SOME OPTIMAL CODES
AND
THEIR DECODING ALGORITHMS

TEE CHEE CHING

UNIVERSITI SAINS MALAYSIA

2008

SOME OPTIMAL CODES
AND
THEIR DECODING ALGORITHMS

by

TEE CHEE CHING

Thesis submitted in fulfillment of the
requirements for the degree of
Master of Science

March 2008

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank my supervisor, Dr. Ang Miin Huey for introducing me to Coding Theory. I am grateful to her support, guidance and supervision in my research, study and preparation of this thesis. I have learned a lot of technique to write a thesis systematically and also the skill to present my result of research from Dr. Ang. Lastly, I want to thank my family, all my friends and School of Mathematical Sciences, Universiti Sains Malaysia.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

BEBERAPA KOD OPTIMUM DAN ALGORITMA PENYAHKODANNYA

Abstrak

Dalam teori pengkodan, suatu kod-(n, M, d) atas $GF(2)$ yang baik seharusnya mempunyai n yang kecil untuk tujuan penghantaran mesej yang cepat; siaz M yang besar supaya dapat memancar lebih banyak mesej yang berlainan dan d yang besar supaya dapat membetulkan lebih banyak ralat. Walau bagaimanapun, ketiga-tiga matlamat ini adalah saling bercanggahan sesama diri. Usaha untuk mengoptimakan parameter M dengan nilai n dan d yang tetap, dirujukkan sebagai "masalah utama dalam teori pengkodan". Dengan menetapkan nilai n dan d, $A_2(n, d)$ mewakili nilai terbesar yang mungkin bagi M di mana wujudnya suatu kod-(n, M, d). Tambahan lagi, kod-(n, $A_2(n, d)$, d) adalah dinamakan sebagai kod optimum.

Dalam disertasi ini, pada mulanya kami membina satu kod binari optimum -(12, 144, 4) dengan menggunakan matrik. Algoritma penyakodan bagi kod-(12, 144, 4) ini akan dibincang secara keseluruhan dari beberapa aspek kebolehannya dalam mengesan dan membetulkan ralat. Kemudian dengan membuangkan kordinat terakhir bagi kod binari optimum-(12, 144, 4) ini, kami dapat membina suatu kod binari optimum-(11, 144, 3). Selain itu, dengan memilih beberapa kordinat yang sesuai untuk dibuang daripada kod binari optimum-(12, 144, 4) ini, kami membina satu kod binari-(9, 38, 3) dan satu kod binari optimum-(10, 72, 3). Bagi ketiga-tiga kod binary optimum yang terakhir, algorima penyahkodan pembetulan satu ralat mereka akan juga diberi dan dibincang.

Some Optimal Codes and Their Decoding Algorithms

Abstract

In coding theory, a good (n, M, d) - code over $GF(2)$, should have small n for fast transmission of messages; large M to enable transmission of a wide variety of messages and large d to correct more errors. However, these are contradicting objectives. The optimization of the parameter M for the given values of n and d is often referred to as “the main coding theory problem”. The largest value of M for a given values of n and d is denoted as $A_2(n, d)$ and the $(n, A_2(n, d)\ , d)$ – code is called an optimal code.

In this thesis, we first construct a (12, 144, 4)–optimal binary code using matrix. The decoding algorithms of this optimal code will be discussed thoroughly from different angles of its ability in detecting and correcting errors. Then by puncturing this code, we construct another (11, 144, 3)–optimal binary code. On the other hand, by choosing a few appropriate coordinates to be deleted from the (12, 144, 4)–optimal binary code, we construct a (10, 72, 3)–optimal binary code and a (9, 38, 3)–binary code. For the last three codes, their respective single error correcting decoding algorithms will also be given and discussed.

Chapter 1

Introduction

The theory of coding has been around since 1950's [14, 16]. Since the first day, all coding theorists are interested in knowing the answers to these questions:

i. How do we identify the parameters of a good code?

ii. Does a good code with the required parameters exist?

iii. Can we have a good decoding algorithm for the good codes?

The first two questions arose from the need of finding the "best" code whereas the answer for the last question is needed in the application of the code.

Each code has three important parameters, namely, the length *n*, the size *M* and the distance *d*. The value of *n* indicates the transmission speed of the respective code; the value of *M* indicates the number of distinct messages that can be digitalized and transmitted, whereas the value of *d* measures the ability of the respective code in correcting or detecting errors that occurred during transmission. Hence, it is obvious that a good code should have smaller *n* with larger *M* and *d* [24, 34]. However, in reality, these are conflicting aims and are often referred as the main coding theory problem.

