TU Delft

Delft University of Technology

Faculty of Electrical Engineering, Mathematics and Computer Science
Wireless and Mobile Communications

Master Thesis

New Upper Bounds on the Separating Redundancy of Linear Block Codes

Ngo Minh Tri

Committee members:
Supervisor: Dr.ir. J.H. Weber
Members: Prof.dr.ir. I.G.M.M. Niemegeers, Dr.ir. R.E. Kooij

June, 2009

Acknowledgements

First and foremost, I dedicate this thesis to my parents and my brother, who always give me support and encouragement. They are the motivation for me to reach this far.

I would like to take this opportunity to express my appreciation to my supervisor Jos H. Weber. I am grateful for his dedicated involvement in my thesis work. By frequent discussions and detailed corrections, he helped me to judge both ideas and writing skills, which led to significant improvements in my thesis.

I would like to thank the Netherlands Government who offered me the Netherlands Fellowship Programs scholarship to pursue the Master's degree at Delft University of Technology. These two years are wonderful and unforgettable.

I also send special thanks to Minh and Dony, who have been the most loyal friends. They are simply friends in need.

Besides, I also thank all students in Vietnamese community in Delft, especially Huy, Dang, Thuan…, for having created warm, relaxed and friendly activities.

Ngo Minh Tri
Delft, 2009

Abstract

Most decoding algorithms of linear codes, in general, are designed to correct or detect errors. However, many channels cause erasures in addition to errors. In principle, decoding over such channels can be accomplished by deleting the erased symbols and decoding the resulting vector with respect to a punctured code. For any given linear code and any given maximum number of correctable erasures, in the paper "*Separating Erasures from Errors for Decoding*", [1], Abdel-Ghaffar and Weber introduced parity-check matrices yielding parity-check equations that do not check any of the erased symbols and which are sufficient to characterize all punctured codes corresponding to this maximum number of erasures. This allows for the separation of erasures from errors to facilitate decoding. Typically, these parity-check matrices have redundant rows. To reduce decoding complexity, parity-check matrices with small number of rows are preferred.

The minimum number of rows in a parity-check matrix separating all erasure sets of size at most l is called the lth separating redundancy. In [1], upper and lower bounds on the lth separating redundancy were presented. In this thesis, we give improvements on upper bounds from [1].

Contents

List of Tables

List of Figures

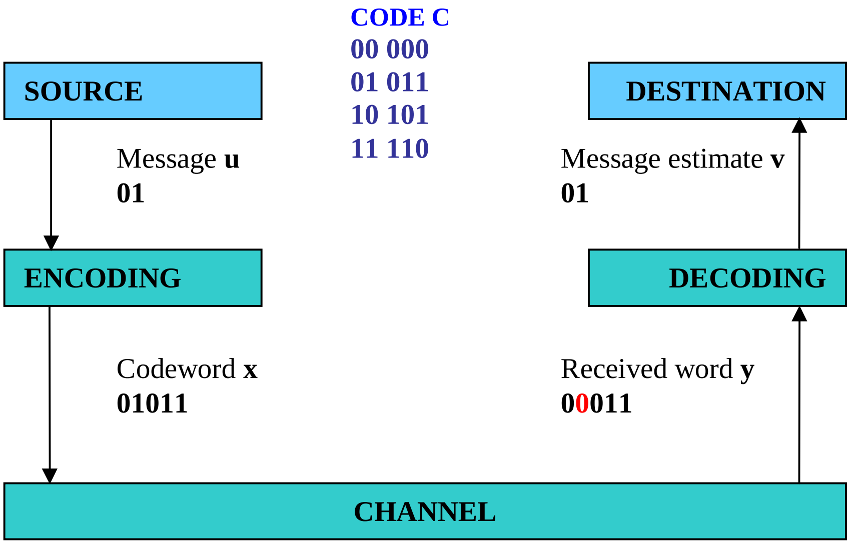
Chapter 1

Introduction

1.1 Error-Correcting Codes

A message generated by a source consists of symbols from an alphabet of a field of order q, $GF(q)$. Symbols can be erroneous due to noisy channel. Error means the received symbol is different from the transmitted symbol. In order to protect data against errors which can occur during transmission, channel coding techniques are required. In error correction techniques, a message of k symbols will be encoded into a codeword of n symbols $(n > k)$. The collection of these codewords forms a code.

Here, we consider an example in which a simplified scheme of a communication system consists of these terms: source, channel encoding, channel, channel decoding and destination.



A source generates a message **u** containing two symbols from $GF(2)$, the binary alphabet. There are four possible messages: **00**, **01**, **10**, and **11**. Each message is encoded into a codeword **x**, in this example by adding other three symbols to **u**, leading to a sequence of five symbols, according to a code C as indicated. Assume that we send the codeword **01011** over a noisy channel. At the receiving side, the received word **y** is **00011**. The second symbol is erroneous. The decoder needs to produce a message estimate **v**, based on its knowledge of the code C and the received word **y**. Here, the decoder chooses the codeword **01011**, which resembles **y** the most, and thus **v** is **01**, which is indeed the original message. Hence, the error has been corrected.

Note that without using a code, any error cannot be corrected or detected. Error-correcting codes are applied in situations where retransmissions are relatively costly or impossible. Using the code makes the system more reliable. However, transmitting more symbols results in the cost of higher bandwidth requirements.

In recent years, due to the mergence of large-scale, high speed data networks for the exchange, processing, and storage of digital information in the commercial, governmental, and military spheres, error-correcting codes play an important role on improving the reliability of such communication systems.

The use of a parity-bit as an error-detecting mechanism is one of the simplest and most well-known schemes used in association with computers and computer communication. Data is portioned into blocks. To each block, an additional bit is appended to make the number of bits which are 1 in the block, including the appended bit, an even number. If a single bit-error occurs, within the block, the number of 1's becomes odd. Hence, this allows for detection of single errors.

The most applications of error-correcting codes are in telecommunications. Many early applications of coding were developed for deep-space and satellite communication systems. For example, satellite photos were taken in space and sent back to earth. The channel for such transmission is space and the earth's atmosphere. These communication systems have limitations on their transmitted power. Solar activity and atmospheric conditions can introduce errors into weak signals coming from the spacecraft. Error-correcting codes are an excellent mean of reducing power needs because the reliable communications can be achieved even when the information is weakly received at its destination. With the applications of error-correcting codes, most of the pictures sent could be correctly recovered here on earth. As examples, a binary (32,6,16) Reed-Muller code was used during the Mariner and Viking mission to Mars around 1970 or a convolutional code was used on the Pioneer 10 and 11 missions to Jupiter and Saturn in 1972. The (24,12,8) Golay code was used in the Voyager 1 and Voyager 2 spacecrafts transmitting color pictures of Jupiter and Saturn in 1979 and 1980. When Voyager 2 went on to Uranus and Neptune, the code was switched to a concatenated Reed-Solomon code-Convolutional code for its substantially more powerful error correcting capabilities.

The block and convolutional codes are also applied to the Global System for Mobile communications (GSM) which is the most popular digital cellular mobile communication system while CDMA2000 used turbo codes. Reed Solomon and Viterbi codes have been used for nearly 20 years for the delivery of digital satellite TV.

Besides, these techniques may also be applied to most storage devices to protect against damage to the stored data. The transmission and storage of digital information have much in common. Both processes transfer data from an information source to a destination. However, instead of transporting data from one place to the other, storage may be considered as transport through time. As an example, in Compact (CD) system, the sound encoded into data bits and modulated into channel bits is sent along the "transmission channel" consisting of write laser, master disc, user disc and optical pickup. Imperfections on the disc will produce errors in the recovered data. Block codes are often used in data storage applications. A "parity track" was present on the first magnetic tape data storage in 1951. The most notable is Reed-Solomon codes because of their widespread uses on the Compact disc, the DVD, and in computer hard drives. Hamming codes, which are single error-correcting or double error-detecting, is commonly used to correct NAND flash memory errors. Modern hard drives use CRC codes to detect and Reed-Solomon codes to correct minor errors in sector reads, and to recover data from sectors that have "gone bad" and store that data in the spare sectors. Computers have error-correcting capabilities built into their random access memories.

Low-density parity-check codes (LDPC codes) are now used in a many recent high-speed communication standards, such as DVB-S2 (Digital video broadcasting), WiMAX (IEEE 802.16e standard for microwave communications), 10GBase-T Ethernet…

In all cases, error-correcting codes ensure proper performance of the systems. They permit communication links to function reliably in the presence of noise, distortion, and interference.

1.2 Linear Block Codes

Let C be an $[n,k,d]$ linear block code over $GF(q)$. It means that C is a k-dimensional subspace of the n-dimensional vector space over an alphabet of size q. The elements of the code C are called codewords. Messages generated by the source are one-to-one mapped to codewords. Hence, the number of codewords, denoted by $|C|$, is also the number of messages. k represents the length of the message generated by the source and n represents the length of the codeword to be transmitted over the channel. d is the Hamming distance of the code C which is the smallest Hamming distance between any two different codewords. The Hamming distance between two vectors of the same length is defined as the number of positions in which these two vectors differ. The Hamming distance between a vector and the all-zero vector is called the weight of the vector. The Hamming distance d is an important parameter of a code C. A code with Hamming distance d can correct $\lfloor (d-1)/2 \rfloor$ or detect $d-1$ errors.

The set of codewords of C can be defined as the null space of the row space of an $r \times n$ parity-check matrix $H = (h_{i,j})$ of rank $n-k$. The row space of H is the $[n, n-k, d^{\perp}]$ dual code $C^{\perp}$ of C. Because a q-ary $\mathbf{x}$ is a codeword of C if and only if $\mathbf{x}H^{T} = 0$, where the superscript T denotes the transpose, from the parity-check matrix H, we can form r parity-check equations, denoted by

$$\mathrm{PCE}_i : \sum_{j=1}^{n} h_{i,j} x_j = 0 \text{ for } i = 1, 2, ..., r.$$

An equation $\mathrm{PCE}_i(\mathbf{x})$ is said to check $\mathbf{x}$ in position j if and only if $h_{i,j} \neq 0$.

1.3 Erasures

Sometimes, at the receiver, the demodulator cannot decide which symbol the received waveform represents. In this case, we declare the received symbol as an erasure. When the received codeword contains erasures instead of errors, exhaustive decoding or the concept of iterative decoding can be applied ([2]).

Here, we consider an example of iterative decoding procedure using the (7,4,3) binary Hamming code with this parity-check matrix,

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

Because a binary vector $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ is a codeword if and only if $\mathbf{x}H^T = 0$. Hence, any codeword satisfies three parity-check equations formed from the parity-check matrix,

$$\begin{aligned} &\text{A: } x_1 + x_3 + x_4 + x_5 = 0 \\ &\text{B: } x_2 + x_4 + x_5 + x_6 = 0 \\ &\text{C: } x_3 + x_5 + x_6 + x_7 = 0. \end{aligned}$$

Equation A is said to check on x_1, x_3, x_4 and x_5. If exactly one of these four symbols is erased, it can be retrieved from this equation. Here, we denote the erased symbol at the receiver side by the symbol *. For example, if the received vector is **010*0, it follows from the equation A that $x_1 = 1$ because $x_3 = 0$, $x_4 = 1$ and $x_5 = 0$. Equation B checks two erased symbols in the position 2 and 6. Hence, none of these erased positions can be retrieved immediately. However, if one of these two erased symbols has been retrieved from the other equation, this equation can be used again to retrieve the one remaining erasure. In this example, equation C gives $x_6 = 0$, and then by returning to equation B, we obtain $x_2 = 1$. The transmitted codeword 1101000 has been retrieved by iterative decoding using equations A, B and C. In general, we could keep on using the parity-check equations iteratively until none of them checks on exactly one erased symbol (more information on iterative decoding can be found in [4], [6]). Erasure decoding is successful if and only if erasures do not fill the positions of a nonempty stopping set. A stopping set is a set of positions of symbol in which there is no parity-check equation that checks exactly one symbol in these positions. The performance of iterative decoding techniques for linear block codes correcting erasures depends on the sizes of the stopping sets associated with the parity-check matrix representing the code. The choice of the parity-check matrix of the code can affect the sizes of stopping sets. The parity-check matrix with redundant rows can benefit decoding performance while increasing decoding complexity. More information on stopping set can be found in [5], [9], [10].

1.4 Separation of Errors from Erasures

Many channels cause erasures in addition to errors. In case errors combine with erasures, we can apply the algorithm, which is applicable to linear codes, use trials in which erasures are replaced by symbols in $GF(q)$ and the resulting vector is decoded using a decoder capable of correcting or detecting errors only. For binary code, two trials are sufficient for decoding. For example, if C is a binary (n,k)-code having distance $d = 2t_{\#} + t_{?} + 1$, then C can correct $t_{\#}$ errors and $t_{?}$ erasures. In the presence of no erasures, C will correct up to $t_{\#} + \lfloor t_{?} / 2 \rfloor$ errors. Let **r** be a received vector having at most $t_{\#}$ errors and at most $t_{?}$ erasures. Suppose the decoder forms two vector $\mathbf{r}_0$ and $\mathbf{r}_1$, where $\mathbf{r}_i$ is obtained from **r** by filling all erasure positions with the symbols $i, i = 0,1$. Since C is binary, in one of $\mathbf{r}_0$ and $\mathbf{r}_1$, at least half the erasure locations have correct symbols. And hence at least one of $\mathbf{r}_0$ and $\mathbf{r}_1$ has distance at most $t_{\#} + \lfloor t_{?} / 2 \rfloor$ from the transmitted codeword. Any standard error correction technique will now correct one of these vectors to the transmitted codeword. If the standard technique decodes both $\mathbf{r}_0$ and $\mathbf{r}_1$ to codewords, and these codewords are the same, then this is the transmitted codeword. If they are different, then that one (and there will be only one) requiring at most $t_{\#}$ changes to non-erasure positions is the desired codeword. Because the number of trials, the steps of filling values in the erasure positions, depends on q, this algorithm is practical only for $q^{t_{?}}$ relatively small. The trials increase rapidly with q restricting the application of this method to codes over large fields. In this thesis, we do not focus on this algorithm. More information on this algorithm can be found in [13].

[1] proposed another way of decoding over such channels. First, all erasures are deleted from the received message. Errors in the resulting codeword will be corrected or detected based on the punctured code whose codewords consist of symbols in positions which are not erased. After all errors have been corrected, the erasures will be recovered by iterative decoding.

If the number of erasures, $t_{?}$, does not exceed $d-1$, which is the maximum number of erasures allowed in a codeword, then at the decoder, we can choose two nonnegative integers $t_{\neq}$ and $t_{!}$ satisfying $t_{?} + 2t_{\neq} + t_{!} \le d-1$ such that the following is true. If the number of errors does not exceed $t_{\neq}$, then the decoder can correct all errors and erasures. Otherwise, if the number of errors is greater than $t_{\neq}$ but at most $t_{\neq} + t_{!}$, then the decoder can detect the occurrence of more than $t_{\neq}$ errors.

