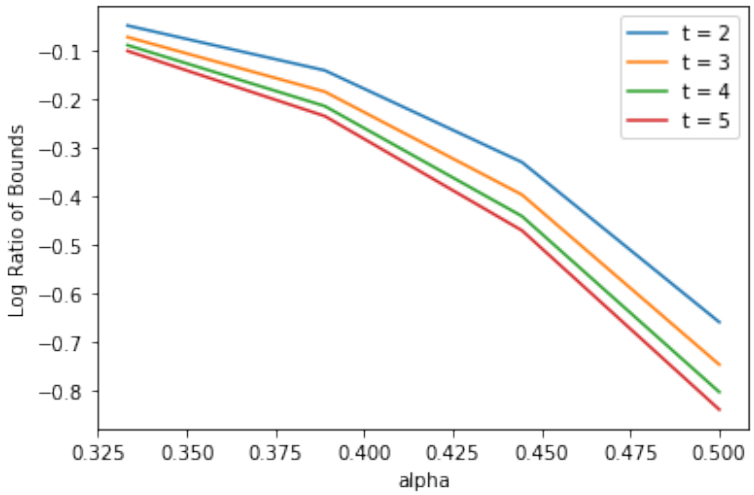

Figure 4.7 A plot of $\log(U_t(RM(\alpha \cdot 18, 18))/R_t(\alpha \cdot 18, 18))$ for various values of α and t.

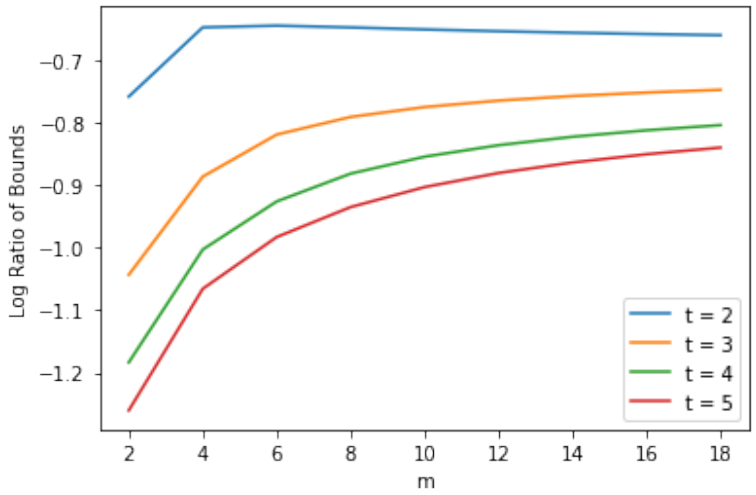

Figure 4.8 A plot of $\log(U_t(RM(.5m, m))/R_t(.5m, m))$ for various values of m and t.

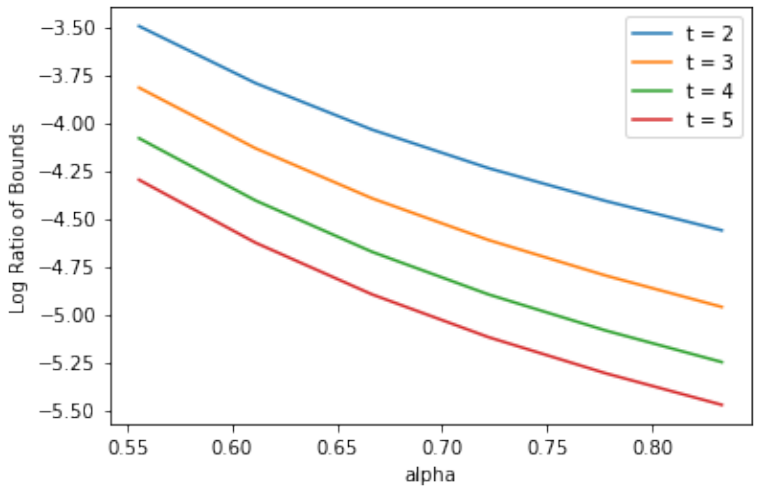

Figure 4.9 A plot of $\log(U_t(RM(\alpha \cdot 18, 18))/R_t(\alpha \cdot 18, 18))$ for various values of α and t.