A lot of work has been done to optimize one of the parameters *n*, *M* or *d* for a given value of the other two parameters and this has brought about the study of bounds in coding theory [6, 9, 12, 17, 21, 23, 29, 33, 35]. The usual version of the coding theory problem is to find $A_q(n,d) \in \square^+$ (or bounds of $A_q(n,d)$), where

$A_q(n,d)$ is the largest size among codes over F_q with given length *n* and distance *d* [9, 12, 17, 23, 29].

Codes over F_q of length *n*, distance *d* and size $A_q(n,d)$ are called optimal codes. For application purposes, some work has also been done to construct optimize codes [5, 7, 8, 28, 30, 31, 32]. One drawback of these results is that no discussion has been made regarding the decoding algorithm of the respective optimal non-linear codes, which plays a very important role in the application of the codes [2, 5, 6, 8, 9, 10, 11, 21, 23, 24, 25, 28, 29, 30, 31, 32, 34].

In this thesis, we first give a construction of an (12, 144, 4)–optimal binary code using matrix. By puncturing this code, we have an (11, 144, 3)–optimal binary code. On the other hand, using the (12, 144, 4) – optimal binary code, we construct an (9, 38, 3)–binary code and an (10, 72, 3)–optimal binary code. The decoding algorithm of these four codes will be discussed thoroughly from different angles with regards to their ability in detecting and correcting errors.

This thesis is divided into 7 chapters. Chapter 1 gives a brief background of our work area. Chapters 2 and 3 list relevant elementary theory of codes and linear codes that are needed in subsequent chapters. The family of binary Hamming codes and binary Golay code are discussed in chapter 3. In chapter 4, we discussed the main coding theory problem with some famous bounds of $A_2(n,d)$. We give the construction of optimal binary codes for distance 3 and length *n*, $n = 5,6,7,8,12,13,14,15,16$ in chapter 5. In chapter 6, we construct an (12, 144, 4) – optimal binary code by using a matrix and its decoding algorithm. By using this code, we can also obtain (11, 144, 3) − optimal code, (10, 72, 3) − optimal code

and (9, 38, 3)−binary code and their decoding algorithms, respectively. The last chapter gives the conclusion and postscript of the directions of future research.

Chapter 2

Getting Started With The Theory of Error-Correcting Codes

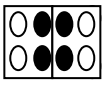
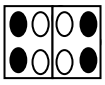
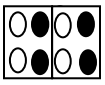
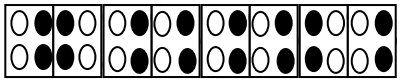
2.1 Elementary Concepts of Codes

Coding theory is a branch of mathematics that studies on how to improve the reliability of information or communication systems. Thus, all messages of a communication system need to be digitalized as the study involves mathematics. Basically, there are two types of communication, that is, communication between two different places and communication between the present and the future. An example of a communication between two places can be represented by the communication between two mobile phones whereas communication between present and future involves storing of present data and leaving it for the future. For the latter case, the most common way is to store data onto Compact Discs. We shall explain one example for each type of communication mentioned above.

Example 2.1.1

First, we discuss about storing of data onto computer hard disk where the data are written in the form of ion positive and ion negative. Assume that the data are musical notes. Let all musical notes be digitalized as below:

Do	1010
Re	1001
Mi	1000
Fa	1100
So	0100
La	1110
Ti	0001

Let  or  represents as 1 and  represents as 0 where ● represent ion positive and 0 represent ion negative. So  will be read as 1001 in digital form and thus it will be identified as Re. However, high magnetic field (referred to as noises in this case) will damage or change the data stored inside the hard disk. As a result, errors might occur when we retrieve the messages if the hard disk has been exposed to high magnetic field. #

Example 2.1.2

Alex wishes to send messages (using SMS) to his friend Lex who is in Johor. Basically he needs 26 letters of the alphabet (all capital letters) and also the spacing function between two words.