The decoder can compute a parity-check matrix for the punctured code after receiving the codeword. However, this leads to time delay which is unacceptable in some applications. To reduce time delay, we can store parity-check matrices of all punctured codes corresponding to all erasure patterns. The drawback of this solution is the requirement of huge memory storage at the decoder.

[1] proposed a useful method which uses the separating matrix with possibly redundant rows, providing enough parity-check equations which do not check any of the erased symbols and are sufficient to form a

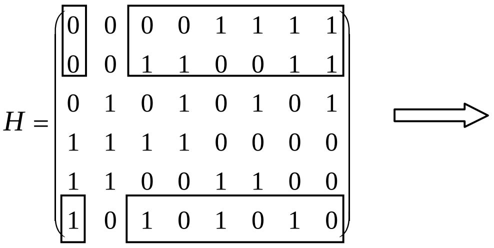
parity-check matrix for the punctured code obtained by deleting the erasures. Having these parity-check equations not checking any of the erased symbols lead to the concept of separation of errors from erasures.

The basic concept of this decoding technique can be illustrated by an example as follows.

We consider an [8,4,4] binary extended Hamming code with parity-check matrix,

$$H = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hdashline 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

In a normal parity-check matrix, we just have four rows as the first four rows in this matrix. In this example, we add two other rows. Allowing redundant rows simplifies the decoding of erasures in addition to errors. Assume that at the decoder, we receive a codeword **r** = 0*011000 with one erasure in the position two. Applying the decoding technique mentioned above, firstly we delete the erasure and obtain the resulting vector **r’** = 0011000. We can consider **r’** as a codeword of the (7,4,3) punctured code. In the parity-check matrix H , the first, the second and the sixth row have zeros in the position two. It means that three related parity-check equations do not check the erased symbol. From these three rows, we can form a parity-check matrix H' for the punctured code.



$$H' = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Using H', we can decode **r’** into (0011010). After updating **r** to (0*011010), the third row of H , which checks the erased symbol, can be used to recover the erasure. The transmitted codeword corresponding to **r** is (01011010).

In this case, a normal parity-check matrix cannot be used for decoding of both errors and erasures. Decoding will be easier if we pay the price of storing parity-check matrix with more rows than necessary. In order to reduce memory storage as well as decoding complexity, parity-check matrices with small number of rows are preferred.

1.5 Problem Statement

For any given linear code and any given maximum number of correctable erasures, in the paper "*Separating Erasures from Errors for Decoding*" , [1], Abdel-Ghaffar and Weber introduce parity-check matrices yielding parity-check equations that do not check any of the erased symbols and which are sufficient to characterize all punctured codes corresponding to this maximum number of erasures. This allows for the separation of erasures from errors to facilitate decoding. These parity-check matrices typically have redundant rows. The authors of [1] also give two constructions of such matrices and prove general bounds on their minimum sizes. These techniques used are related to methods used to prove results on stopping sets ([9], [10]).

The general upper bounds and lower bound on the minimum number of rows in a parity-check matrix with certain separation properties given in [1] are rather far apart. In this thesis, we give improvements on the upper bounds from [1]. The rest of this thesis is organized as follows. The summarization of important points in [1] is covered in Chapter 2. Besides, in this chapter, we also consider the number of useful rows in the matrices built by the second construction from [1]. Chapter 3 and Chapter 4 introduce two methods which can construct such matrices with a smaller total number of rows. Comparisons between upper bounds will be given in Chapter 5 while Chapter 6 concludes the thesis.

Chapter 2

Separating Matrices and Separating Redundancy

2.1 Set Separation

Let $H=(h_{i,j})$ of rank $n-k$ be an $(r\times n)$ parity-check matrix of C, $r\geq n-k$. Let $\mathcal{S}$ be a subset of $\{1,2,...,n\}$ and $\mathcal{T}$ be a subset of $\{1,2,...,r\}$, define $H_{\mathcal{S}}^{\mathcal{T}}=(h_{i,j})$ with $i\in\mathcal{T}$ and $j\in\mathcal{S}$, be a $|\mathcal{T}|\times|\mathcal{S}|$ submatrix of H. For the code C of length n, define $C_{\overline{\mathcal{S}}}=\left\{c_{\overline{\mathcal{S}}}:c\in C\right\}$ be the punctured code consisting of all codewords of C in which the symbols in positions indexed by $\mathcal{S}$, $\mathcal{S}=\{1,2,...,n\}\backslash\overline{\mathcal{S}}$ are deleted. Clearly, $C_{\overline{\mathcal{S}}}$ is a linear code over $GF(q)$ of length $n'=|\overline{\mathcal{S}}|$, dimension $k'\leq k$, and Hamming distance $d'\geq d-|\mathcal{S}|$. Let $\widetilde{\mathcal{S}}=\left\{i:1\leq i\leq r,h_{i,j}=0 \quad \forall j\in\mathcal{S}\right\}$, define $H(\mathcal{S})=H_{\overline{\mathcal{S}}}^{\widetilde{\mathcal{S}}}$.

Definition: *A parity-check matrix* H *separates* $\mathcal{S}\subseteq\{1,2,...,n\}$ *if and only if* $H(\mathcal{S})$ *is a parity-check matrix of* $C_{\overline{\mathcal{S}}}$.

Here, we consider again the example mentioned in Chapter 1. We consider an [8,4,4] binary extended Hamming code with this parity-check matrix,

$$H=\left(\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1\\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1\\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1\\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0\\
\hline
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0\\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0
\end{array}\right)$$

Assume that at the decoder, we receive a codeword **r** = 0*011000 with one erasure in the position two. After deleting the erasure, we obtain the resulting vector **r’** = 0011000. We can consider **r’** as a codeword of the (7,4,3) punctured code. In the parity-check matrix H, the first, the second and the sixth row have zeros in the position two. It means that three related parity-check equations do not check the erased symbol. From these three rows, we can form a parity-check matrix H' for the punctured code.

$$H=\begin{pmatrix} \boxed{0} & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ \boxed{0} & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ \boxed{1} & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \Longrightarrow H'=\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Hence, we can say that the parity-check matrix H separates the set {2}.

Theorem 1 ([1]): *A parity-check matrix H of an $[n,k,d]$ linear code C separates a set $\mathcal{S}$ of size $|\mathcal{S}| \le d-1$ if and only if $H(\mathcal{S})$ has rank $n-k-|\mathcal{S}|$.*

2.2 Separating Matrix

2.2.1 Definitions

Let H be a parity-check matrix of an $[n,k,d]$ linear code over $GF(q)$.

- *H is l-separating of C if it separate every set $\mathcal{S}$ of size $|\mathcal{S}| = 0,1,\ldots,l$ with $0 \le l \le \min\{d, n-k\}-1$.*
- *Let C be an $[n,k,d]$ MDS linear code over $GF(q)$, i.e., $d = n-k+1$. Then any parity-check matrix, H , of C separates all sets of size $d-1$. In particular, any $(d-2)$-separating parity-check matrix of C is $(d-1)$-separating.*

Theorem 2 ([1]): *If H separates all sets of size l for a fixed $l \le \min\{d, n-k\}-1$, then it is l-separating.*

2.2.2 Characteristics

If H is an *l*-separating parity-check matrix of the code C , from H , we can form parity-check matrices of all codes punctured up to a fixed number of symbols, denotes by l . H has two features:

- H can separate erasures from errors because H has enough parity-check equations which do not check any erased symbols and are sufficient to characterize the punctured code. It means that the punctured codeword, which is formed by deleting the erased symbols, can be corrected or detected errors in it by a sub-matrix of H .
- In case $0 \le l \le \min\{d, n-k\}-1$, H has no nonempty stopping set of size l or less. For any pattern of l or fewer erasures, not only are there enough parity-check equations not checking any of the erased symbols that characterize the punctured code, but also there is a parity-check equation that checks exactly one of the erased symbols. It means that after all errors have been corrected, the erasures can be recovered by iterative decoding procedure.

2.2.3 The necessary and sufficient condition for a parity-check matrix to be an *l*-separating matrix

Basing on definitions and theorems, we can conclude that H is an l -separating matrix of the code C if

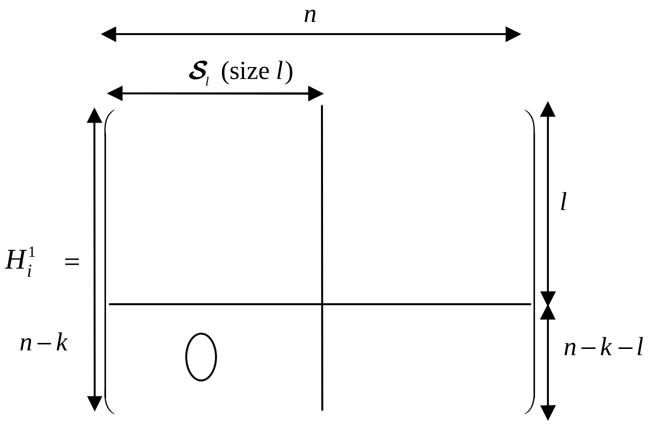
- In case $l \le d-2$ or $l = d-1 \le n-k-1$: $H(\mathcal{S})$ has rank $n-k-l$ for each of all sets $\mathcal{S}$ of size l ,
- In case $l = d-1, d = n-k+1$, $H(\mathcal{S})$ has rank $n-k-(l-1)$ for each of all sets $\mathcal{S}$ of size $l-1$.

2.3 The l-Separating Parity-Check Matrix Constructions

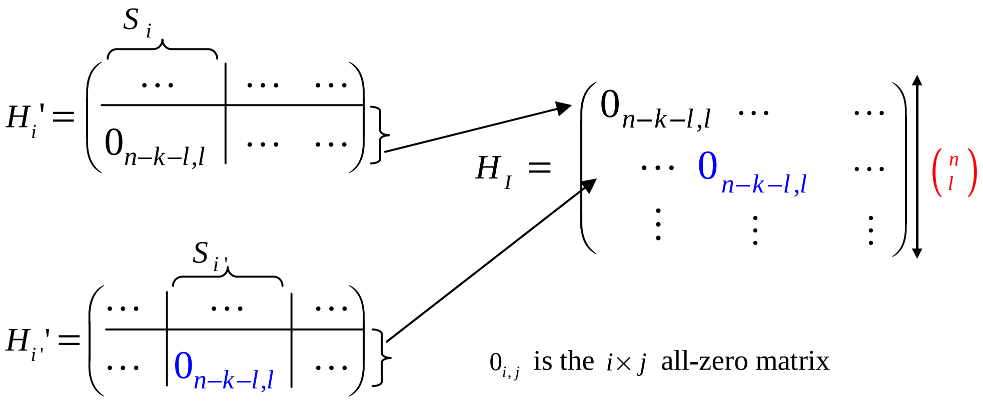
2.3.1 The first construction

- If $l \le d-2$ or $l = d-1 \le n-k-1$

Let H' be a full rank parity-check matrix, $\mathcal{S}_i \subseteq \{1,2,...,n\}$, where $i = 1,2,...,\binom{n}{l}$, be the distinct subsets of $\{1,2,...,n\}$ of size l . For each $i = 1,2,...,\binom{n}{l}$, $H'_{\mathcal{S}_i}$ has rank $l\ (l \le d-1)$. By elementary row operations on H' , we can obtain an $(n-k)\times n$ matrix, H_i^1, for each $i = 1,2,...,\binom{n}{l}$, of rank $n-k$ such that its last $n-k-l$ rows have zeros in the positions indexed by $\mathcal{S}_i$.



Let H_I be the matrix whose set of rows is the union of the sets of the last $n-k-l$ rows in H_i^1 for $i = 1,2,...,\binom{n}{l}$. H_I is an *l*-separating matrix of the code C , ([1]), and it has at most $\binom{n}{l}(n-k-l)$ rows.



$0_{i,j}$ is the $i \times j$ all-zero matrix

- If $l = d-1, d = n-k+1$

Instead of $\mathcal{S}_i$ of size l, apply the same procedure with subsets $\mathcal{S}_i \subseteq \{1,2,...,n\}$, where $i = 1,2,...,\binom{n}{l-1}$, of size $l-1$. In this case, H_I has at most $\binom{n}{l-1}(n-k-(l-1)) = \binom{n}{l-1}$ rows.

2.3.2 The second construction

Normalized vector: A nonzero vector $\mathbf{x} = (x_1, x_2, ..., x_n)$ over $GF(q)$ is said to be normalized if its leading nonzero term is equal to 1. The weight of the vector **x** is $|\{j : x_j \neq 0\}|$.

- If $l \leq d-2$ or $l = d-1 \leq n-k-1$

Let A be a matrix over $GF(q)$ whose rows are all the nonzero normalized vectors of length $n-k$ and weight at most $l+1$. Define $H_{II} = AH'$. H_{II} is an *l*-separating matrix of the code C, ([1]), and it has $\sum_{i=1}^{l+1}\binom{n-k}{i}(q-1)^{i-1}$ rows.

- If $l = d-1, d = n-k+1$

Apply the same procedure, but A is a matrix over $GF(q)$ whose rows are all the nonzero normalized vector of length $n-k$ and weight at most l. H_{II} has $\sum_{i=1}^{l}\binom{n-k}{i}(q-1)^{i-1}$ rows.

2.4 Separating Redundancy and the Upper Bounds

Define the *l*th separating redundancy, s_l, of the code C to be the minimum number of rows in an *l*th separating parity-check matrix of C.

Therefore, the upper bounds can be derived from two above constructions.

- **The first construction**

$$s_l \le \begin{cases} \binom{n}{l}(n-k-l) & \text{if } l \le d-2 \text{ or } l = d-1 \le n-k-1, \\ \binom{n}{l-1} & \text{if } l = d-1, d = n-k+1. \end{cases} \quad (2.1)$$

Basing on the first construction, in Chapter 3 and Chapter 4, we will introduce two methods which can construct l-separating matrices of the code C with the smaller numbers of rows than in (2.1).

- **The second construction**

$$s_l \le \begin{cases} \sum_{i=1}^{l+1}\binom{n-k}{i}(q-1)^{i-1} & \text{if } l \le d-2 \text{ or } l = d-1 \le n-k-1, \\ \sum_{i=1}^{l}\binom{n-k}{i}(q-1)^{i-1} & \text{if } l = d-1, d = n-k+1. \end{cases} \quad (2.2)$$

In the next part, we will consider the number of useful rows in an l-separating matrix built in the second construction.

2.5 New Upper Bound in the Second Construction

Here, we just consider the case $l \le d-2$ or $l = d-1 \le n-k-1$. In case $l = d-1$ and $d = n-k+1$, the result can be derived similarly. In the second construction, H' is a full rank parity check matrix, A is a matrix over $GF(q)$ whose rows are all the nonzero normalized vectors of length $n-k$ and weight at most $l+1$. $H_{II} = AH'$. The number of rows in H_{II} is equal to the number of rows in A and is $\sum_{i=1}^{l+1}\binom{n-k}{i}(q-1)^{i-1}$.