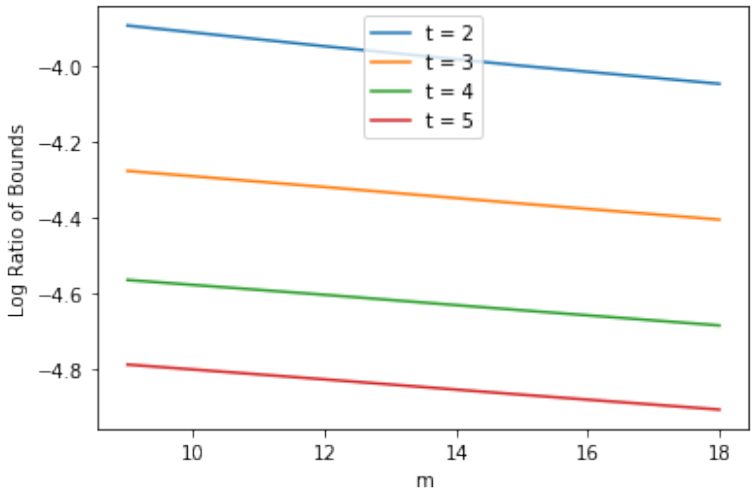

Figure 4.10 A plot of $\log(U_t(RM(\frac{2}{3}m, m))/R_t(\frac{2}{3}m, m))$ for various values of m and t.

Chapter 5

A General Covering Algorithm

In Chapter 4 we established a new bound on the generalized covering radius for all codes which satisfy the chain condition. The paper (4) establishes a bound on the generalized covering radius of Reed-Muller codes, and goes on to define an algorithm which could theoretically be used to find codewords of an RM code which can cover any given vector v. It is a natural question to ask whether the same can be done for U_t. The answer is yes, and in this section we present an algorithm which, given that the code satisfies the chain condition and the generator matrix is in a specific form, can cover any arbitrary input vector in $O(n \cdot k \cdot \log(q)^2)$ time. Furthermore, a major shortcoming of the covering algorithm presented in (4) is that it required many brute force searches of the codewords of $RM(1, m)$ to cover vectors as a basic function of the covering algorithm. However, our new algorithm can be easily applied to cover vectors in $RM(1, m)$ in linear time. This leads to a significant asymptotic speed up of the old covering algorithm. However, the effect on performance is unknown and can be easily researched in the future.

5.1 Derivation

The derivation of this algorithm comes from the complete proof of Theorem 4 discussed in Section 2.4. Thus, we will dig into this proof, starting from the basics.

For the rest of this chapter, we will let C be an $[n, k]$ code and J be a subset of the coordinates of C. Furthermore, we consider a generator matrix of C of the form

$$\begin{matrix} & J \\ \begin{bmatrix} \mathrm{g(C_0)} & 0 \\ A & \mathrm{g(C_J)} \end{bmatrix} \end{matrix}$$

where $g(C_J)$ is the generator matrix of C_J, the projection of C onto the coordinates J, and $g(C_0)$ is the generator matrix of C_0, the subcode of C which is 0 on J (but does not contain the coordinates in J).

Theorem 12. (10) $R(C) \leq R(C_J) + R(C_0)$.

Proof. Let v be an arbitrary vector of length n. We see that v can be written as (v_0, v_J), where v_J is a length $|J|$ vector in the coordinates J, and v_0 is a length $n - |J|$ vector in the remaining coordinates. Then there is some codeword of C which is of the form (a, c_J) where $c_J \in C_J$, such that $d(v_J, c_J) \leq R(C_j)$. Furthermore, there is a codeword of the form $(c_0, 0)$, where $c_0 \in C_0$, such that $d(v_0 + a, c_0) \leq R(C_0)$. Therefore v is at most distance $R(C_J) + R(C_0)$ from $(a, c_J) + (c_0, 0)$. □

We also introduce a new definition:

Definition 20. *Let C be a code which satisfies the chain condition and has generalized Hamming weights $d_1, d_2, ..., d_k$. Then there exist codewords of C, $c_1, ..., c_k$ such that $d_i = |\bigcup_{j=1}^{i} \mathrm{supp}(c_j)|$ and each c_i is of the form $(e_i, 0)$, where $|e_i| = d_i$. We define the generator matrix*

$$\Gamma(C) := \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{bmatrix}$$

From this definition we can see that $\Gamma(C)$ has the form

$$\begin{matrix} & d_r \\ \begin{bmatrix} \mathrm{g(C_r)} & 0 \\ A & \mathrm{g(C_0)} \end{bmatrix} \end{matrix}$$

where C_r is the $[d_r, r]$ code generated by the nonzero coordinates of $c_1, ..., c_r$. It has this form for all r, $1 \leq r \leq k$ (although in the $r = k$ case it is understood that the entire matrix is just $g(C_r)$).

Theorem 13. (7) *Let C be an $[n, k]$ code that satisfies the chain condition with GHWs $d_1, d_2, ..., d_k$. Then the covering radius of C, $R(C)$ satisfies the following bound*:

$$R(C) \leq n - \sum_{r=1}^{k} \left\lceil \frac{d_r - d_{r-1}}{q} \right\rceil$$

Proof. Let C be an $[n, k]$ code which satisfies the chain condition and has GHWs $d_1, d_2, ..., d_k$. Consider the matrix $\Gamma(C)$ which is of the form described in definition 20. We will prove this theorem by induction on r, or the number of rows of $g(C_r)$. In our base case $r = 1$, and $G(C_1) = e_1$. We note that e_1 has weight d_1 and is thus a codeword of minimum distance of C. Since e_1 is 0 nowhere, by the pidgeonhole principle any vector v must share at least $\lceil d_1/q \rceil$ equivalent coordinates with e_1 or a scalar multiple of e_1—in other words $d(v, e_1) \leq d_1 - \lceil d_1/q \rceil$, so $R(C_1) \leq d_1 - \lceil d_1/q \rceil$.

In the inductive step, we assume that $R(C_r) \leq d_r - \sum_{i=1}^{r} \lceil (d_i - d_{i-1})/q \rceil$. Next, consider that by construction

$$g(C_{r+1}) = \begin{array}{c} \begin{matrix} d_r & d_{r+1} - d_r \end{matrix} \\ \begin{bmatrix} \mathrm{g(C}_r\mathrm{)} & \mathrm{O} \\ \multicolumn{2}{c}{\mathrm{e}_{r+1}} \end{bmatrix} \end{array}$$

Since the last $d_{r+1} - d_r$ elements of e_{r+1} must be nonzero (because $g(C_{r+1})$ cannot have columns of all zeros), we can apply Theorem 9 to this to see that $R(C_{r+1}) \leq R(C_r) + d_{r+1} - d_r - \lceil (d_{r+1} - d_r)/q \rceil$. Using our induction hypothesis, this implies that $R(C_{r+1} \leq d_r - \sum_{i=1}^{r} \lceil (d_i - d_{i-1})/q \rceil + d_{r+1} - d_r - \lceil (d_{r+1} - d_r)/q \rceil = d_{r+1} - \sum_{i=1}^{r+1} \lceil (d_i - d_{i-1})/q \rceil$. Thus, our induction is complete. Simply substituting k into the proven formula, and observing that $C_k = C$ and $d_k = n$ yields the desired result.