These letters and the spacing function can be represented by a string of '0's and '1's as shown below:

A	00001	O	01111
B	00010	P	10000
C	00011	Q	10001
D	00100	R	10010
E	00101	S	10011
F	00110	T	10100
G	00111	U	10101
H	01000	V	10110
I	01001	W	10111
J	01010	X	11000
K	01011	Y	11001
L	01100	Z	11010
M	01101	spacing	11011
N	01110		

The mobile phone transmits two different signals and let's assume that the signals are represented by image shapes like  and  . Let  represents 0 and  represents 1. Then, as an example, the letter Z will be transmitted in signal shapes like

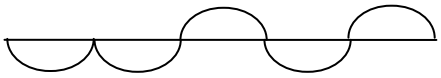


Figure 2.1: Signal shapes for alphabet Z

When the signal reaches Lex's mobile phone, the signal may become weak due to the long distance travelled (where the signal had to pass through high terrains or be interrupted by natural phenomena such as lightning or magnetic field, which are known as *noises* in coding theory). If we display the signal on the monitor, the received signal might look like

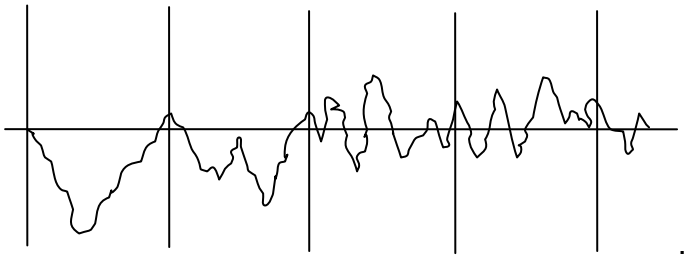


Figure 2.2: The received signal on monitor

In Figure 1.2, the first and second signals can be easily identified as 1 but it is hard for us to identify the third signal. Even though it is easy to identify from its shape, the fourth signal originally is 1 but due to noises, the received signal has misled us to recognize it as 0. #

Example 2.1.1 and Example 2.1.2 show that errors can occur during communication due to noises. In coding theory, we study methods to add protection to the transmitted or stored message and as a result to reduce the effect of errors that occurred during the transmission.

The “alphabet" and “musical notes" in Example 2.1.1 and Example 2.1.2 are *messages,* while the string of digits that represent each message is called a *message word.* In order to protect the message word, two important processes needed are given below:-

i) An encoding process to add shield to each message before it is transmitted.

ii) A decoding process to deduce the most likely word transmitted from each received word by detecting or correcting the errors occurred if its number is within certain limits.

Most of the transmitting devices are only capable of transmitting two types of signals (as shown in Example 2.1.2). Hence, from the point of application, all message words are strings of ‘0’s and ‘1’s where 0 and 1 are elements in $GF(2)$. Let $F_2 = GF(2)$ throughout this thesis.

Definition 2.1

Elements in F_2 are called *digits*. Any element of F_2^n is called a *word of length n.*

We write a word in F_2^n as $(a_0, a_1, ..., a_{n-1})$ or $a_0\, a_1 \cdots a_{n-1}$, whichever is more convenient. With reference to Example 2.1.1 and Example 2.1.2, all message words are respectively words of length 4 and 5. In general, it is clear that message words are elements of F_2^n for some n.

During the encoding process, each message word will be encoded into a *codeword*. The set of all the codewords is called a *code*. In general, the definition of a code is as follow:

Definition 2.2

A code over F_2 is called a *code of length n* if it is a non-empty subset of F_2^n.

As $F_2 = \{0,1\}$ throughout this thesis, all codes that are discussed are called binary codes. Below are two examples of encoding methods.

Example 2.1.3

In this example, we introduce an encoding method that involves matrix. Let $M = \{(1,\ 0,\ 1,\ 1), (0,\ 0,\ 1,\ 0), (1,\ 1,\ 0,\ 1), (0,\ 1,\ 1,\ 0)\}$ be the set of message words. Let $A = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$. We construct a code C by encoding each message word $m \in M$ into a codeword mA that is, $C = \{mA \mid m \in M\}$. Note that $|C| = 4$ and it is clear that C is a code of length 7 as it is a subset of F_2^7. #

Example 2.1.4

In this example, we give a simple encoding method. Let $M = \{01,10\}$ be the set of message words. Each of these message words will then be encoded into a

codeword by repeating itself 3 times as is given in Table 2.2. This encoding method gives us a code $C = \{010101,101010\}$ of length 6.