For each $\mathcal{S}_i$ of size l, let $\left\{r_{v_1}, r_{v_2}, \ldots, r_{v_l}\right\}$ be the set of rows in H' in which the set of $\left\{(r_{v_1})_{\mathcal{S}_i}, (r_{v_2})_{\mathcal{S}_i}, \ldots, (r_{v_l})_{\mathcal{S}_i}\right\}$ forms a basic for the l – dimensional vector space over $GF(q)$.

For each $r_v \ne r_{v_j}$ $(j = 1, \ldots, n)$, we can find a row in A, $(a_1, a_2, \ldots, a_l)$, which satisfies

$$(r_v)_{\mathcal{S}_i} + a_1(r_{v_1})_{\mathcal{S}_i} + a_2(r_{v_2})_{\mathcal{S}_i} + \ldots + a_l(r_{v_l})_{\mathcal{S}_i} = \underline{\underline{0}} \quad (vector).$$

$a_i \in GF(q)$. The set $\left\{a_1, a_2, \ldots, a_l\right\}$ is unique.

Now we consider the matrix A

- Rows in A are the nonzero normalized vectors of length $n-k$ and weight from 1 to $l+1$.
- Choose any i, $1 \le i \le l+1$, from $n-k$, we can form $(q-1)^{i-1}$ rows.

However, not all $(q-1)^{i-1}$ rows play an useful role in H_{II}. We have $\binom{n}{l}$ subsets $\mathcal{S}_i$ (we can imagine that we have $\binom{n}{l}$ baskets). Now, we select rows in the set of $(q-1)^{i-1}$ rows to put into $\binom{n}{l}$ baskets.

- Row is put into one specific basket if and only if it has all zeros in the positions indexed by that basket.
- One row can be put into more than one basket.
- In the set of $(q-1)^{i-1}$ rows, because the set $\{a_1, a_2, ..., a_l\}$ is unique, there is maximum one row put into one basket.

Hence, in case $(q-1)^{i-1} \geq \binom{n}{l}$, the maximum number of useful rows in H_{II} is

$$\sum_{i=1}^{l+1} \binom{n-k}{i} \min\left\{(q-1)^{i-1}, \binom{n}{l}\right\}.$$

Let s_l be the lth separating redundancy in the second construction,

$$s_l \leq \begin{cases} \sum_{i=1}^{l+1} \binom{n-k}{i} \min\left\{(q-1)^{i-1}, \binom{n}{l}\right\} & \text{if } l \leq d-2 \text{ or } l = d-1 \leq n-k-1, \\ \sum_{i=1}^{l} \binom{n-k}{i} \min\left\{(q-1)^{i-1}, \binom{n}{l}\right\} & \text{if } l = d-1, d = n-k+1. \end{cases} \tag{2.3}$$

2.6 Chapter Summary

In this chapter, the reviews of the concept of separating matrices for decoding over channels causing both errors and erasures together with upper bounds on the minimum number of rows in such matrices were given. A new upper bound on the separating redundancy in the second construction was also introduced. In the next two chapters, we will propose two methods which can construct l-separating matrices with smaller total numbers of rows than in (2.1).

Chapter 3

New Upper Bounds based on Covering Design

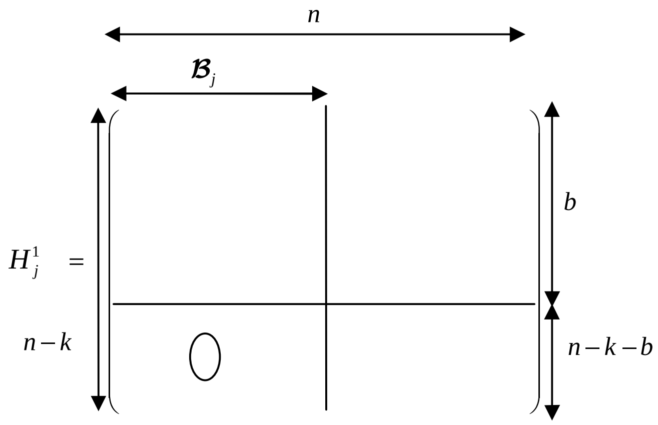
3.1 The First Method

In this chapter, we introduce a method (method 1) based on the first construction which can construct an l-separating matrix with a smaller total number of rows. Let H' be a full rank parity-check matrix of C which has the Hamming distance of d. It means that every $d-1$ or less columns of H' are linearly independent.

Method 1

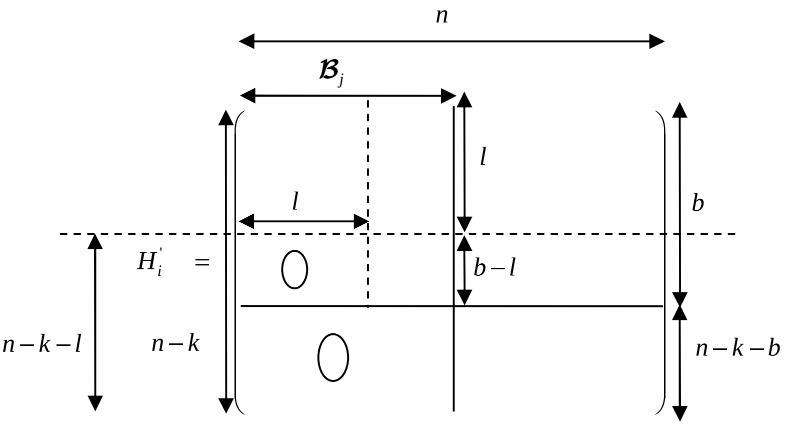
Step 1

Let $\mathcal{B}$ be a set of b-element subsets, $\mathcal{B}_j$, of $\mathcal{N}=\{1,2,...,n\}$, $1 \le l \le b \le d-1$, such that every l-element subset $\mathcal{S}_i$ is contained in at least one member of $\mathcal{B}$. Assign to each $\mathcal{S}_i$, $i=1,2,...,\binom{n}{l}$, an element $\mathcal{B}_j$ of $\mathcal{B}$ such that $\mathcal{S}_i$ is contained in this $\mathcal{B}_j$. $H'_{\mathcal{B}_j}$ has rank b. For any $\mathcal{B}_j$, by elementary row operations on H', we can obtain an $(n-k)\times n$ matrix of rank $n-k$ such that its last $n-k-b$ rows have zeros in the positions indexed by $\mathcal{B}_j$. After arranging columns, we obtain a matrix having this format. We call it H_j^1,

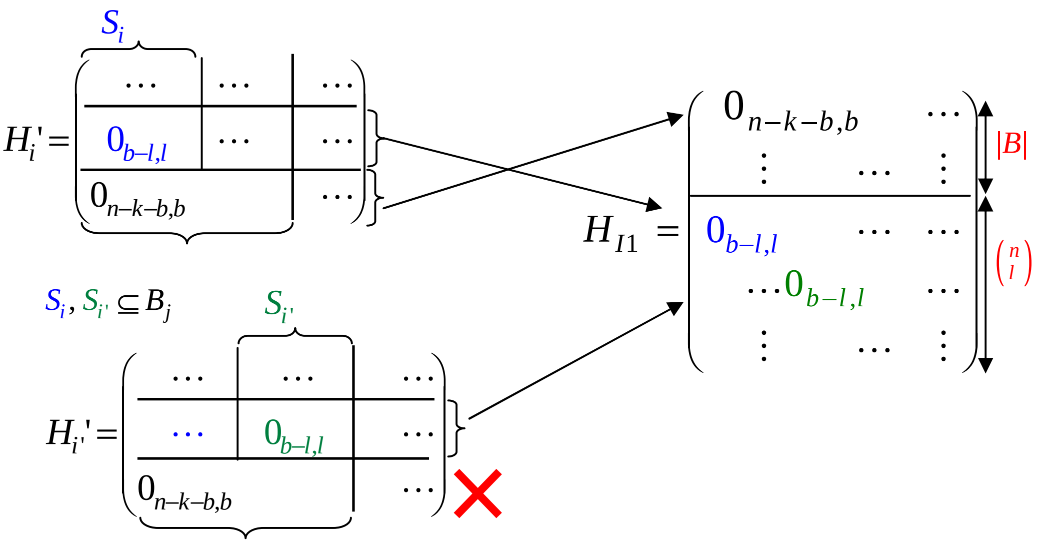


Step 2

For any $\mathcal{S}_i$ assigned to a certain $\mathcal{B}_j$, again, by elementary row operations, the matrix H_j^1 can be further changed into an $(n-k)\times n$ matrix $H_i^{'}$, still of rank $n-k$, which rows $l+1$, $l+2$,..., b have zeros in the positions indexed by $\mathcal{S}_i$, and which rows $b+1$, $b+2,\ldots,$ $n-k$ have zeros in the position indexed by $\mathcal{B}_j$. After column arrangement, we obtain a matrix having this format,



By this method, if $\mathcal{S}_i$ and $\mathcal{S}_{i'}$ belong to the same $\mathcal{B}_j$, the set of the last $n-k-b$ rows in $H_i^{'}$ and $H_{i'}^{'}$ will be the same. From the proof of Theorem 2 in [1], it follows that the matrix whose set of rows is the union of the sets of the last $n-k-b$ rows in H_j^{1} for $j=1,2,\ldots,|\mathcal{B}|$ and the rows $l+1$, $l+2,\ldots,b$ of $H_i^{'}$ for $i=1,2,\ldots,\binom{n}{l}$ is an l-separating parity-check matrix of the code C. Let $B(n,b,l)$ denote the minimum size of $\mathcal{B}$, $B(n,b,l)=\min|\mathcal{B}|$. This matrix has at most $(n-k-b)B(n,b,l)+\binom{n}{l}(b-l)$ rows.



3.2 Covering Designs

For $1 \le t \le u \le v$, *a* (v,u,t) *covering design is a collection of* u*-element subsets of* $\mathcal{V} = \{1, 2, ..., v\}$, *called blocks, such that every* t*-element subset of* $\mathcal{V}$ *is contained in at least one block.*

We need to find $B(v,u,t)$ which denotes the minimum size of a (v,u,t) covering design.

Example 1: $v = 8, u = 3, t = 2$,

All 2-element subsets (28 subsets):

```
12  23  34  45  56  67  78.
13  24  35  46  57  68
14  25  36  47  58
15  26  37  48
16  27  38
17  28
18
```

All 3-element subsets (56 subsets):

from these 21 subsets of size 3, we can form all $\binom{8}{2}$ subsets of size 2

```
123  134  145  156  167  178
124  135  146  157  168
125  136  147  158
126  137  148
127  138
128
```

```
234  245  256  267  278
235  246  257  268
236  247  258
237  248
238
```

```
345  356  367  378          456  467  478
346  357  368               457  468
347  358                    458
348

567  578          678
568
```

From the elements in the subset $\{1, 2, 3\}$, we can form $\{1, 2\}, \{2, 3\}, \{3, 1\}$. If we try all 21 subsets of size 3 in the box, we can form all 28 subsets of size 2. ■

The *covering design* problem has been investigated since many years ago. However, until now, there is no general formula of $B(v,u,t)$ for all triples (v,u,t). The optimal solutions, which satisfy the Schonheim lower bound ([3]), were achieved for some special cases or some specific triples. In the website www.ccrwest.org ([7]), we can find optimal solutions for the ranges $v \le 100, u \le 25, t \le 8$. Outside these ranges, optimal solutions have not been found yet.

It is clear that in case $u = t$, $B(v,u,t) = \binom{v}{u}$ and in case $t = 1$, $B(v,u,t) = \lceil v/u \rceil$. Here, we propose a *covering design* valid for all triples (v,u,t), $1 < t < u < v$. This solution is not optimal but it can give a general upper bound for the *covering number*, $B(v,u,t)$. We need at most $\binom{v-(u-t)}{t} - \sum_{\substack{k=1 \\ (\lfloor t/2 \rfloor > 1)}}^{\lfloor t/2 \rfloor} \binom{v-(u-t)-2k}{t-(2k-1)}$ u-element subsets of $\mathcal{V} = \{1, 2, ..., v\}$, called blocks, such that every t-element subset of $\mathcal{V}$ is contained in at least one block.

Before achieving this result, we begin with the first approach in which we show that with at most $\binom{v-(u-t)}{t}$ subsets of size u, we can form all $\binom{v}{t}$ subsets of size t.

3.2.1 Approach 1

Step1:

1. From the set $\mathcal{V} = \{1, 2, ..., v\}$, we take the first $u - t$ elements out of $\mathcal{V} = \{1, 2, ..., v\}$.
2. The rest of the set is $\{u-t+1, u-t+2, u-t+3, ..., v-1, v\}$. From these elements, we form all subsets of size t. The number of subsets is $\binom{v-(u-t)}{t}$.

Step2:

Now, we put the first $u - t$ elements into each subset of size t to form subset of size u. With these $\binom{v-(u-t)}{t}$ subsets of size u, it is easy to see that we can form all $\binom{v}{t}$ subsets of size t.

Example 2: $v = 6, u = 4, t = 3$, apply *approach 1*:

- Take $\{1\}$ out of $\{1, 2, 3, 4, 5, 6\}$. The rest of the set is $\{2, 3, 4, 5, 6\}$.
- Form subsets of size 3 from $\{2, 3, 4, 5, 6\}$. We obtain 10 subsets,

234	245	256	345	356	456.
235	246		346		
236					

- Put $\{1\}$ into these subsets.

$$\begin{array}{llllll}
1234 & 1245 & 1256 & 1345 & 1356 & 1456. \\
1235 & 1246 & & 1346 & & \\
1236 & & & & &
\end{array}$$

From these 10 subsets of size 4, we can form all 20 subsets of size 3 as if we form subsets of size 3 from $\{1, 2, 3, 4, 5, 6\}$,

$$\begin{array}{lllllll}
123 & 134 & 145 & 156 & 234 & 245 & 256 \\
124 & 135 & 146 & & 235 & 246 & \\
125 & 136 & & & 236 & & \\
126 & & & & & &
\end{array}$$

$$\begin{array}{lll}
345 & 356 & 456. \\
346 & &
\end{array}$$

■

If we take more than $u-t$ elements out of the set $\{1, 2, ..., v\}$, such as taking the first $u-t+1$ elements out and forming $\binom{v-(u-t)}{t-1}$ subsets of size $t-1$, we cannot form all $\binom{v}{t}$ subsets of size t.