□

Hopefully from this proof the covering algorithm becomes clear. If we have the matrix $\Gamma(C)$, then starting with the very last row (which is e_k), we find the closest multiple of the rightmost $d_k - d_{k-1}$ coordinates of e_k to those same coordinates in the vector we wish to cover with a codeword. We then add the remaining coordinates of e_k to our vector and continue the process with e_{k-1}, iterating until we arrive at e_1. We are effectively reversing the induction used to prove the bound, covering our input vector in parts as we do. The actual algorithm is presented below:

Algorithm 1 A t-covering algorithm for any code which satisfies the chain condition

Input: $\Gamma(C), v, (d_1, ..., d_k)$ ▷ To find a t-covering, v must be a vector over $\mathbb{F}_q^t$
Output A vector $c \in \mathbb{F}_q^t$ such that $d(c, v) \leq U_t(C)$
$V \leftarrow v$
$K \leftarrow k$
while $K \geq 1$ **do**
  $c_K \leftarrow \Gamma(C)^K$ ▷ Get Kth row of $\Gamma(C)$
  $a \leftarrow$ Min_dist_coeff(V, c_K, $(d_{K-1}, d_K]$)
  $V \leftarrow V - a \cdot c_K$
  $K \leftarrow K - 1$
end while
Return $v - V$

5.2 Asymptotic Analysis

Now we turn to analyzing the running time of this algorithm:

Theorem 14. *The algorithm T-COVER*$(\Gamma(C), v, (d_1, ..., d_k))$ *runs in* $O((k + \log(q)^2)(n))$ *time.*

Proof. The function "Mindist" in this algorithm simply computes $V^i(c_k^i)^{-1}$, $\forall i \in (d_{K-1}, d_K])$, and then finds the most frequently occurring value out of these values. Computing $(c_k^i)^{-1}$ can be done using the extended Euclidean algorithm which runs in $O(\log(q)^2)$ time. Thus cover has complexity $O(\log(q)^2 \cdot (d_K - d_{K-1}))$. However, when we consider the total amount of calls of Mindist, we see that it is computing in total n inverses, and so takes in total $O(\log(q)^2 \cdot n)$ time.

Furthermore, computing $V - a \cdot c_K$ takes $O(nt)$ time and this computation is done k times, taking $O(n \cdot t \cdot k)$ time in total.

Thus, in total the algorithm runs in $O((kt + \log(q)^2)(n))$ time.

□

5.2.1 A note on usefulness

At first glance this algorithm seems quite attractive. It's running time is linear or sublinear in all code parameters, and as we've seen in previous sections the bound can be quite good compared to other bounds. Unfortunately, for almost all codes C, computing $\Gamma(C)$ is an open problem with unknown

complexity. One method to find this generator matrix would be to iteratively compute codewords of minimum distance. Furthermore, computing the codeword of minimum distance is believed to be NP-Hard in most cases, although efficient algorithms do exist for cyclic codes (1). Fortunately, once $\Gamma(C)$ is computed for a code, it never needs to be computed again and no additional computation is needed to run the $t-$Covering algorithm. It is unclear how useful this algorithm is in practice, although it is useful when we have an "infinite" stream of vectors to cover such as in the example presented in (3), and the amount of preprocessing that needs to be done is irrelevant.

5.2.2 Example

Let's quickly run through an example of the algorithm at work to make things clear. Consider a code C , which satisfies the chain condition, and for which

$$\Gamma(C) = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

We can easily read off from the rows of $\Gamma(C)$ that it has a GHW hierarchy of $(4, 6, 7, 8)$. We know that C satisfies the chain condition because this is in fact the generator matrix of an $RM(1, m)$ code for which this is proven (in the next section!). Now suppose we want to use the algorithm to find the closest codeword to $v = (1, 0, 0, 1, 1, 1, 0, 1)$. Our first pass through of the algorithm simply looks at the last coordinate place and sees that they are matching. So we compute $v' = v - \Gamma(C)^4 = (0, 1, 1, 0, 0, 0, 1, 0)$. Next, we look at the 7th coordinate place and see that they are also now matching in both vectors, so we compute $v'' = v' - \Gamma(C)^3 = (1, 1, 0, 0, 1, 0, 0, 0)$. We repeat the same procedure but in the 5th and 6th coordinate places. When we do this we see that $1 \cdot (1, 1)$ and $0 \cdot (1, 1)$ are equidistant from $v''[5 : 6] = (0, 1)$, so it doesn't matter what vector we add, but each one will give us different results. If we choose to compute $v''' = 1 \cdot \Gamma(C)^2 + v'' = (0, 0, 0, 0, 0, 1, 0, 0)$ and we are done! We see that $v - v''' = (1, 0, 0, 1, 1, 0, 0, 1)$ is a codeword close enough to v to be within our bound. However, if we compute $v''' = 0 \cdot \Gamma(C)^2 + v''$, then we go on to our last vector and find again that $0 \cdot \Gamma(C)^1$ and $1 \cdot \Gamma(C)^1$ are equidistant to v''', and in fact it doesn't matter which one we chose. Choosing