Table 2.1: Repetition code

Message word	Codewords
01	01 01 01
10	10 10 10

#

2.2 Concept of Error Detection and Error Correction

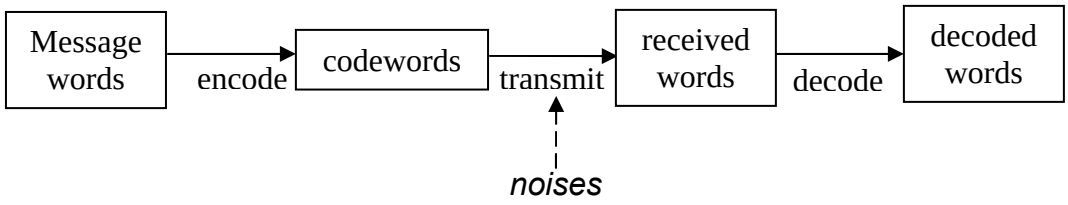


Figure 2.3: General idea of information transmission system

After the message words has been encoded, the codewords will be transmitted through a channel. The words received by the receiver are called *received words.* Cables and atmosphere are examples of *channel*. In reality, perfect channel does not exist due to the presence of noises and thus errors might occur during the transmission and the received words can be different from the transmitted codewords. In order to deduce the most likely codeword transmitted, all received words are decoded into decoded words using some decoding rules.

Let *C* be a code. Suppose that *w* is the received word. If *w* is a codeword in *C*, then we conclude that no error occurred during the transmission, otherwise we know that some errors have occurred. In the latter case, after the errors are detected, we need a decoding rule to deduce the most likely codeword transmitted.

To discuss further, we assume that the communication channel that we used is a binary symmetric channel (BSC). In a BSC,

i) Each digit that is transmitted has the same probability $p < \frac{1}{2}$ of being received in error, where *p* is called the crossover probability. This *p* reflects the level of noises in BSC. Thus, the bigger the value of *p*, the noisier the BSC.

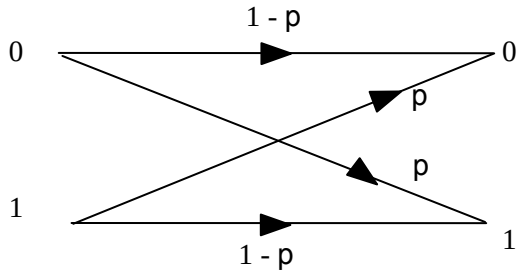


Figure 2.4: Binary Symmetric Channel

ii) Each transmission of a digit in F_2 is independent from the outcome of the previous transmission. Thus, the probability of received $w = x_1 x_2 \cdots x_n$ while $v = y_1 y_2 \cdots y_n$ is sent, is

$$\text{P}(x_1 x_2 \cdots x_n \text{ received} \mid y_1 y_2 \cdots y_n \text{ is sent}) = \prod_{i=1}^{n} \text{P}(x_i \text{ received} \mid y_i \text{ is sent}).$$

Assume that a codeword from a code *C* have been transmitted over a BSC with crossover probability *p* and *w* is received. Then, the most likely codeword transmitted is v_w such that

$$\text{P}(w \text{ received} \mid v_w \text{ is sent}) = \max_{v \in C} \{\text{P}(w \text{ received} \mid v \text{ is sent})\}.$$

Example 2.2.1

Suppose that a codeword from the code $C = \{010101, 101010\}$ in Example 2.1.4 is being sent over BSC with crossover probability $p = 0.05$. Assume that the word 010100 is received. We find the most likely codeword transmitted by computing the following probabilities:

$$\begin{aligned} P(010100\,\textit{received} \mid 010101\,\textit{sent}) &= P(0\,\textit{received} \mid 0\,\textit{sent})^3 \times P(0\,\textit{received} \mid 1\,\textit{sent}) \\ &\quad \times P(1\textit{received} \mid 1\,\textit{sent})^2 \\ &= (0.95)^3(0.05)(0.95)^2 \\ &= 3.8689 \times 10^{-2} \end{aligned}$$

$$\begin{aligned} P(010100\,\textit{received} \mid 101010\,\textit{sent}) &= P(0\,\textit{received} \mid 0\,\textit{sent}) \times P(0\,\textit{received} \mid 1\,\textit{sent})^3 \\ &\quad \times P(1\textit{received} \mid 0\,\textit{sent})^2 \\ &= (0.95)(0.05)^3(0.05)^2 \\ &= 2.96875 \times 10^{-7} \end{aligned}$$

Since the first probability is larger than the second one, we conclude that 010101 is the most likely codeword transmitted. #

From Example 2.2.1, it is clear that P(*w* received | *v* is sent) $= p^a(1-p)^{n-a}$ where *a* is the total number of positions in *v* and *w* which have different digits and *n* is the length of the code *C*. Thus, $n-a$ gives the total number of the positions in *v* and *w* which have the same digit. Since $p < \frac{1}{2}$, it follows that $1-p > p$ and thus, P(*w* received | *v* is sent) will be larger if $n-a$ is larger. Hence, P(*w* received | *v* is sent) will be maximum if *a* is the minimum, that is, the most

likely codeword transmitted is the codeword v_w in C where v_w has the least amount of position differ from w. This observation leads to the study of Hamming distance.