Example 3: $v=6, u=4, t=3$,

- Take $\{1, 2\}$ out of $\{1, 2, 3, 4, 5, 6\}$. The rest of the set is $\{3, 4, 5, 6\}$.
- Form subsets of size 2 from $\{3, 4, 5, 6\}$. We obtain 6 subsets,

$$\begin{array}{lll}
34 & 45 & 56. \\
35 & 46 & \\
36 & &
\end{array}$$

- Put $\{1, 2\}$ into these subsets,

$$\begin{array}{lll}
1234 & 1245 & 1256. \\
1235 & 1246 & \\
1236 & &
\end{array}$$

With these 6 subsets, we cannot form some subsets such as,

$$\begin{array}{lll}
345 & 356 & 456. \\
346 & &
\end{array}$$

■

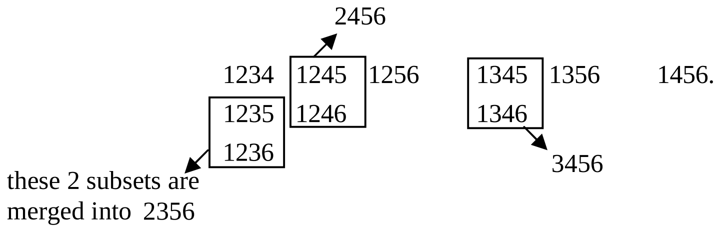
If we take less than $u-t$ elements out of the set $\boldsymbol{v}=\{1,2,...,v\}$ and apply the same method, the number of subsets of size u formed will be greater.

Example 4:

In example 2, $v=6, u=4, t=3$, we need 10 subsets of size 4 to form all 20 subsets of size 3. 10 subsets of size 4 are

1234	1245	1256	1345	1356	1456.
1235	1246		1346		
1236					

However, we can merge some subsets in order to reduce the number of needful subsets to fewer than $\binom{v-(u-t)}{t}$,



With these 7 subsets of size 4, $\{1,2,3,4\},\{2,3,5,6\},\{2,4,5,6\},\{1,2,5,6\},\{3,4,5,6\},\{1,3,5,6\},\{1,4,5,6\}$, we still can form all 20 subsets of size 3.■

3.2.2 Approach 2

In the first approach, we show that we need at most $\binom{v-(u-t)}{t}$ subsets of size u in order to form all $\binom{n}{t}$ subsets of size t but we also show that we can merge some subsets in order to reduce $|\mathcal{B}|$.

In order to get a better result than in *approach 1*, we come back **Step1** of *approach 1* and modify it.

Step1:

1. From the set $\mathcal{V}=\{1,2,...,v\}$, we take the first $u-t$ elements out of $\mathcal{V}=\{1,2,...,v\}$.
2. The rest of the set is $\{u-t+1,u-t+2,u-t+3,...,v-1,v\}$. From these elements, we form all subsets of size t and arrange them into columns based on these rules:
   - Elements in each subset are arranged in ascending order.
   - Subsets are arranged into columns. Subsets are in one column if and only if they have the same first $t-1$ elements (except the *special column*). It means that subsets in one column are different from each other only in the last element. The subset with the smaller last element will be put above.
   - *Special column*: If $t \geq 2$, we have the *special column*. The *special column* consists of subsets containing both elements $'v-1','v'$. It is easy to see that there are $\binom{v-(u-t)-2}{t-2}$ subsets in this column.

Example 5: $v=6, u=4, t=3$, apply *approach 2,* **Step1**:

- Take $\{1\}$ out of $\{1,2,3,4,5,6\}$. The rest of the set is $\{2,3,4,5,6\}$.
- Form subsets of size 3 from $\{2,3,4,5,6\}$ and arrange them into columns,

special column

234	245	345	256
235	246	346	356
236			456.

the *longest column*

We call the column which contains the most subsets, $\{2,3,4\},\{2,3,5\},\{2,3,6\}$, the *longest column* (except the *special column*). ■

In all cases, the *longest column* always begins with the subset $\{u-t+1, u-t+2, ..., u-2, u-1, u\}$. There are $v-u+1$ subsets in this column. The first $t-1$ elements in these subsets are the same.

Example 6: $v=9, u=5, t=4$,

Put '1' out of $\{1,2,3,4,5,6,7,8,9\}$. Form all subsets of size 4 and arrange them into columns, except the *special column*, the *longest column* will be

2345
2346
2347
2348
2349.

Basing on the first $t-1$ elements in the first subset in the *longest column*, we can form all $t-1$ *second longest columns*. We call this set the *original set* for these columns. The *original set* is the set from which we form new columns.

The first $t-1$ elements in the first subset in the *longest column* are

$$\{u-t+1, u-t+2, ..., u-3, \underset{\text{the \textit{second place} (count from the last)}}{u-2}, \overset{\text{the \textit{first place} (count from the last)}}{u-1}\}.$$

Convention: *the first place is the place of the last element among the first* $t-1$ *elements. The second place is the place of the next element to the left and so on…*

The set of the first $t-1$ elements in all subsets in the first *second longest column* is formed by choosing the element in the *first place* and increasing it by 1, $'u-1' \to 'u'$. We keep the first $t-2$ element as in the *original set*. The first subset in the first *second longest column* is

$$\{u-t+1, u-t+2, ..., u-3, u-2, \underset{\text{the \textit{chosen element}}}{u}, \overset{\text{the succeeding element is always greater than the preceding element}}{u+1}\}.$$

The set of the first $t-1$ elements in all subsets in the second *second longest column* is formed by choosing the element in the *second place* and increase it by 1, $'u-2' \to 'u-1'$. It means that the first $t-3$ elements in all subsets in this column are kept the same as in the *original set* and the element in the *first place* must be automatically increased to be greater than the preceding element. The first subset in the second *second longest column* is

$$\{\ u-t+1,\ u-t+2, ..., u-3,\ \underset{\substack{\nearrow \\ \text{the } \textit{chosen element}}}{u-1},\ \overset{\substack{\text{the succeeding element is always} \\ \text{greater than the preceding element} \\ \swarrow}}{u}, u+1\}.$$

The set of the first $t-1$ elements in subsets in the next column is formed by choosing the next element in the next place and so on... The set of the first $t-1$ elements in subsets in the last column is formed by choosing the element in the $(t-1)^{\text{th}}$ *place*, which is also the first element in the *original set* and increasing it by 1, $'u-t+1' \to 'u-t+2'$. All the succeeding elements will be automatically increased.

The first subset in the $(t-1)^{\text{th}}$ *second longest column* is

$$\{\underset{\substack{\nearrow \\ \text{the } \textit{chosen element}}}{u-t+2},\ \overset{\substack{\text{the succeeding element is always} \\ \text{greater than the preceding element} \\ \swarrow}}{u-t+3}, ..., u-2,\ u-1,\ u,\ u+1\}.$$

We can see that all the *second longest columns* and subsets in them have the same two features:

- The last element in the first subset in each column is $'u+1'$. It means that the number of subsets in each column is one fewer than the number of subsets in the *longest column.*
- The first $t-1$ elements in all subsets are contained in the *original set.*

Example 6 (cont): continue the above example, $v=9, u=5, t=4$.

The first subset in the *longest column* is $\{2,3,4,5\}$. Basing on the first three elements in this *original set*, we can form three *second longest columns*.

The set of the first three elements in all subsets in the first *second longest column* is formed by choosing the element in the *first plac*e, ‘4’, and increasing it by 1. The first subset in this column is $\{2,3,5,6\}$. The entire column is

2356
2357
2358
2359.

The set of the first three elements in subsets in the second *second longest column* is formed by choosing the element in the *second place*, ‘3’, and increasing it by 1. The first subset in this column is $\{2,4,5,6\}$. The entire column is

2456
2457
2458
2459.

By similar way, the first subset in the third *second longest column* is $\{3,4,5,6\}$. The entire column is

3456
3457
3458
3459.

We can see that the number of subsets in each *second longest column* is four which is one fewer than the number of subsets in the *longest column* and the first three elements in each subset are contained in the *original set*.■

Now, we will form all the *third longest columns* by using the first subset in each of the *second longest columns*.

The first subset in the first *second longest column* is

$$\{u-t+1, u-t+2,..., u-3, u-2, \underset{\nearrow}{u}, u+1\}.$$

the *chosen element* in the previous step

We will use this subset to form new columns. This subset is their *original set*. The *chosen element* in the previous step is in the *first place*. The number of new columns depends on the place of this *chosen element*. From now, the rule is that *we can choose the elements from the first place to the place of the chosen element in the previous step and increase each of them to form the set of the first* $t-1$ *elements for each new column*. For example, if the *chosen element* in the previous step is in the *fourth place*, we can choose four elements from the *first place* to the *fourth place* and increase each of them to form four new columns.

Therefore, with this *original set*, we can form only one *third longest column* by increasing the *chosen element* by 1, $'u' \to 'u+1'$. The first subset in this column is

$$\{u-t+1, u-t+2,..., u-3, u-2, \underset{\nearrow}{u+1}, u+2\}.$$

the *chosen element*

The first subset in the second *second longest column* is

$$\{u-t+1, u-t+2,..., u-3, \underset{\nearrow}{u-1}, u, u+1\}.$$

the *chosen element* in the previous step

The *chosen element* in the previous step is in the *second place*; hence we can use this subset to form other two *third longest columns*. Of course, this set is the *original set* for these two columns.

The set of the first $t-1$ elements in subsets in one new column is formed by choosing the element in the *first place* in the *original set* and increasing it by 1, $'u' \to 'u+1'$. The first subset in this column is

$$\{u-t+1, u-t+2, ..., u-3, u-1, \underset{\nearrow}{u+1}, u+2\}.$$

the *chosen element*

The set of the first $t-1$ elements in subsets in the other new column is formed by choosing the element in the *second place* in the *original set* and increasing it by 1, $'u-1' \to 'u'$. The first subset in this column is

$$\{u-t+1, u-t+2, ..., u-3, \underset{\nearrow}{u}, u+1, u+2\}.$$

the *chosen element*

Applying this procedure to the first subsets in all the *second longest columns*, we will form all the *third longest columns*. We can see that all the *third longest columns* and subsets in them have the same two features:

- The last element in the first subset in each column is $'u+2'$. It means that the number of subsets in each column is one fewer than the number of subsets in *second longest columns*.
- The first $t-1$ elements in all subsets in each column are contained in its *original set*.

Using the first subsets in all the *third longest columns* with the similar procedure, we will form all the *fourth longest columns* and so on…The *shortest columns* will have two subsets in each of them. In each step, all new columns and subsets in them have the same two features:

- The number of subsets in each new column is one fewer than the number of subsets in the column containing its *original set*.
- Except the *longest column* and the *special column*, the first $t-1$ elements in all subsets in each new column are contained in its *original set*.

Example 6 (cont): continue the above example, $v=9, u=5, t=4$.

The first subset in the first *second longest column* is $\{2,3,5,6\}$ with the *chosen element* in the *first place*. Hence, from this *original set*, we can form one new column.

Choose the element in the *first place* and increase it by 1, $'5' \to '6'$. The first subset in this new column is $\{2,3,6,7\}$. The entire column is

2367

2368

2369.

The first subset in the second *second longest column* is $\{2,4,5,6\}$ with the *chosen element* in the *second place*. From this *original set*, we can form two new columns.

The set of the first three elements in subsets in one new column is formed by choosing the element in the *first place* in the *original set* and increasing it by 1, $'5' \to '6'$. The first subset in this column is $\{2,4,6,7\}$. The entire column is

2467

2468

2469.

The first subset in other new column is $\{2,5,6,7\}$. The entire column is

2567

2568

2569.

Applying a similar procedure to the first subset in the third *second longest column* with the *chosen element* in the *third place*, $\{3,4,5,6\}$, we can form three new *third longest columns,*

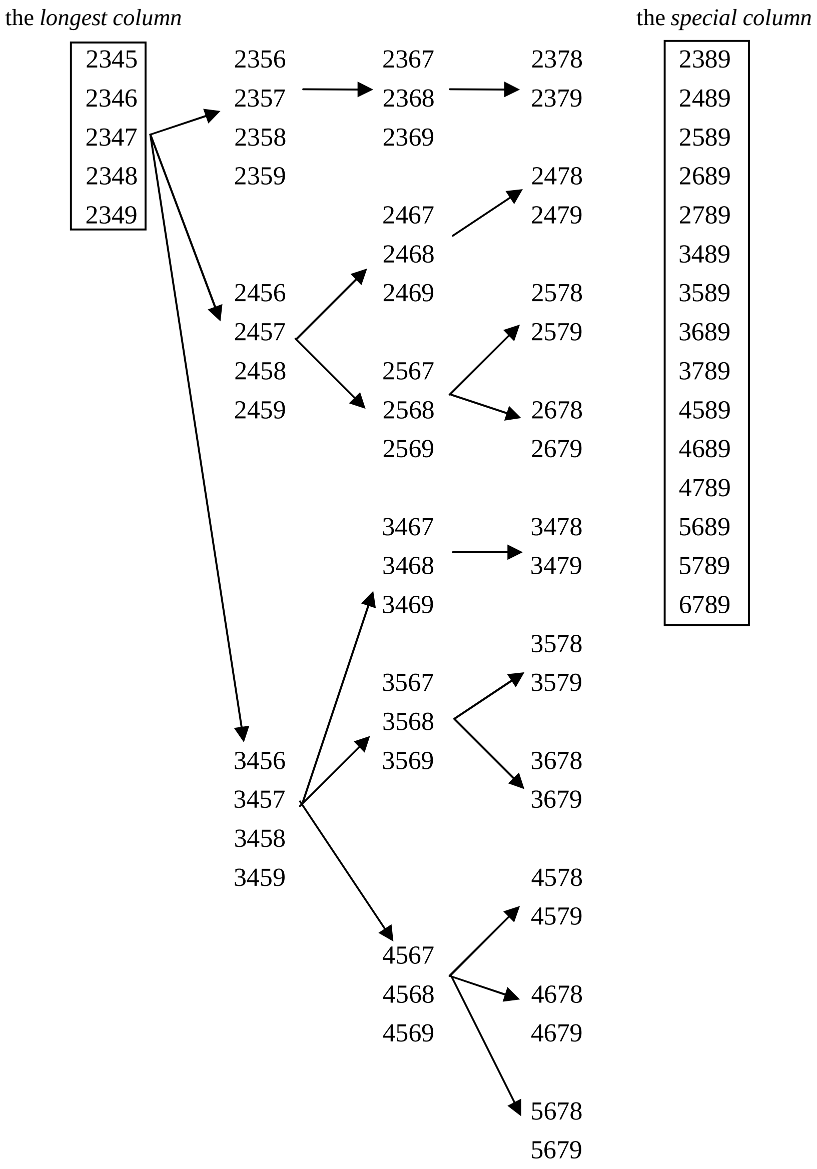
3467	3567	4567
3468	3568	4568
3469	3569	4569.

We can see that the number of subsets in each *third longest column* is three which is one fewer than the number of subsets in the *second longest columns* and the first three elements in each subset are contained in its *original set.*

Repeat the procedure until the number of subsets in all columns is two.

With $v=9, u=5, t=4$, the *special column* consists of subsets containing both ‘8’ and ‘9’. There are $\binom{v-(u-t)-2}{t-2}=\binom{9-(5-4)-2}{4-2}=\binom{6}{2}=15$ subsets in this column.

Finally, we can obtain this result:



■

Step2

1. We put the first $u-t$ elements, $\{1,2,...,u-t\}$, into each subset. Now, size of each subset is u.
2. If the number of subsets in the *longest column* is greater or equal to three and the *special column* exists, we can merge the last two subsets, which contain $'v-1'$ or $'v'$, in each column (except the *special column*) into one, the *merged set*, by this rule:
    - Eliminate the element $'1'$ in one subset.
    - Put the last element of the other subset into it.