the coefficient 1, we get that $v'''' = (0, 0, 1, 1, 1, 0, 0, 0)$, so the codeword we find in this instance is $v - v'''' = (1, 0, 1, 0, 0, 1, 0, 1)$, which is distance 3 away from v.

5.2.3 $\Gamma(RM(1, m))$

In order to apply our covering algorithm to any code C we need to know $\Gamma(C)$. Thus, in order to apply our covering algorithm to $RM(1, m)$, we need to know $\Gamma(RM(1, m))$. Recall that an $RM(1, m)$ code is the subspace of all m-variate polynomials of total degree 1 or less in the space of m-variate polynomials over $\mathbb{F}_2$. Furthermore, recall that the encoding of one of these polynomials is simply their evaluation at every point in $\mathbb{F}_2^m$.

To find the matrix $\Gamma(RM(1, m))$, we need to find a list of polynomials which are representative of the GHW hierarchy.

Theorem 15. (11) *Let C be an $RM(1, m)$ code. Then $d_r(C) = 2^{m-1} + 2^{m-2} + ... + 2^{m-r}$ for $1 \leq r \leq m$, and $d_{m+1}(C) = 2^m$.*

Now that we know this hierarchy, if we can find a series of polynomials whose encodings' total support is equal to these GHWs, then we also have $\Gamma(C)$. Fortunately, finding and proving a series of polynomials is quite easy.

Theorem 16. *Let C be an $RM(1, m)$ code in variables $x_1, x_2, ..., x_m$, and let $H(P(\overrightarrow{x}))$ be a linear transformation from $\mathbb{F}_2^{m+1} \to \mathbb{F}_2^{2^m}$ which encodes degree 1 polynomials as codewords in C. Let $P_k(\overrightarrow{x}) = \sum_{i=1}^{k} x_i$ and $P_0(\overrightarrow{x}) = 1$.*

$$\textit{Then } \Gamma(C) = \begin{bmatrix} H(P_m) \\ H(P_{m-1}) \\ \vdots \\ H(P_1) \\ H(P_0) \end{bmatrix}$$

Proof. To prove this we will prove that $|\bigcup_{i=0}^{r-1} \text{supp}(H(P_{m-i}))| = d_r(C)$ for all $1 \leq r \leq m$, and we will do so by inducting on r.

Our base case is when $r = 0$. Consider that in the evaluation of the polynomial $P_m(x) = x_1 + ... + x_m$, we are really evaluating the parity of the weight of the binary vector $(x_1, ..., x_m)$, and the support set of $H(P_m)$ has size equal to the number of binary vectors of length m with odd weight. We can prove that this quantity is 2^{m-1} using another inductive argument on m.

In our base case is when $m = 1$. The two vectors are 1 and 0, so $2^{m-1} = 1$ have odd weight.

Next, assume this is true for some $m \in \mathbf{N}$. Clearly the vectors of odd weight in $\mathbb{F}_2^{m+1}$ are all of the vectors of even weight in $\mathbb{F}_2^m$ concatenated with a 1, as well as all of the vectors of odd weight in $\mathbb{F}_2^m$ concatenated with a 0. The total number of these vectors is $2^{m-1} + (2^m - 2^{m-1}) = 2^m$.

Therefore, $|\text{supp}(H(P_m))| = 2^{m-1} = d_1(C)$.