Definition 2.3

The *Hamming distance* between two codewords v_1 and v_2 of F_2^n is the number of positions in which they are different. It is denoted by $d(v_1, v_2)$.

It can be shown easily that Hamming distance fulfilled the following property [18, 24, 25].

Proposition 2.4

For any $v, u, w \in F_2^n$, $d(v,w) \leq d(v,u) + d(u,w)$.

Using Hamming distance, we have the following minimum distance decoding rule:

Minimum Distance Decoding (MDD) Rule

Suppose that codewords from a code C have being transmitted over a BSC and w is received. Then w will be decoded as v_w if v_w is the unique codeword in C that satisfies

$$d(w, v_w) = \min_{v \in C} d(w, v).$$

In general, there are 2 types of MDD

i) Complete minimum distance decoding (CMDD)

   If there are more than one v_w, select one of them arbitrarily.

ii) Incomplete minimum distance decoding (IMDD)

   If there are more than one v_w, request a retransmission.

Example 2.2.2

Let $C = \{000, 011\}$ be the code we used.

(i) Assume that $w = 100$ is received. It is clear that $w \notin C$. Hence, the errors are detected. To find v_w, we need to know $d(w, 000)$ and $d(w, 011)$. Note that $d(w, 000) = 1$ and $d(w, 011) = 3$. Hence, $v_w = 000$.

(ii) Assume that $w = 010$ is received. Clearly, $w \notin C$ and thus, we detect errors in w. Note that $d(w, 000) = d(w, 011) = 1$. If we use *CMDD*, then we may arbitrarily choose 000 or 011 as v_w. If we use the *IMDD* then retransmission will be requested. #

Assume that v in C is sent while w is received and t errors have occurred. Then, the MDD rule is said to be able to correct the errors if and only if v is the unique codeword in C such that $d(w, v) \leq t$.

2.3 The Minimum Distance of Codes

Every code has three important parameters, which are *n,* the length of the code, *m* the size of the code and *d* the minimum distance of the code. Below is the definition of the minimum distance of a code:

Definition 2.5

The *minimum distance* of a code C, denoted as $d(C)$, is defined to be the smallest distance among every two distinct codewords in C. That is,

$$d(C) = \min\{d(x, y) \mid x, y \in C, x \neq y\}$$

In short, we shall refer a code with length n and size m as a (n,m)-code. If the minimum distance of the code is known, say d, then we refer the code as a (n,m,d)-code.

The minimum distance of a code describes the ability of the code in detecting or correcting errors.

Definition 2.6

Let s be a positive integer. A code C is $s-$error detecting if, whenever s or fewer errors occurred during the transmission, the resulting received word is not a codeword in C.

Theorem 2.7

A code C is $s-$error detecting if and only if $d(C) \geq s+1$.

Proof :

Given that $d(C) \geq s+1$. Let $k \leq s$. Assume that a codeword v in C has been transmitted and k errors occurred. Thus $d(v,w) = k$ where w is the received word. If w is a codeword in C then

$$k = d(v,w) \geq d(C) \geq s+1.$$

This gives a contradiction, since $k \leq s$. Thus, $w \notin C$ and the errors are detected.

Conversely, assume that $d(C) = a < s+1$. Then there exist $v,w \in C, \quad v \neq w$ such that $d(v,w) = a$. Suppose v is transmitted and w is received. Then, a errors have occurred. As $w \in C$, these a errors escape from being detected. Thus, C is not s-error–detecting, as $a \leq s$. Ω

Definition 2.8

Let t be a positive integer. A code C is $t-error-correcting$ if whenever t or fewer errors occurred during the transmission, the MDD rule is able to correct the errors.

Theorem 2.9

A code C is $t-$error correcting if and only if $d(C)\geq 2t+1$.

Proof:

Given that $d(C)\geq 2t+1$. Assume that a codeword v in C is transmitted and w is received such that $d(v,w)=s,\ \ s\leq t$. Assume that there exist a $z\in C,\ z\neq v$, such that $d(z,w)\leq s$. Then by Proposition 2.4,

$$\begin{aligned} d(v,z) &\leq d(v,w)+d(w,z) \\ &\leq s+s \\ &\leq 2t \end{aligned}$$

which is a contradiction, since $d(C)\geq 2t+1$. Thus, for all $z\in C,\ z\neq v$, $d(w,z)>s$. Hence, MDD is able to correct the errors. So by Definition 2.8, C is t error correcting.