We can merge the last two subsets in each column (except the *special column*) because:

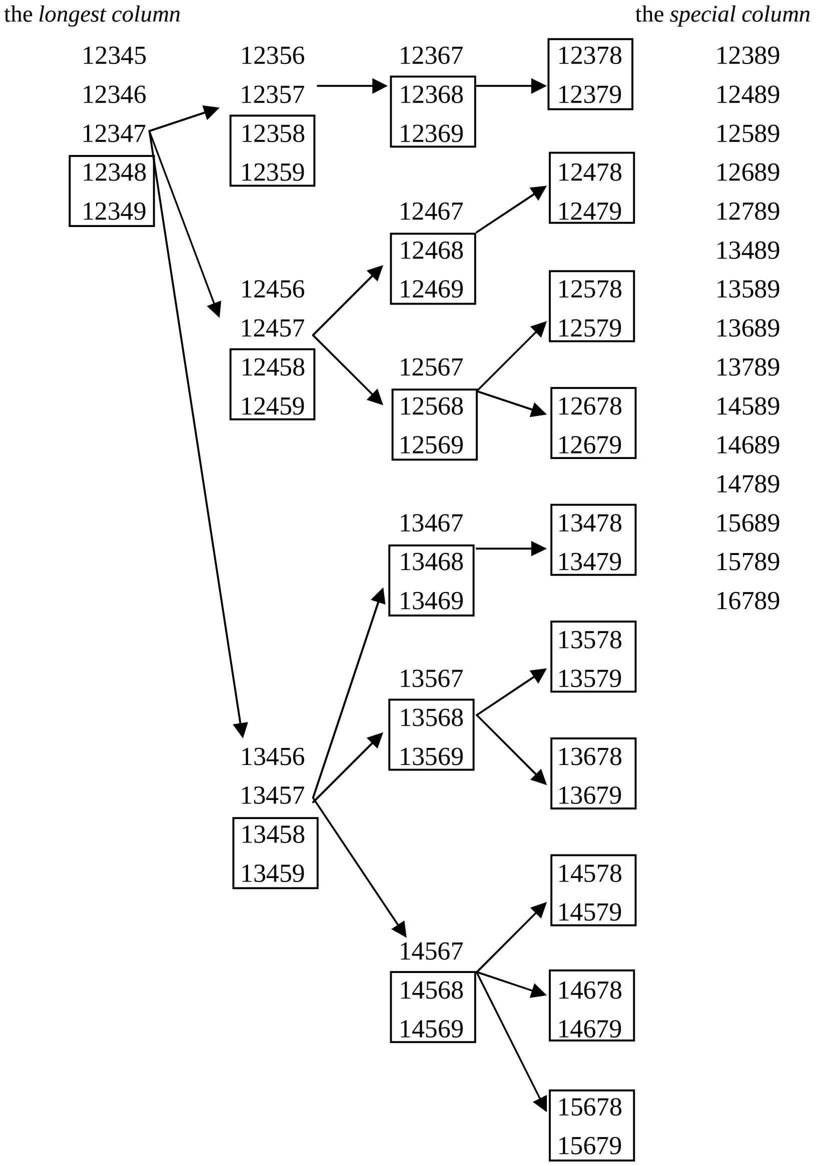
- The first $u-1$ elements in each subset are contained in the *original set*. Therefore, any subset of size t which is formed by using these $u-1$ elements can be formed by the *original set.*
- The last $u-1$ elements in each subset are contained in the *merged set*. Therefore, any subset of size t which is formed by using these $u-1$ elements can be formed by the *merged set.*
- Any subset of size t containing $\{1,v-1\}$ or $\{1,v\}$ can be formed by subsets in the *special column.*

Special column: there are $\binom{v-(u-t)-2}{t-2}$ subsets in this column. If $t=2$, the *special column* has only one subset, $\{1,2,...,u-2,v-1,v\}$. If $t\geq 3$, this column can be formed by this way:

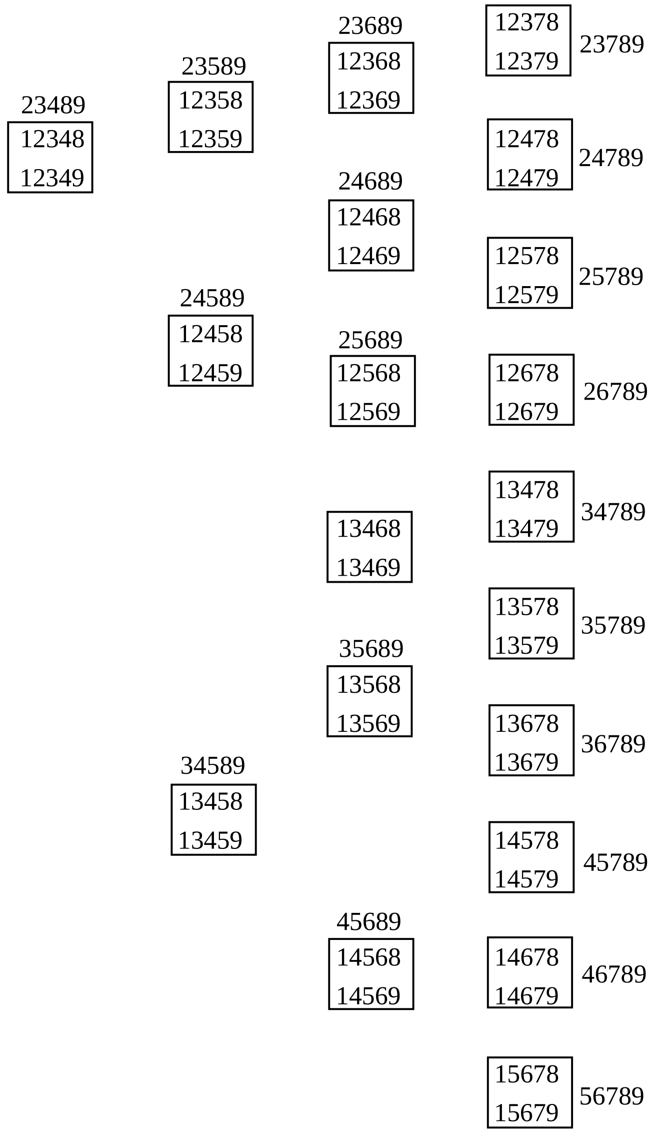
1. Put three elements $'1','v-1','v'$ out of the set $\mathcal{V}=\{1,2,3,...,v\}$.
2. The remaining set is $\{2,3,...,v-3,v-2\}$. From these elements, applying *approach 1*, we form subsets of size $u-3$ such as every $(t-2)$-element subset of $\{2,3,...,v-3,v-2\}$ is contained in at least one of them. We obtain $\binom{v-(u-t)-2}{t-2}$ subsets.
3. Next, we put three elements $'1','v-1','v'$ into each subset of size $u-3$ to form subset of size u. Now, we have all subsets in the *special column* and it is easy to see that with these $\binom{v-(u-t)-2}{t-2}$ subsets, we can form all subsets of size t containing $\{1,v-1\}$, $\{1,v\}$ or $\{1,v-1,v\}$.

Example 6 (cont): continue the above example: $v=9,u=5,t=4$.

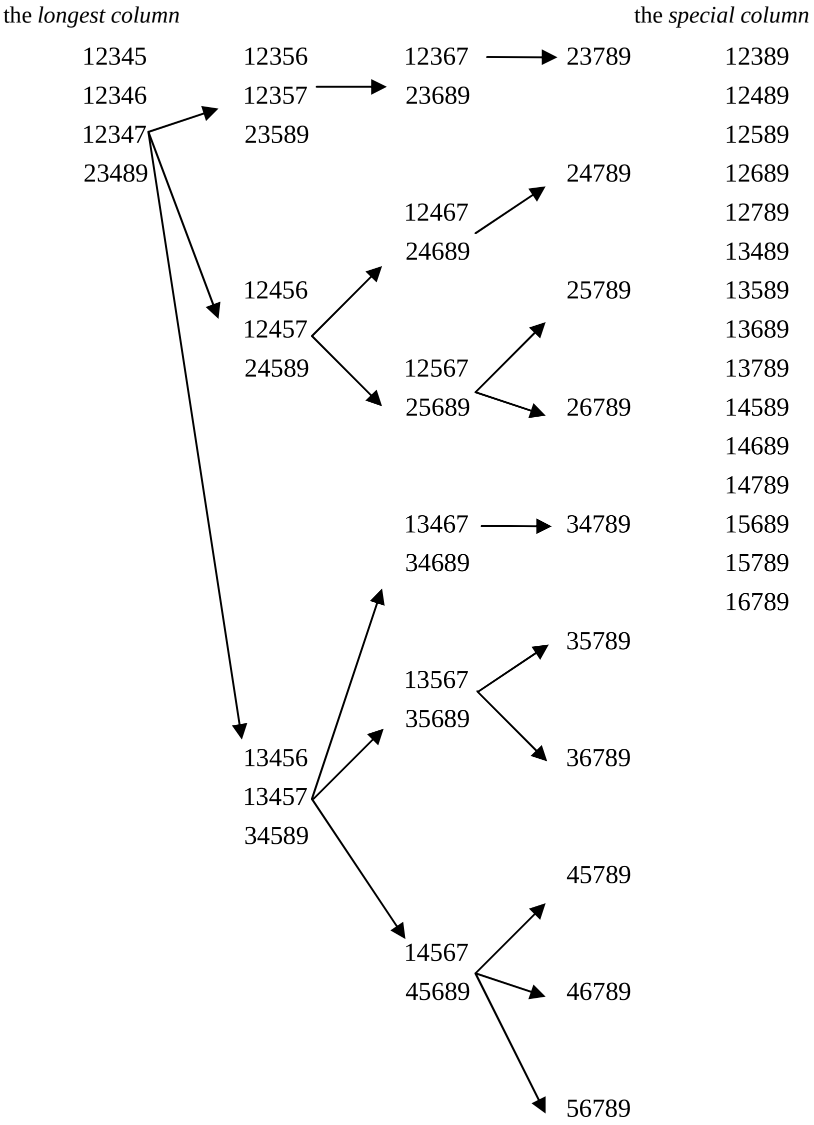
- Put ‘1’ into each subset.
- Merge the last two subsets in each column (except the *special column*).


the *longest column*
the *special column*
12345
12346
12347
12348
12349
12356
12357
12358
12359
12456
12457
12458
12459
13456
13457
13458
13459
12367
12368
12369
12467
12468
12469
12567
12568
12569
13467
13468
13469
13567
13568
13569
14567
14568
14569
12378
12379
12478
12479
12578
12579
12678
12679
13478
13479
13578
13579
13678
13679
14578
14579
14678
14679
15678
15679
12389
12489
12589
12689
12789
13489
13589
13689
13789
14589
14689
14789
15689
15789
16789

The *merged subsets*



The resulting subsets



With these subsets, we can form all subsets of size 4 as if we form all subsets of size 4 from $\{1, 2, \ldots, 9\}$ ■

The number of subsets which can be reduced is equal to the number of subsets containing the element $'v-1'$ or $'v'$. Therefore, the number of reduced subsets is $\binom{v-(u-t)-2}{t-1}$.

Until now, we need at most $\binom{v-(u-t)}{t}-\binom{v-(u-t)-2}{t-1}$ u-element subsets of $\mathcal{V}$ in order to form all subsets of size t from $\mathcal{V}=\{1,2,...,v\}$. As in the above example, after merging some subsets, we need at most $\binom{9-1}{4}-\binom{9-1-2}{4-1}=70-20=50$ subsets of size 5 in order to form all 126 subsets of size 4.

However, if we look at the *special column*, in the design of subsets of size $u-3$ in order to form all subsets of size $t-2$, by *approach 2,* we can merge some subsets in this design. If $t\geq 4$, the number of reduced subsets, which is $\binom{v-(u-t)-4}{t-3}$, is equal to the number of subsets containing the element $'v-3'$ or $'v-2'$. We continue with the *special column* in this design and so on…

Finally, we need at most $\binom{v-(u-t)}{t}-\sum_{\substack{k=1\\(\lfloor t/2\rfloor>1)}}^{\lfloor t/2\rfloor}\binom{v-(u-t)-2k}{t-(2k-1)}$ u-element subsets of $\mathcal{V}=\{1,2,...,v\}$ such that every t-element subset of $\mathcal{V}$ is contained in at least one member of them. The upper bound for k, $\lfloor t/2\rfloor$, can be determined by the condition of the existence of the *special column* in each design.

3.3 New Upper Bounds Following Method 1

In case $1<l<d-1$, for any $l<b\leq d-1$, we have $B(n,b,l)\leq\binom{n-(b-l)}{l}-\sum_{\substack{k=1\\(\lfloor l/2\rfloor>1)}}^{\lfloor l/2\rfloor}\binom{n-(b-l)-2k}{l-(2k-1)}$. (3.1)

Hence,

$$s_l\leq(n-k-b)B(n,b,l)+\binom{n}{l}(b-l) \tag{3.2}$$

$$\leq(n-k-b)\left(\binom{n-(b-l)}{l}-\sum_{\substack{k=1\\(\lfloor l/2\rfloor>1)}}^{\lfloor l/2\rfloor}\binom{n-(b-l)-2k}{l-(2k-1)}\right)+\binom{n}{l}(b-l). \tag{3.3}$$

For a given value of l, we can choose an appropriate value of b to get the best result. In general, in order to estimate a new upper bound for s_l, we can choose $b=d-1$ and the new upper bound is

$$s_l\leq(n-k-(d-1))\left(\binom{n-(d-1-l)}{l}-\sum_{\substack{k=1\\(\lfloor l/2\rfloor>1)}}^{\lfloor l/2\rfloor}\binom{n-(d-1-l)-2k}{l-(2k-1)}\right)+\binom{n}{l}(d-1-l). \tag{3.4}$$

In case $l=d-1$, we have the trivial result that $B(n,d-1,d-1)=\binom{n}{d-1}$ and we thus obtain the upper bounds from [1]. In case $l=1$,

$$s_1=(n-k-b)B(n,b,1)+n(b-1)=(n-k-b)\lceil n/b\rceil+n(b-1) \tag{3.5}$$

$$\leq(n-k-(d-1))\lceil n/(d-1)\rceil+n(d-2). \tag{3.6}$$

3.4 Chapter Summary

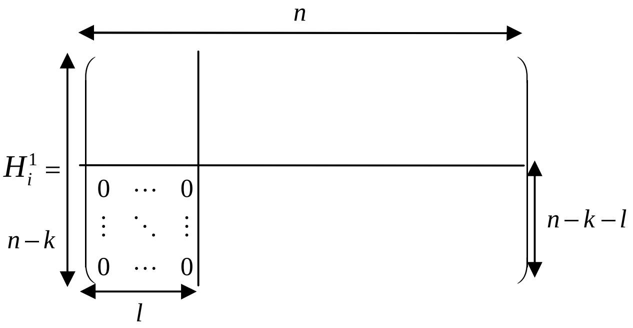
In this chapter, a method which can construct an l-separating matrix with a smaller total number of rows in comparison with (2.1) was introduced. A general upper bound on the *covering number* was also given. The efficiency of this method on some popular codes will be presented in chapter 4. In the next chapter, another method which can reduce the upper bound on the lth separating redundancy given in (2.1) in case $l \leq d-2$ will be introduced.