Now, assume our inductive hypothesis is true for some $r \leq m$. Clearly, the coordinates which are 0 on all $P_m, P_{m-1}, ..., P_{m-r+1}$ are 0 only when $x_m = x_{m-1} = ... = x_{m-r+1} = 0$. (Consider that if $P_k(x) = 0$ and $P_{k-1}(x) = 0$, then $P_k(x) + P_{k-1}(x) = x_k = 0$). Therefore, the remaining 0 coordinates are precisely the zeros of P_{m-r+1} considering only the first $m - r + 1$ variables (the rest of the variables must be 0). By our previous inductive argument, half these zeros are when $x_{m-r-1} = 1$ and $P_{m-r} = 1$. Therefore half of them will be 1 on P_{m-r}. Thus, since the number of zeros of P_{m-r+1} when the last $r - 1$ coordinates are fixed to 0 is 2^{m-r}, the total support will increase by $2^{m-r}/2 = 2^{m-r-1}$. Thus, $|\bigcup i = 0^{(r+1)-1}\text{supp}(H(P_{m-i}))| = d_r(C) + 2^{m-r-1} = d_{r+1}(C)$ by Theorem 15. □

5.2.4 $\Gamma(RM(r, m))$

As we now know the form of $\Gamma(RM(1, m))$, it is nice to note that we may be able use this to compute $\Gamma(RM(r, m))$ for arbitrary Reed-Muller codes. We see that the generator matrix for $RM(r, m)$ as given in theorem 14 follows the form of $\Gamma(C)$. This leads to the following conjecture:

Conjecture 17. *Let C be an $RM(r, m)$ code with generator matrix*

$$G_{r,m} = \begin{bmatrix} G_{r-1,m-1} & 0 \\ G_{r,m-1} & G_{r,m-1} \end{bmatrix}.$$

Then

$$\Gamma(C) = \begin{bmatrix} \Gamma(G_{r-1,m-1}) & 0 \\ G_{r,m-1} & \Gamma(G_{r,m-1}) \end{bmatrix}.$$

and the following theorem:

Theorem 18. *Conjecture* 17 *is true for all $m \leq 20$*

Proof. Done by checking all cases by computer. □

The proof for this theorem comes from verifying all possible cases up to $m = 20$. It is only a conjecture that this theorem holds for all r, m, and the proof likely lies in the form of the GHW hierarchy for Reed-Muller codes.

We also note that since $\Gamma(RM(m,m))$ is the identity matrix, this allows us to recursively compute $\Gamma(RM(r,m))$ in general, since we know the exact form of $\Gamma(RM(1,m))$ and $\Gamma(RM(m,m))$.

5.3 An improvement to Elimelech's covering algorithm

In (4) a t-covering algorithm is presented to find t-coverings of vectors for Reed-Muller codes within a bound that they specify within the paper.

A major shortcoming of the algorithm is that the base case requires a brute force computation of the minimum distance between all codewords of $RM(1,m)$ and a fixed vector v. This leads the algorithm to be at minimum quadratic in the length of the code, and exponential in t. Fortunately, our algorithm can be used to avoid this brute-force method and significantly improve the run-time, while having a negligable impact on performance.

The algorithm, taken from (4) is as follows:

Algorithm 1: A t-covering algorithm for $\mathrm{RM}(r,m)$ with radius $U_t(r,m)$

Function `recursive` $(\mathbf{v}, r)$
- **Input** : $\mathbf{v} \in \mathbb{F}_2^{t\times 2^m}$, $r \in \mathbb{N}$, $1 \leqslant r \leqslant m$
- `// Check edge cases`
- **if** $r = m$ **then return** $\mathbf{v}$
- **if** $r = 1$ **then return** $\operatorname{argmin}_{\mathbf{c}\in \mathrm{RM}(1,m)^t} d^{(t)}(\mathbf{v},\mathbf{c})$
- `// Use the` $(u, u+v)$ `recursion`
- Let $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{F}_2^{t\times 2^{m-1}}$ s.t. $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2)$
- $\mathbf{c}_1 \leftarrow$ `recursive`$(\mathbf{v}_1, r)$
- $\mathbf{c}_2 \leftarrow$ `recursive`$(\mathbf{v}_2 - \mathbf{c}_1, r-1)$
- **return** $(\mathbf{c}_1, \mathbf{c}_1 + \mathbf{c}_2)$