Conversely, assume that $d(C)<2t+1$. Let $d(C)=s\leq 2t$. Consider $t+1\leq s\leq 2t$. This is because if $s<t+1$, then C is definitely not t error correcting, as it is not even t errors detecting by Theorem 2.7. Let $v,z\in C,\ \ v\neq z$ such that $d(v,z)=s$. Permute the digits in v and z using a permutation σ to get v' and z' respectively such that only the first s positions of v' and z' are different as below

$$\begin{aligned} v' &= a_1....a_s a_{s+1}....a_n \\ z' &= b_1....b_s a_{s+1}....a_n \end{aligned}$$

that is, $d(a_1a_2\cdots a_s, b_1b_2\cdots b_s) = s$.

Let $w' = a_1....a_tb_{t+1}....b_sa_{s+1}....a_n$. Note that $d(w',z') = t$ and $d(w',v') = s-t$. Let w be the word that we get from w' by permuting the digits in w' using σ^{-1}. Then clearly

$$d(w,z) = d(w',z') = t$$
$$d(w,v) = d(w',v') = s-t$$

Assume that z is sent, t errors occurred and w is received. If $s < 2t$, we have $s-t < t$ and thus there exist $v, z \in C$, $v \neq z$ such that $d(w,z) \leq t$ and $d(w,v) \leq t$. Thus, the MDD rule cannot correct these t errors. If $s = 2t$, then $s-t = t$. Thus, $d(w,z) = d(w,v)$. Hence again, the MDD rule cannot correct these t errors. Thus by Definition 2.8, C is not t errors correcting. Ω

Theorem 2.7 and Theorem 2.9 discuss about the ability of detecting and correcting error of a code C separately. Thus if $d(C)=2s+1$, then we say that C is a $2s$-error detecting code by Theorem 2.7 or C is a s-error correcting code by Theorem 2.9. To discuss the ability of C in detecting and correcting error simultaneously, first we need the following modified MDD rule.

Modified MDD Rule (MMDD)

Suppose that codewords from a $(t+s)-error$ detecting and $t-error$ correcting code C have being transmitted over a BSC and w is received. If there exist a unique codeword v in C such that $d(v,w) \leq t$, then decode w as v. Otherwise, declare that uncorrected errors have occurred.

Using the MMDD rule, we are now ready to give the following definition:

Definition 2.10

Let t and s be two positive integers. A code C is $(t+s)-$*error detecting* and $t-$*error*$-$*correcting* if

i) whenever t or fewer errors occurred during the transmission, the MDD rule is able to correct the errors.

and ii) whenever d errors occurred during the transmission, $t < d \leq t+s$, then MMDD will not be able to decode the resulting received word.

Example 2.3.1

Let $C = \{000000, 101010, 010101, 111111\}$. Note that $d(C) = 3$. By Theorem 2.7 and Theorem 2.9, C is either a 2 errors detecting or 1 error correcting code. However, we shall show in this example that C is not a 2 errors detecting and 1 error correcting code.

Assume that $w = 100000$ is received when $v = 101010$ in C is sent. Clearly, two errors have occurred during the transmission. Note that by MMDD rule, w will be decoded as $u = 000000$ where $u \neq v$. Thus by Definition 2.10, C is not a 2 errors detecting and 1 error correcting code. #

Next theorem tells us when a code C can be a $(t+s)-$*error detecting* and $t-$*error correcting* code.

Theorem 2.11

A code C is $(t+s)-$*error detecting* and $t-$*error correcting* if and only if $d(C) \geq 2t+s+1$.

Proof :

Assume that $d(C) \geq 2t+s+1$. As $d(C) \geq 2t+1$, by Theorem 2.9, C is a t error correcting code and thus C is also a t error detecting code.

Assume a errors have occurred, where $t < a \leq t+s$, when v is transmitted and w is received. Hence, $t < d(v,w) = a \leq t+s$. We claim that there did not exist any $u \in C$ such that $d(u,w) \leq t$. Otherwise, we have

$$2t+s+1 \leq d(u,v) \leq d(u,w)+d(w,v) \leq t+s+t = 2t+s$$

which is a contradiction. Thus by MMDD rule, these a errors are detected without any correction as w will not be decoded. Hence, C is a t error correcting and $t+s$ error detecting code.