Chapter 4

Other Upper Bounds on the lth Separating Redundancy

4.1 The Second Method

In this chapter, we introduce another method (method 2) which can give a smaller upper bound in comparison with the value given in (2.1) in case $l \le d-2$. Again, H' is a full rank parity check matrix and $\mathcal{S}_i$ is a subset of $\{1,2,...,n\}$ of size l. $H'_{\mathcal{S}_i}$ has rank l and by elementary row operations on H', we can obtain an $(n-k)\times n$ matrix of rank $n-k$ such that its last $n-k-l$ rows have zeros in the positions indexed by $\mathcal{S}_i$. After arranging columns, we obtain a matrix having this format,



Taking the last $n-k-l$ rows in H_i^1 which have zeros in the positions indexed by $\mathcal{S}_i$ to form a new matrix, we call it H_i^2. In [1], H_l, an l-separating matrix in the first construction, is the matrix whose set of rows is the union of all H_i^2 for $i = 1,2,...,\binom{n}{l}$.

Method 2

By elementary row operations and column arrangement on H_i^2, we obtain a matrix, H_i^3, having the format,

$$H_i^3 = \begin{array}{c}
\begin{array}{ccc}
\overleftrightarrow{\quad l \quad} & \overleftrightarrow{\qquad n-k-l \qquad} & \overleftrightarrow{\quad k \quad}
\end{array} \\
\left(\begin{array}{ccc|cccccc|cc}
0 & \cdots & 0 & Y & 0 & 0 & \cdots & 0 & 0 & X & \cdots \\
0 & \cdots & 0 & 0 & Y & 0 & \cdots & 0 & 0 & X & \cdots \\
0 & \cdots & 0 & 0 & 0 & Y & \cdots & 0 & 0 & X & \cdots \\
\vdots & \cdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
0 & \cdots & 0 & 0 & 0 & 0 & \cdots & Y & 0 & X & \cdots \\
0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & Y & X & \cdots
\end{array}\right) \updownarrow n-k-l \\
\begin{array}{cc}
\underbrace{\qquad\qquad}_{\text{Group A}} & \underbrace{\qquad\qquad\qquad\qquad}_{\text{Group B}}
\end{array}
\end{array}$$

Y: a non-zero symbol.
X: an arbitrary symbol.
We call:

- Group A: set of l positions in which each column contains all zero symbols.
- Group B: set of $n-k-l$ positions in which each column contains only one non-zero symbol.

Now, we consider $\mathcal{S}_j$ which consists of $l-1$ positions in Group A and one position in Group B, for example: the last $l-1$ positions in Group A and the first position in Group B. According to the first construction, we need H_j^2 of rank $n-k-l$ which has zero columns in the positions indexed by $\mathcal{S}_j$.

$$H_j^2 = \begin{array}{c} \overbrace{\hphantom{X\ 0\ \cdots\ 0}}^{\mathcal{S}_i} \\ \left(\begin{array}{c|ccc|c|cc} X & 0 & \cdots & 0 & 0 & X & \cdots \\ X & 0 & \cdots & 0 & 0 & X & \cdots \\ X & 0 & \cdots & 0 & 0 & X & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X & 0 & \cdots & 0 & 0 & X & \cdots \\ X & 0 & \cdots & 0 & 0 & X & \cdots \end{array}\right) \\ \underbrace{\hphantom{0\ \cdots\ 0\ 0}}_{\mathcal{S}_j} \end{array} \left.\vphantom{\begin{array}{c}X\\X\\X\\ \vdots\\X\\X\end{array}}\right\} n-k-l$$

However, if we look at H_i^3,

$$H_i^3 = \left(\begin{array}{c|ccc|c|ccccc|cc} 0 & 0 & \cdots & 0 & Y & 0 & 0 & \cdots & 0 & 0 & X & \cdots \\ 0 & 0 & \cdots & 0 & 0 & Y & 0 & \cdots & 0 & 0 & X & \cdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & Y & \cdots & 0 & 0 & X & \cdots \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & Y & 0 & X & \cdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & Y & X & \cdots \end{array}\right) \updownarrow n-k-l$$

$$\underbrace{\hphantom{0\ 0\ \cdots\ 0}}_{\text{Group A}} \quad \underbrace{\hphantom{Y\ 0\ 0\ \cdots\ 0\ 0}}_{\text{Group B}}$$

$n-k-l-1$ rows have zeros in the positions indexed by $\mathcal{S}_j$

H_i^3 already has $n-k-l-1$ rows which have zeros in the positions indexed by $\mathcal{S}_j$. Therefore, instead of using all $n-k-l$ rows of H_j^2, we need only one row accompanied with the set of $n-k-l-1$ rows from H_i^3. In other word, the last $n-k-l-1$ rows in H_i^3 are also useful in case of $\mathcal{S}_j$. Let d' be the Hamming distance of the punctured code formed by deleting the symbols in positions belonging to the subset $\mathcal{S}_j$. Since $d' \geq d-l$, in case $l \leq d-2$, $d' \geq 2$. Therefore, in H_j^2 , except l all-zero columns, each column must contain at least one non-zero symbol. The only row we need from H_j^2 should contain a non-zero symbol in the position which is in the subset $\mathcal{S}_i$ but not in the subset $\mathcal{S}_j$. The set of $n-k-l-1$ rows from H_i^3, which are linear independent, has zeros in this position. Any row which has a non-zero symbol in this position will be linear independent with the rows in this set.

In general, H_i^3 has $n-k-l-1$ rows having zeros in the positions indexed by any $\mathcal{S}_j$ consisting of $l-1$ positions in Group A and one position in Group B. From l positions in the Group A and $n-k-l$ positions in the Group B, we can form $\binom{l}{l-1}\binom{n-k-l}{1}=l(n-k-l)$ subsets $\mathcal{S}_j$. For each of these $l(n-k-l)$ subsets $\mathcal{S}_j$, we need only one row.

In H_1, for $\mathcal{S}_i$, we replace the set of $n-k-l$ rows of H_i^2 with H_i^3 and for each of $l(n-k-l)$ subsets $\mathcal{S}_j$, we can remove $n-k-l-1$ rows in each H_j^2. Since this new matrix belongs to the class of H_1 matrix, it is also an l-separating parity-check matrix of the code C.

Now, we consider $\mathcal{S}_k$ consisting of $l-2$ positions in Group A and two positions in Group B, for example, the last $l-2$ positions in Group A and the first two positions in Group B. In H_i^3, we have $n-k-l-2$ rows which have zeros in the positions indexed by $\mathcal{S}_k$.

$$H_i^3=\begin{array}{c}\left(\begin{array}{cccc|cccccc|cc}
0 & 0 & \cdots & 0 & Y & 0 & 0 & \cdots & 0 & 0 & X & \cdots\\
0 & 0 & \cdots & 0 & 0 & Y & 0 & \cdots & 0 & 0 & X & \cdots\\
0 & 0 & \cdots & 0 & 0 & 0 & Y & \cdots & 0 & 0 & X & \cdots\\
\vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots\\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & Y & 0 & X & \cdots\\
0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & Y & X & \cdots
\end{array}\right)\updownarrow n-k-l\\
\underbrace{\qquad\qquad}_{\text{Group A}}\ \underbrace{\qquad\qquad\qquad\qquad}_{\text{Group B}}\qquad\qquad\end{array}$$

$n-k-l-2$ rows have zeros in the positions indexed by $\mathcal{S}_k$

In this case, we can remove $n-k-l-2$ rows in H_k^2 if there are two rows in H_k^2 which are linearly independent with the set of $n-k-l-2$ rows from H_i^3. However, H_k^2 can have this format

$$H_k^2=\begin{array}{c}\qquad\qquad\qquad\mathcal{S}_k\\
\left(\begin{array}{cc|ccc|cc|cc}
0 & 0 & 0 & \cdots & 0 & 0 & 0 & X & \cdots\\
Y & Y & 0 & \cdots & 0 & 0 & 0 & X & \cdots\\
0 & 0 & 0 & \cdots & 0 & 0 & 0 & X & \cdots\\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\
0 & 0 & 0 & \cdots & 0 & 0 & 0 & X & \cdots\\
0 & 0 & 0 & \cdots & 0 & 0 & 0 & X & \cdots
\end{array}\right)\\
\underbrace{\qquad\qquad\qquad\quad}_{\mathcal{S}_i}\qquad\qquad\qquad\end{array}$$

In this case, we cannot find two rows in H_k^2 to ensure that these two rows are linear independent with $n-k-l-2$ rows from H_i^3. It means that we cannot gain advantage in this case. Therefore, we can conclude that this method is used only in case $\mathcal{S}_j$ differs $\mathcal{S}_i$ in only one position and this position is chosen from Group B.

4.2 Application

Now, we propose the ways applying this method to different values of l. We mention some important points of this method:

- Size of Group A is l and size of Group B is $n-k-l$.
- Positions appearing in Group B depend on positions in Group A and structure of the parity-check matrix.
- We gain advantage from subsets $\mathcal{S}_j$ consisting of one position in Group B and $l-1$ positions in Group A.
- Here, we define the couple (A| B) = {positions in A | positions in B}.

For example: $l=3$, $n-k-l=4$, assume that Group A = {1, 2, 3}, Group B = {4, 6, 7, 9}, then (A| B) = {1, 2, 3 | 4, 6, 7, 9}.

- With each couple (A| B), we can form $l(n-k-l)$ subsets $\mathcal{S}_j$ consisting of $l-1$ positions in Group A and one position in Group B. For each of these $l(n-k-l)$ subsets $\mathcal{S}_j$, we need only one row.

For example: with the couple (A| B) = {1, 2, 3 | 4, 6, 7, 9}, we can form 12 subsets $\mathcal{S}_j$: $\mathcal{S}_1=\{1,2,4\}$, $\mathcal{S}_2=\{1,2,6\}$, $\mathcal{S}_3=\{1,2,7\}$, $\mathcal{S}_4=\{1,2,9\}$, $\mathcal{S}_5=\{1,3,4\}$, $\mathcal{S}_6=\{1,3,6\}$, $\mathcal{S}_7=\{1,3,7\}$, $\mathcal{S}_8=\{1,3,9\}$, $\mathcal{S}_9=\{2,3,4\}$, $\mathcal{S}_{10}=\{2,3,6\}$, $\mathcal{S}_{11}=\{2,3,7\}$, $\mathcal{S}_{12}=\{2,3,9\}$.

- A new couple (A| B) is formed (valid) if and only if $l(n-k-l)$ subsets $\mathcal{S}_j$ formed by this couple are totally different from all other $\mathcal{S}_j$ formed by all previous couples.

For example: with Group A* = {1, 5, 6}, we suppose that Group B* is {2, 8, 10, 11}. In this case, the couple (A*| B*) cannot be formed because among 12 subsets formed by (A*| B*) = {1, 5, 6 | 2, 8, 10, 11}, the subset $\mathcal{S}_j=\{1,2,6\}$ has been already formed by (A| B). Hence, the couple (A*| B*) is not valid.

- More couples can be formed, more advantage we can gain.
- Let m be the number of couples valid. From these m couples, we can form $ml(n-k-l)$ subsets $\mathcal{S}_j$ for each of which, we need only one row. For each of the rest, $\binom{n}{l}-ml(n-k-l)$ subsets, we still need $n-k-l$ rows. Let H_{l2} be the matrix constructed by *method 2.* This matrix has at most $\left(\binom{n}{l}-ml(n-k-l)\right)(n-k-l)+ml(n-k-l)=\binom{n}{l}(n-k-l)-ml(n-k-l)(n-k-l-1)$ rows.

4.2.1 $l=1$

In this special case, we slightly change *method 2* to get a better result. From an $(n-k)\times n$ parity-check matrix H' of C, by elementary row operations, we can obtain a full rank $(n-k)\times n$ parity-check matrix H'' which contains an $(n-k)\times(n-k)$ submatrix D with zeros in all entries outside the main diagonal.

$$H''=\begin{array}{cc} \overset{n-k}{\longleftrightarrow}\quad\overset{k}{\longleftrightarrow} & \\ \left(\begin{array}{cccc|cc} Y & 0 & \cdots & 0 & X & \cdots \\ 0 & Y & \cdots & 0 & X & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & Y & X & \cdots \end{array}\right) & \Big\updownarrow\, n-k \end{array}$$

Hence, for all $n-k$ sets $\mathcal{S}_i=\{i\}$ corresponding to the column indices of D, the matrix H'' has $n-k-1$ zeros in column i. For each of the remaining k sets $\mathcal{S}_i=\{i\}$, by elementary row operations on H', we can obtain an $(n-k)\times n$ matrix, $H_i^{'}$, of rank $n-k$ such that its last $n-k-1$ rows have zeros in the positions indexed by $\mathcal{S}_i$. Let H_{l2} denote the matrix whose set of rows is the union of the last $n-k-1$ rows in these k matrix $H_i^{'}$ and the rows of the matrix H''. The number of row in H_{l2} is at most $k(n-k-1)+(n-k)$. Since the matrix H_{l2} belongs to the class of H_l matrices, H_{l2} is a 1-separating parity-check matrix of the code C. Then,

$$s_1\le(k+1)(n-k-1)+1 \text{ with } d\ge 2,\ n-k\ge 2.$$

4.2.2 $l=2$

The first couple: we can choose the first two positions to be Group A$_1$,

$$H_1^3=\left(\begin{array}{cc|ccccc|cc} 0 & 0 & Y & 0 & \cdots & 0 & 0 & X & \cdots \\ 0 & 0 & 0 & Y & \cdots & 0 & 0 & X & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & Y & 0 & X & \cdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & Y & X & \cdots \\ \multicolumn{2}{c}{\underbrace{\qquad}_{\text{Group A}_1}} & \multicolumn{5}{c}{\underbrace{\qquad\qquad\qquad}_{\text{Group B}_1}} & \multicolumn{2}{c}{\underbrace{\qquad}_{\text{Group C}_1}} \end{array}\right).$$

The second couple: we choose any two positions in Group C$_1$ (C = {1,2,…,n} \ (A$\cup$B)) to be Group A$_2$. Because each of these two positions does not appear in $2(n-k-2)$ subsets formed by the first couple, without noticing Group B$_2$, we can assure that $2(n-k-2)$ subsets formed by (A$_2$| B$_2$) are totally different from all previous subsets.