Function `subadditive` $(\mathbf{v}, r)$
- **Input** : $\mathbf{v} \in \mathbb{F}_2^{t\times 2^m}$, $r \in \mathbb{N}$, $1 \leqslant r \leqslant m$
- `// Use subadditivity`
- Let $\overline{v}_i$ be the i-th row of $\mathbf{v}$
- **forall** $i \in [t]$ **do**
  - $\overline{c}_i \leftarrow$ `recursive`$(\overline{v}_i, r)$
- **return** $(\overline{c}_1^\intercal, \ldots, \overline{c}_t^\intercal)^\intercal$

Function `cover` $(\mathbf{v}, r)$
- **Input** : $\mathbf{v} \in \mathbb{F}_2^{t\times 2^m}$, $r \in \mathbb{N}$, $1 \leqslant r \leqslant m$
- $\mathbf{c}_{\min} \leftarrow$ `recursive`$(\mathbf{v}, r)$
- $\mathbf{c}'_{\min} \leftarrow$ `subadditive`$(\mathbf{v}, r)$
- **return** $\operatorname{argmin}_{\mathbf{c}\in\{\mathbf{c}_{\min}, \mathbf{c}'_{\min}\}} d^{(t)}(\mathbf{v},\mathbf{c})$

We see that because we know $\Gamma(RM(1,m)$, we can use our algorithm on

the $r = 1$ base case, which can greatly increase the running time, but has a negative impact on performance due to the fact that we are no longer finding the closest vector to v in the base case, but just a vector within our bound.

5.3.1 Runtime analysis

We will first note, without proof, the following:

Theorem 19. (4) *For any $t, r, m \in \mathbf{N}$, $COVER(v, r)$ has complexity*

$$O(t2^t 2^{(t+1)(m+1)}(2^{t+1} - 1)^{-r} + tm2^m).$$

Furthermore, let COVER' be the version of the above algorithm with the modified base case which runs in $O(n \cdot k \cdot t)$ time. Then we have the following:

Theorem 20. *For any $t, r, m \in \mathbf{N}$, $COVER'(v, r)$ has complexity $O(n \cdot \log(n) \cdot t$, where $n = 2^m$.*

Proof. We will prove that this is the complexity of RECURSIVE(v, r) , and hence the complexity of the entire algorithm. We denote the running time of RECURSIVE(v, r) as $T(t, r, m)$. We have two base cases:

We see that when $r = m$, $T(t, m, m) = c' \leq c \cdot n \log(n) \cdot t$ for some constant c. Furthermore, when $r = 1$, we see that $k = \log(2^m) + 1 = \log(n) + 1 \leq c \cdot \log(n)$. Therefore, $T(t, m, m) = c' \cdot (n \cdot (\log(n) + 1)t \leq c \cdot n \log(n) \cdot t$. In our inductive step, we assume that the claim holds for all $T(t, r, m - 1)$, for all $1 \leq r \leq m - 1$, and want to prove that it holds for $T(t, r, m)$. If $r \neq 1$ and $r \neq m$, then the algorithm splits a matrix of size $t \cdot n$ and then calls two recursive instances. Thus,

$$\begin{aligned} T(t, r, m) &= c' \cdot nt + T(t, r - 1, m - 1) + T(t, r, m - 1) \\ &= c' \cdot nt + c \cdot n/2 \log(n/2)t + c \cdot (n/2 \log(n/2) \cdot t) \\ &\leq c'(nt + n \log(n)t - nt) = c \cdot n \log(n) \cdot t. \end{aligned}$$

This completes the proof of RECURSIVE(v, r). To complete the proof of the overall algorithm, notice that SUBADDITIVE is just calling RECURSIVE for $t = 1$ t times, and so does not increase the complexity of the overall algorithm. □

Chapter 6

Conclusions and Future Work

This thesis provides the necessary background needed to understand the basics of the field of error correcting codes and their generalized covering radius. Furthermore, we establish a new bound and compare it asymptotically to previously established bounds for the generalized covering radius of Reed-Muller codes. We further prove the existence of an algorithm to find vectors within this bound and show that the algorithm can be used to vastly improve the efficiency of the algorithm in (4). Furthermore, we show that our algorithm can likely be applied to find t-coverings for general Reed-Muller codes.