Conversely, assume that $d(C) \leq 2t+s$. We claim that C is not a t error correcting and $t+s$ error detecting code.

Suppose that $s=0$, then $d(C) \leq 2t$. Hence by Theorem 2.9, C is not a t error correcting code and thus is definitely not a t error correcting and $t+s$ error detecting code by Definition 2.10.

Suppose that $s>0$ and $d(C)=d$. Then $2t < d(C) = d \leq 2t+s$. Let $u,v \in C$, $u \neq v$, such that $d(u,v)=d$. Permute the digits in u and v using a permutation σ to get u' and v' respectively such that only the first d positions of u' and v' are different as following

$$u' = x_1 \cdots x_d a_1 \cdots a_m$$
$$v' = y_1 \cdots y_d a_1 \cdots a_m$$

Let $w' = x_1 \cdots x_t y_{t+1} \cdots y_d a_1 \cdots a_m$. Then

$$d(u',w') = d - t \ \ and \ \ d(v',w') = t .$$

Let w be the word that we get from w' by permuting the digit in w' using σ^{-1}. It is clear that $d(u,w) = d(u',w') = d - t$ and $d(v,w) = d(v',w') = t$. Note that $t < d(u,w) \le t + s$ since $2t < d \le 2t + s$. Suppose there exist another $u'' \in C$, $u'' \neq v$ such that $d(u'',w) \le t$. Then we have $2t < d \le d(u'',v) \le d(u'',w) + d(w,v) \le 2t$ which is a contradiction. Hence, if u is sent and w is received, where $d - t$ errors have occurred, $t < d - t \le t + s$, then w will be decoded as v by MMDD rule. Thus by Definition 2.10, C is not a t error correcting and $t + s$ error detecting code. Ω

To shorten the decoding process of a code, an efficient decoding algorithm which is designed according to the MMDD rule is needed. Example 2.3.2 gives two examples of decoding algorithm.

Example 2.3.2 :

Let $C = \{000000, 101010, 010101, 111111\}$. Note that $d(C) = 3$ and thus by Theorem 2.11, C is either a 2 errors detecting or a 1 error correcting and 1 error detecting code.

As a 1 error correcting and 1 error detecting code, C has the following decoding algorithm:

Assume that $w = a_1a_2a_3a_4a_5a_6$ is received and $u = b_1b_2b_1b_2b_1b_2$ is the corresponding decoded word.

Step 1: Consider a_1, a_3, a_5. Let b_1 be the digit that appears most frequently among a_1, a_3, a_5.

Step 2: Consider a_2, a_4, a_6. Let b_2 be the digit that appears most frequently among a_2, a_4, a_6.

As a 2 errors detecting code, C has the following decoding algorithm:

Assume that $w = a_1a_2a_3a_4a_5a_6$ is received. We now separate w to three parts, that is, a_1a_2, a_3a_4 and a_5a_6. If $a_1a_2 = a_3a_4 = a_5a_6$, we declare that no error occurred and w is the codeword that has been sent. If any portion of these three parts did not agree, we declare that errors have occurred. #

In general, it is not easy to design an efficient decoding algorithm for a code. There are some good codes such as the Reed Solomon Codes that still better decoding algorithm to be found.

Chapter 3

Linear Codes

3.1 Basic Properties of Linear Codes

Linear codes are also vector spaces. The vector space structure of linear codes make it easier to be described using its bases, easier to find its minimum distance and easier to design its encoding and decoding algorithms.

Definition 3.1

A subset C of F_2^n is a *linear code* if and only if C is a subspace of F_2^n.

The following theorem gives a way to determine whether a given code C is linear or not. The proof of this theorem can be found in [34]

Theorem 3.2

A nonempty subset C of F_2^n is a *linear code* if and only if

i) $u+v \in C$, for all u and v in C, and

ii) $au \in C$, for all $u \in C$, all $a \in F_2$.

Example 3.1.1

Let $C_1 = \{00,11\}$, $C_2 = \{00,10,01,11\}$, $C_3 = \{000,110,101,011\}$, $C_4 = \{1000,0111\}$ and $C_5 = \{0100,1100,1000\}$. By Theorem 3.2, it can be shown easily that C_1, C_2 and C_3 are linear codes whereas C_4 and C_5 are not linear codes as both do not fulfill the second condition of Theorem 3.2 when $a = 0$. #

Definition 3.3

Let $v \in F_2^n$. Then, the Hamming weight of *v*, denoted as $wt(v)$, is defined to be the number of nonzero positions in *v*, that is, $wt(v) = d(v, \mathbf{0})$ where $\mathbf{0}$ is the zero vector in F_2^n.