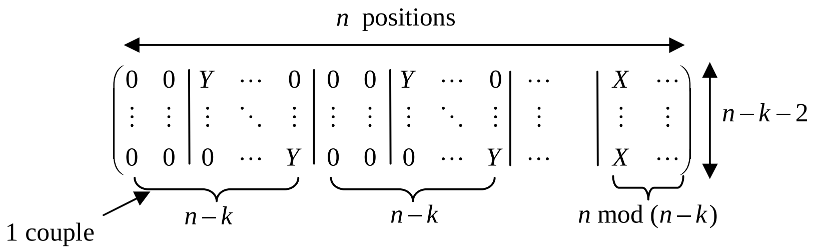
For example: $l=2, n=12, n-k-l=4$, Group A$_1$= {1, 2}, we suppose that Group B$_1$ = {4, 5, 7, 8}

The first couple $(A_1 \mid B_1)$ = {1, 2 | 4, 5, 7, 8}. From this couple, we form 8 subset $\mathcal{S}_j$ = {1, 4}, {1, 5}, {1, 7}, {1, 8}, {2, 4}, {2, 5}, {2, 7}, {2, 8}.

Group C_1 = {3, 6, 9, 10, 11, 12}. Now, we choose any two positions in Group C_1 to be Group A_2. Suppose that we choose Group A_2 = {3, 6}. With this Group A_2, 8 subsets $\mathcal{S}_j$ formed by $(A_2 \mid B_2)$ contain either {3} or {6}, which is not in 8 previous subsets $\mathcal{S}_j$. Hence, we can guarantee that $(A_2 \mid B_2)$ is valid.■

The third couple: we should choose any two positions which are in the intersection of Group C_1 and Group C_2. Group A of the next couple is always chosen from the positions in the intersection of all previous Groups C.

Now, we should compute the minimum number of couples can be formed. The worst case happens when positions in Group B of the next couple are also in the intersection of all previous Groups C. We can model roughly like this,



If $(n \bmod (n-k)) \le 1$, the minimum number of couples can be formed is $\left\lfloor \frac{n}{n-k} \right\rfloor$ ($\lfloor\ \rfloor$means taking the integer part, e.g. $\lfloor 2.7 \rfloor = 2$).

If $(n \bmod (n-k)) \ge 2$, the minimum number of couples can be formed is $\left\lfloor \frac{n}{n-k} \right\rfloor + 1$.

If $(n \bmod (n-k)) \le 1$, there are at least $\left\lfloor \frac{n}{n-k} \right\rfloor 2(n-k-2)$ subsets, for each of which, we need only one row. For each of the rest, $\binom{n}{2} - \left\lfloor \frac{n}{n-k} \right\rfloor 2(n-k-2)$ sets, we need $(n-k-2)$ rows.

The maximum number of rows in H_{l2} is

$$\left(\binom{n}{2} - \left\lfloor \frac{n}{n-k} \right\rfloor 2(n-k-2)\right)(n-k-2) + \left\lfloor \frac{n}{n-k} \right\rfloor 2(n-k-2)$$
$$= \binom{n}{2}(n-k-2) - \left\lfloor \frac{n}{n-k} \right\rfloor 2(n-k-2)(n-k-3).$$

If $(n \bmod (n-k)) \geq 2$, there are at least $\left(\left\lfloor \frac{n}{n-k} \right\rfloor + 1\right) 2(n-k-2)$ subsets, for each of which, we need only one row. For each of the rest, $\binom{n}{2} - \left(\left\lfloor \frac{n}{n-k} \right\rfloor + 1\right) 2(n-k-2)$ sets, we need $(n-k-2)$ rows.

The maximum number of rows in H_{l2} is

$$\begin{aligned}&\left(\binom{n}{2} - \left(\left\lfloor \frac{n}{n-k} \right\rfloor + 1\right) 2(n-k-2)\right)(n-k-2) + \left(\left\lfloor \frac{n}{n-k} \right\rfloor + 1\right) 2(n-k-2)\\ &= \binom{n}{2}(n-k-2) - \left(\left\lfloor \frac{n}{n-k} \right\rfloor + 1\right) 2(n-k-2)(n-k-3).\end{aligned}$$

4.2.3 $l \geq 3$

In this case, if Group A of each couple differs from each other in at least three positions, we can ensure that all $\mathcal{S}_j$ formed by each couple will be different from all other subsets. It means that we have to solve the problem: *choose groups of l positions from n positions, each group must differ from the others in at least 3 positions*.

For example: if $l = 5$, {1, 2, 3, 4, 5} and {4, 5, 6, 7, 8} differ from each other in three positions while {1, 2, 3, 4, 5} and {3, 4, 5, 6, 7} differ from each other only in two positions.

Here, we propose a way to attain $\left\lfloor \frac{n}{3} \right\rfloor$ groups.

1. Write n positions in a line.
2. Copy the first $l-3$ positions to the end of the line. Now, the line has $n+l-3$ positions.
3. The first group consists of the first l positions.
4. The next group is formed by choosing the last $l-3$ positions in the previous group and the next three positions.
5. Continue to the end of the line.

For example: $n = 10, l = 5$,

Step1:	1	2	3	4	5	6	7	8	9	10		
Step2:	1	2	3	4	5	6	7	8	9	10	1	2
Step3, 4, 5:	1	2	3	4	5	6	7	8	9	10	1	2
	x	x	x	x	x							
				x	x	x	x	x				
							x	x	x	x	x	

Three groups can be formed: (1,2,3,4,5), (4,5,6,7,8), (7,8,9,10,1).

We can choose these groups as Groups A to make $\left\lfloor \frac{n}{3} \right\rfloor$ couples.■

By similar explanation, the total number of rows in H_{l2} in case $l \geq 3$,

$$\begin{aligned} &\left(\binom{n}{l} - \left\lfloor \frac{n}{3} \right\rfloor l(n-k-l) \right)(n-k-l) + \left\lfloor \frac{n}{3} \right\rfloor l(n-k-l) \\ =\ &\binom{n}{l}(n-k-l) - \left\lfloor \frac{n}{3} \right\rfloor l(n-k-l)(n-k-l-1). \end{aligned}$$

4.3 New Upper Bounds Following Method 2

Let s_l be the lth separating redundancy.

- $l=1$: $s_1 \leq (k+1)(n-k-1)+1$. (4.1)
- $l=2$, $(n \bmod (n-k)) \leq 1$: $s_2 \leq \binom{n}{2}(n-k-2) - \left\lfloor \frac{n}{n-k} \right\rfloor 2(n-k-2)(n-k-3)$. (4.2)
- $l=2$, $(n \bmod (n-k)) \geq 2$: $s_2 \leq \binom{n}{2}(n-k-2) - \left(\left\lfloor \frac{n}{n-k} \right\rfloor + 1 \right) 2(n-k-2)(n-k-3)$. (4.3)
- $3 \leq l \leq d-2$: $s_{l\ (l \geq 3)} \leq \binom{n}{l}(n-k-l) - \left\lfloor \frac{n}{3} \right\rfloor l(n-k-l)(n-k-l-1)$. (4.4)

4.5 Chapter Summary

In this chapter, another method which can reduce the total number of rows in an l-separating matrix in the first construction in case $l \leq d-2$ was introduced. In the next chapter, the efficiencies of *method 1* and *method 2* on some popular codes will be examined. Besides, comparisons between upper bounds will be also presented.

Chapter 5

Comparisons

In this part, we just consider the case in which the number of erasures, l, is at least 1 and at most $d-1$. We denote h_1 as the number of rows of an l-separating matrix of the code C, H_I, built by the first construction from [1],

$$h_1 = \begin{cases} \binom{n}{l}(n-k-l) & \text{if } l \le d-2 \text{ or } l = d-1 \le n-k-1, \\ \binom{n}{l-1} & \text{if } l = d-1, d = n-k+1. \end{cases}$$

h_{11} denotes the number of rows of an l-separating matrix of the code C constructed by *method 1* mentioned in Chapter 3 with $b = d-1$ and $1 < l < d-1$,

$$h_{11} = \binom{n}{l}(d-1-l) + \left(\binom{n-(d-1-l)}{l} - \sum_{\substack{k=1 \\ (\lfloor l/2 \rfloor > 1)}}^{\lfloor l/2 \rfloor} \binom{n-(d-1-l)-2k}{l-(2k-1)} \right)(n-k-(d-1)).$$

In case $l = d-1$, $h_{11} = h_1$. In case $l = 1$, $h_{11} = (n-k-(d-1))\lceil n/(d-1) \rceil + n(d-1-l)$.

h_{12} denotes the number of rows of an l-separating matrix of the code C constructed by *method 2* mentioned in Chapter 4:

- If $l = 1$, $h_{12} = (k+1)(n-k-1)+1$.
- If $l = 2$ and $n \bmod (n-k) \le 1$, $h_{12} = \binom{n}{2}(n-k-2) - \left\lfloor \frac{n}{n-k} \right\rfloor 2(n-k-2)(n-k-3)$.
- If $l = 2$ and $n \bmod (n-k) \ge 2$, $h_{12} = \binom{n}{2}(n-k-2) - \left(\left\lfloor \frac{n}{n-k} \right\rfloor + 1 \right) 2(n-k-2)(n-k-3)$.
- If $3 \le l \le d-2$, $h_{12} = \binom{n}{l}(n-k-l) - \left\lfloor \frac{n}{3} \right\rfloor l(n-k-l)(n-k-l-1)$.

h_2 denotes the number of rows of an l-separating matrix of the code C, H_{II}, built by the second construction from [1], $h_2 = \sum_{i=1}^{l+1} \binom{n-k}{i} (q-1)^{i-1}$.

h_{21} denotes the number of useful rows in H_{II}, $h_{21} = \sum_{i=1}^{l+1} \binom{n-k}{i} \min \left\{ (q-1)^{i-1}, \binom{n}{l} \right\}$, discussed in Section 2.5.

The upper bound on the lth separating redundancy can be determined by $\min\{h_{11}, h_{12}, h_{21}\}$.

Here, we use Matlab to compute the upper bounds mentioned in the thesis for some special codes.

5.1 Hamming Codes

The class of Hamming codes is one of the oldest families of error-correcting codes. The codes are defined by:

$$n = 2^m - 1,\ (m \geq 2)$$
$$k = 2^m - 1 - m$$
$$d = 3.$$

Hamming codes have been widely used for error control in digital communication and data storage systems over the years owing to their high rate and decoding simplicity. The Hamming distance of the code is three; hence the code is single error-correcting or double error-detecting. Here, we just examine codes with short lengths.

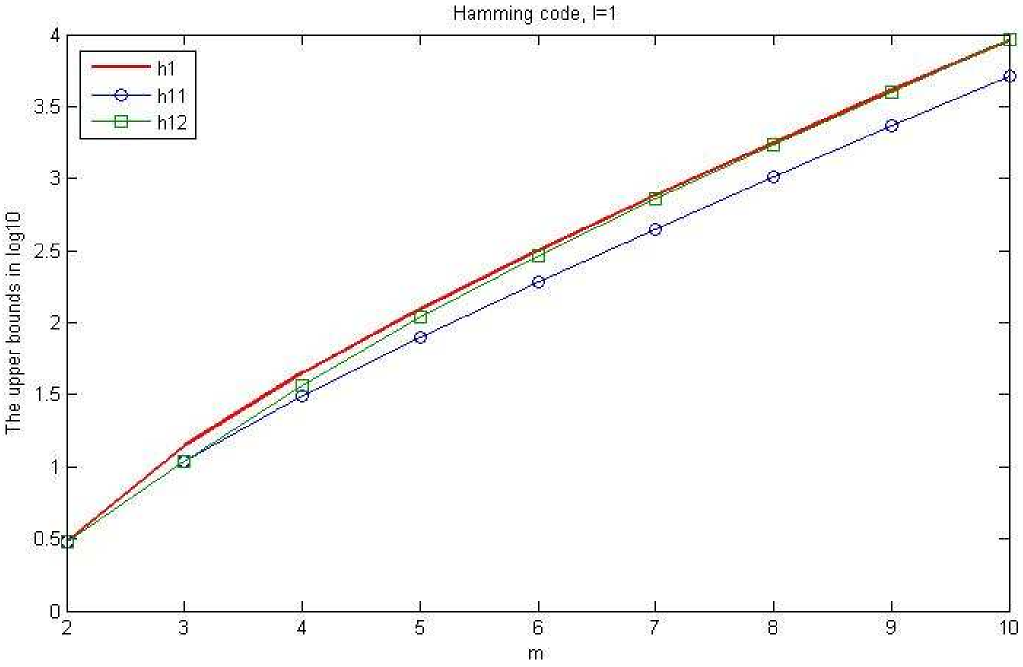
Because $d = 3$, l can be either 1 or 2.

5.1.1 $l = 1$

Table 5.1 h_1, h_{11} and h_{12} of Hamming codes in case $l = 1$.

m	2	3	4	5	6	7	8	9	10
h_1	3	14	45	124	315	762	1785	4088	9207
h_{11}	3	11	31	79	191	447	1023	2303	5119
h_{12}	3	11	37	109	291	727	1737	4025	9127

Figure 5.1 h_1, h_{11} and h_{12} of Hamming codes in log10 in case $l = 1$.



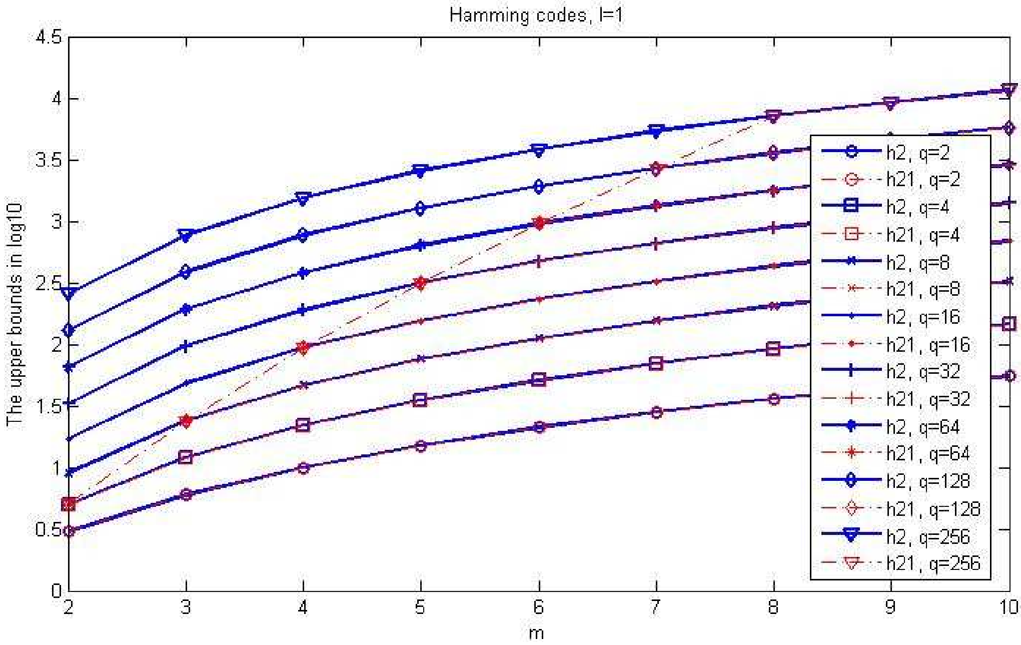
With Hamming codes, in case $l=1$, h_{11} is always smaller than h_{12}. Based on values in Table 5.1, we sketch Figure 5.1. In this figure (and other figures in this section), x axis denotes values of m while y axis denotes upper bounds in log10. From Table 5.1 and Figure 5.1, we can see that with greater values of m, *method 1* shows strong efficiency compared with *method 2.*

The values of h_2 and h_{21} with different m and q are given in Table 5.2 and Figure 5.2. Table 5.2 shows that h_{21} is much smaller than h_2 with large q and small m.