There are many areas of this thesis that can be easily expanded on in the future. Since so little work has been done in this field, the work is likely quite tractable. The most challenging area is likely any theoretical work regarding U_t of Reed-Muller codes because of the lack of any closed form of their generalized Hamming weight hierarchies. Potential future areas of work are listed below:

- Evaluating $U_t(C)$ for other codes which satisfy the chain condition and comparing to other known bounds.
- Exploring methods for computing $\Gamma(C)$ for codes which are not Reed-Muller codes, and furthermore exploring the complexity of computing $\Gamma(C)$ for Reed-Muller codes.
- Computational confirmation of the complexity of the T-COVER algorithm.
- Finding a general proof for the form of $\Gamma(RM(r,m))$ in line with Conjecture 17.

- Finding more upper and lower bounds on the generalized covering radius.
- Theoretical asymptotic analysis of U_t, especially for Reed-Muller codes, which would allow for a theoretical comparison with the bounds in Table 4.1.
- Comparison of performance of the modified Elimelech's covering algorithm to the performance of the original as discussed in Section 5.3.

Bibliography

[1] Chen, Chin-Long. 1969. Some results on algebraically structured error-correcting codes. ProQuest Dissertations and Theses 136. URL http://ccl.idm.oclc.org/login?url=https://www.proquest.com/dissertations-theses/some-results-on-algebraically-structured-error/docview/302432127/se-2?accountid=10141.

[2] Cohen, Gérard D., Iiro S. Honkala, Simon Litsyn, and Antoine Lobstein. 2005. Covering codes. In North-Holland mathematical library.

[3] Elimelech, Dor, Marcelo Firer, and Moshe Schwartz. 2021. The generalized covering radii of linear codes. In 2021 IEEE International Symposium on Information Theory (ISIT), 302–307. IEEE.

[4] Elimelech, Dor, Hengjia Wei, and Moshe Schwartz. 2021. On the generalized covering radii of reed-muller codes. 2107.09902.

[5] Graham, Ronald L., and N. J. A. Sloane. 1985. On the covering radius of codes. IEEE Trans Inf Theory 31:385–401.

[6] Hamming, R. W. 1950. Error detecting and error correcting codes. The Bell System Technical Journal 29(2):147–160. doi:10.1002/j.1538-7305.1950.tb00463.x.

[7] Janwa, H., and A. K. Lal. 2007. On generalized hamming weights and the covering radius of linear codes. In Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, eds. Serdar Boztaş and Hsiao-Feng (Francis) Lu, 347–356. Berlin, Heidelberg: Springer Berlin Heidelberg.

[8] Janwa, Heeralal, and Harold F. Mattson. 1999. Some upper bounds on the covering radii of linear codes over fq and their applications. Designs, Codes and Cryptography 18:163–181.

[9] MacWilliams, F.J., and N.J.A. Sloane. 1978. The Theory of Error-Correcting Codes. North-holland Publishing Company, 2nd ed.

[10] Mattson, H.F. 1983. An upper bound on covering radius. In Combinatorial Mathematics, eds. C. Berge, D. Bresson, P. Camion, J.F. Maurras, and F. Sterboul, North-Holland Mathematics Studies, vol. 75, 453–458. North-Holland. doi:https://doi.org/10.1016/S0304-0208(08)73421-2. URL https://www.sciencedirect.com/science/article/pii/S0304020808734212.

[11] Wei, V.K. 1991. Generalized hamming weights for linear codes. IEEE Transactions on Information Theory 37(5):1412–1418. doi:10.1109/18.133259.

[12] Wei, V.K., and Kyeongcheol Yang. 1993. On the generalized hamming weights of product codes. IEEE Transactions on Information Theory 39(5):1709–1713. doi:10.1109/18.259662.