Theorem 3.4

If $x = (x_1, x_2, \ldots, x_n), y = (y_1, y_2, \ldots, y_n) \in F_2^n$, then $d(x, y) = wt(x + y)$.

Proof :

Let $x = (x_1, x_2, \ldots, x_n), y = (y_1, y_2, \ldots, y_n) \in F_2^n$. Thus,

$$\begin{aligned} wt(x + y) &= \text{total number of nonzero positions in } x + y \\ &= \text{total number of } x_i + y_i = 1 \ \forall i = 1, 2, \ldots, n \\ &= \text{total number of positions where x and y are different} \\ &= d(x, y) \end{aligned}$$

Ω

Theorem 3.5

Let $x = (x_1, x_2, \ldots, x_n), y = (y_1, y_2, \ldots, y_n) \in F_2^n$, then

$$d(x, y) = wt(x) + wt(y) - 2wt(x * y)$$

where $x * y = (x_1 y_1, x_2 y_2, \cdots\cdots, x_n y_n)$.

Proof:

Note that $wt(x * y)$ is the total number of positions where both x and y have digit 1 at the same position. Hence, it is clear that $wt(x + y) = wt(x) + wt(y) - 2wt(x * y)$.

As $d(x, y) = wt(x + y)$, the result follows. Ω

As given in the following theorem, we can use *Hamming weight* to find the minimum distance of linear codes. This makes the minimum distance of linear codes easier to determine.

Theorem 3.6

Let C be a linear code with $wt(C) = \min\{wt(v) \mid v \in C, v \neq \mathbf{0}\}$. Then, $d(C) = wt(C)$.

Proof:

By the definition of minimum distance of code, there exist codewords $x, y \in C$, $x \neq y$, such that $d(C) = d(x, y)$. Then, by Theorem 3.4,

$$d(C) = d(x, y) = wt(x + y) \geq wt(C).$$

On the other hand, for some codeword $x \in C$, we have $wt(C) = wt(x) = d(x, \mathbf{0}) \geq d(C)$ as $\mathbf{0} \in C$. Hence, $d(C) = wt(C)$. Ω

As vector spaces, linear codes can be easily described using their bases.

Example 3.1.2

From Example 3.1.1, we know that $C_1 = \{00, 11\}$, $C_2 = \{00, 10, 01, 11\}$ and $C_3 = \{000, 110, 101, 011\}$ are linear codes. It is obvious that $C_1 = \langle 11 \rangle$, $C_2 = \langle 10, 01 \rangle$ and $C_3 = \langle 110, 101 \rangle$. Note that $\alpha_1 = \{11\}$, $\alpha_2 = \{10, 01\}$ and $\alpha_3 = \{110, 101\}$ are linearly independent and thus they are respectively basis of C_1, C_2 and C_3. Thus, $\dim(C_1) = 1$, and $\dim(C_2) = \dim(C_3) = 2$. #

3.2 Generator Matrix and Parity Check Matrix

Let C be a length n and k-dimensional linear code. Then, $|C| = 2^k$. Thus, C is a $(n, 2^k)$-linear code. However, it is more often to denote a length n and k-dimensional linear code as a $[n,k]$-code. If the minimum distance of C is known as d, then we denote C as a $[n,k,d]$-code.

Let $\alpha = \{v_1, v_2, \ldots, v_k\}$ be a basis of a $[n,k]$-code C. Thus,

$$C = \langle \alpha \rangle = \{b_1 v_1 + b_2 v_2 + \cdots + b_n v_n \mid b_1, b_2, \cdots b_n \in F_2\}.$$

While designing an encoding algorithm of C, α is often written as $k \times n$ matrix form, $G = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{pmatrix}$ where v_i is the i^{th} row of G. Then we can design the encoding of C using the following one-to-one linear transformation T.

Define $T : F_2^{\,k} \longrightarrow F_2^{\,n}$ such that

$$\left((a_1, a_2, \ldots, a_k)\right)T = (a_1, a_2, \ldots, a_k)G = (a_1, a_2, \ldots, a_k)\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{pmatrix} = a_1 v_1 + a_2 v_2 + \cdots + a_k v_k \in C.$$

Hence, we have $C = \mathrm{Im}(T)$ if $F_2^{\,k}$ is used as the set of message words. Reflecting by its role, the matrix G is given a name as follows

Definition 3.7