Table 5.2 h_2 and h_{21} of Hamming codes in case $l=1$.

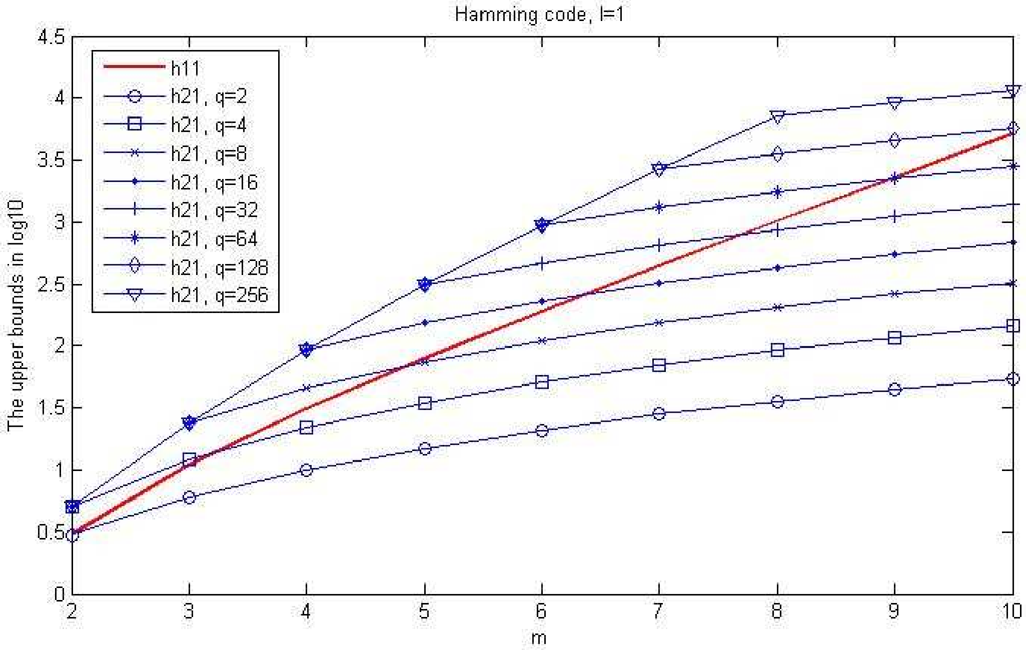
m	2	3	4	5	6	7	8	9	10
$h_2, q=2$	3	6	10	15	21	28	36	45	55
$h_{21}, q=2$	3	6	10	15	21	28	36	45	55
$h_2, q=4$	5	12	22	35	51	70	92	117	145
$h_{21}, q=4$	5	12	22	35	51	70	92	117	145
$h_2, q=8$	9	24	46	75	111	154	204	261	325
$h_{21}, q=8$	5	24	46	75	111	154	204	261	325
$h_2, q=16$	17	48	94	155	231	322	428	549	685
$h_{21}, q=16$	5	24	94	155	231	322	428	549	685
$h_2, q=32$	33	96	190	315	471	658	876	1125	1405
$h_{21}, q=32$	5	24	94	315	471	658	876	1125	1405
$h_2, q=64$	65	192	382	635	951	1330	1772	2277	2845
$h_{21}, q=64$	5	24	94	315	951	1330	1772	2277	2845
$h_2, q=128$	129	384	766	1275	1911	2674	3564	4581	5725
$h_{21}, q=128$	5	24	94	315	951	2674	3564	4581	5725
$h_2, q=256$	257	768	1534	2555	3831	5362	7148	9189	11485
$h_{21}, q=256$	5	24	94	315	951	2674	7148	9189	11485

Figure 5.2 h_2 and h_{21} of Hamming codes in log10 in case $l = 1$.



Because $s_l \le \min\{h_{11}, h_{12}, h_{21}\} = \min\{h_{11}, h_{21}\}$, the upper bounds on the 1st separating redundancies of Hamming codes can be determined by the comparison between h_{11} and h_{21}, which is illustrated in the next figure.

Figure 5.3 h_{11} and h_{21} of Hamming codes in log10 in case $l = 1$.



From Figure 5.3, we can see two important features. The first is that the role of h_{21} on the 1st separating redundancy is meaningless. Whenever the upper bounds are determined by h_{21}, $h_{21} = h_2$. In other words, whenever the condition $(q-1)^{i-1} \geq \binom{n}{l}$ satisfies, the new upper bounds are always determined by h_{11}. The second is that the role of h_{11} on upper bounds on the separating redundancy increases with greater values of q.

5.1.2 $l = 2$

In case $l = d - 1 = 2$, m should be equal or larger than 3. If $m = 2 \Rightarrow n = 3$, the punctured code has length equal to 1. Error (if it occurs) cannot be corrected or detected.

In this case, *method 2* cannot be applied and $h_{11} = h_1$. The values of h_1, h_2 and h_{21} of Hamming codes in case $l = 2$ are given in Table 5.3 while the comparison between h_2 and h_{21} is illustrated in Figure 5.4.

Table 5.3 h_{11}, h_1, h_2 and h_{21} of Hamming codes in case $l = 2$.

m	3	4	5	6	7	8	9	10	11
$h_{11} = h_1$	21	210	1395	7812	40005	194310	912135	4182024	18846729
$h_2, q = 2$	7	14	25	41	63	92	129	175	231
$h_{21}, q = 2$	7	14	25	41	63	92	129	175	231
$h_2, q = 4$	21	58	125	231	385	596	873	1225	1661
$h_{21}, q = 4$	21	58	125	231	385	596	873	1225	1661
$h_2, q = 8$	73	242	565	1091	1869	2948	4377	6205	8481
$h_{21}, q = 8$	45	242	565	1091	1869	2948	4377	6205	8481
$h_2, q = 16$	273	994	2405	4731	8197	13028	19449	27685	37961
$h_{21}, q = 16$	69	514	2405	4731	8197	13028	19449	27685	37961
$h_2, q = 32$	1057	4034	9925	19691	34293	54692	81849	116725	160281
$h_{21}, q = 32$	87	610	4965	19691	34293	54692	81849	116725	160281
$h_2, q = 64$	4161	16258	40325	80331	140245	224036	335673	479125	658361
$h_{21}, q = 64$	87	802	5285	40011	140245	224036	335673	479125	658361
$h_2, q = 128$	16513	65282	162565	324491	567189	906788	1359417	1941205	2668281
$h_{21}, q = 128$	87	1054	5925	40971	282709	906788	1359417	1941205	2668281

Figure 5.4 h_2 and h_{21} of Hamming codes in log10 in case $l=2$.

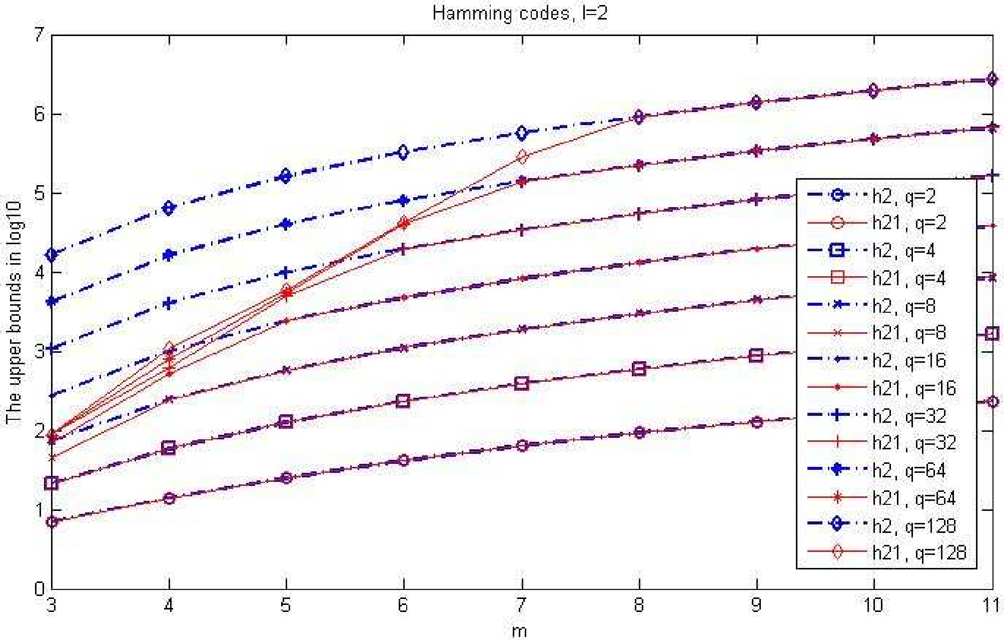
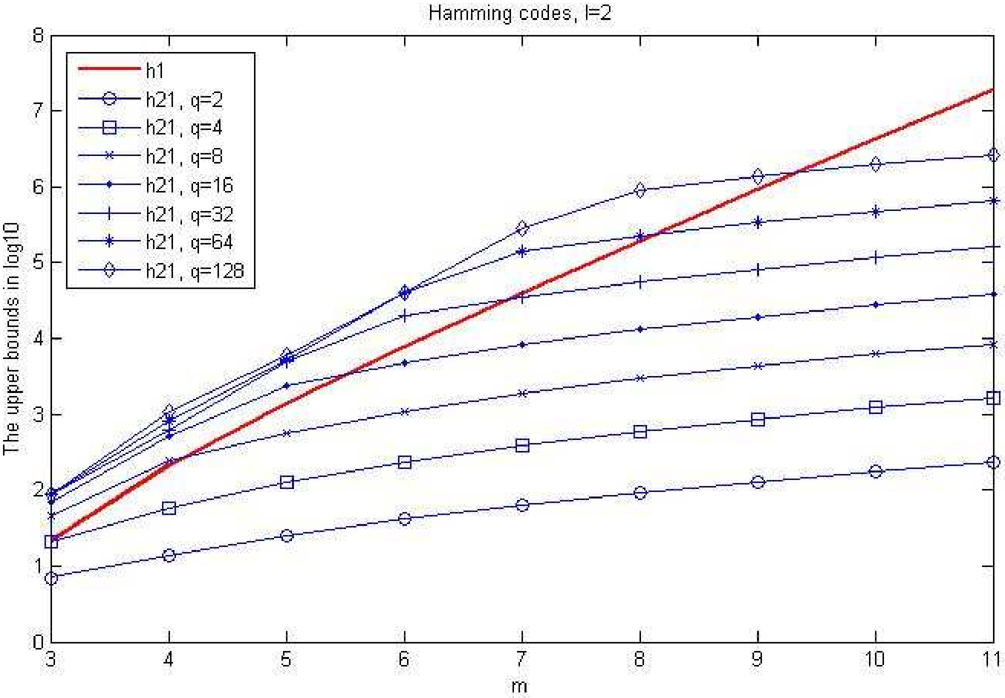


Figure 5.5 h_1 and h_{21} of Hamming codes in log10 in case $l=2$.



Similar to the case $l=1$, h_{21} is much smaller than h_2 with large q and small m. The comparison between h_1 and h_{21}, which is illustrated in Figure 5.5, will give information about the upper bounds on the 2nd separating redundancy.

Again, two features similar to the case $l = 1$ are repeated.

5.2 Reed-Muller Code (32,6,16)

Another class of linear block codes constructed in the early days for error correction and detection was the class of Reed-Muller codes. Reed-Muller codes were first invented for switching circuit design and error detection, but later on, they were reformulated for error correction and detection in communication and data storage systems. These codes, which are simple in construction and rich in structural properties, are useful for multiple random error correction.

Here, we examine upper bounds on the separating redundancy of the famous (32,6,16) Reed-Muller code. This code was used in spacecrafts of NASA from 1969 to 1977. A very prominent mission was Mariner 9, which was devoted to the photographic observation of the surface of Mars, [2]. It is a low rate code with good error correction capabilities. The Hamming distance of 16 is very high for a code of length 32. Because $d = 16$, l can receive values from 1 to 15.

In all figures from now, x axis denotes the values of l while y axis denotes the upper bounds in log10.

Table 5.4 The upper bounds for Reed-Muller code (32,6,16).

l	h_1	h_{11}	h_{12}	$h_2 = h_{21}, q = 2$	$h_2 = h_{21}, q = 4$
1	800	481	176	351	1001
2	11904	8142	9696	2951	24401
3	114080	70377	98900	17901	428051
4	791120	450549	772640	83681	5756231
5	4228896	2248456	4207896	313911	61702121
6	18123840	9031460	18101040	971711	541238321
7	63951264	29858271	63927324	2533986	3.9579e+009
8	189329400	82594013	189304920	5658536	2.4458e+010
9	476829600	193696948	476805120	10970271	1.2901e+011
10	1.0322e+009	389956198	1.0322e+009	18696431	5.8523e+011
11	1.9354e+009	684918913	1.9353e+009	28354131	2.2961e+012
12	3.1611e+009	1.0794e+009	3.1611e+009	38754731	7.8234e+012
13	4.5159e+009	1.6101e+009	4.5158e+009	48412431	2.3221e+013
14	5.6572e+009	2.4731e+009	5.6572e+009	56138591	6.0175e+013
15	6.2229e+009	6.2229e+009		61450326	1.3639e+014

From Table 5.4, we can see that with some values of l, *method 1* can reduce more than half the numbers of rows in the matrices H_l while the values of h_{12} are almost the same as h_1 except in case $l=1$. The greater value of l is, the more rows are reduced by *method 1*. In case $l=1$, the upper bound following *method 2* is extremely smaller than the one following *method 1*. The efficiencies of *method 1* and *method 2* are illustrated in Figure 5.6.

In case $q=2$ or $q=4$, $h_2 = h_{21}$. It means that the condition $(q-1)^{i-1} \geq \binom{n}{l}$ does not satisfy with $q=2,4;\ i=1,...,l+1$.

To have the upper bound on the lth separating redundancy, we should compare $\min\{h_{11}, h_{12}\}$ with h_{21}. The comparison is given in Figure 5.7.

In case $q=2$, the upper bound on the lth separating redundancy can be determined by $\min\{h_{11}, h_{12}, h_2\}$. From $q=4$ to greater values of q, h_{21} is extremely greater than $\min\{h_{11}, h_{12}\}$ which does not depend on q. It means that in case $q \geq 4$, $\min\{h_{11}, h_{12}, h_{21}\} = \min\{h_{11}, h_{12}\}$. The upper bound is always equal to $\min\{h_{11}, h_{12}\}$.

Figure 5.6 h_1, h_{11} and h_{12} of Reed-Muller code (32,6,16) in log10.

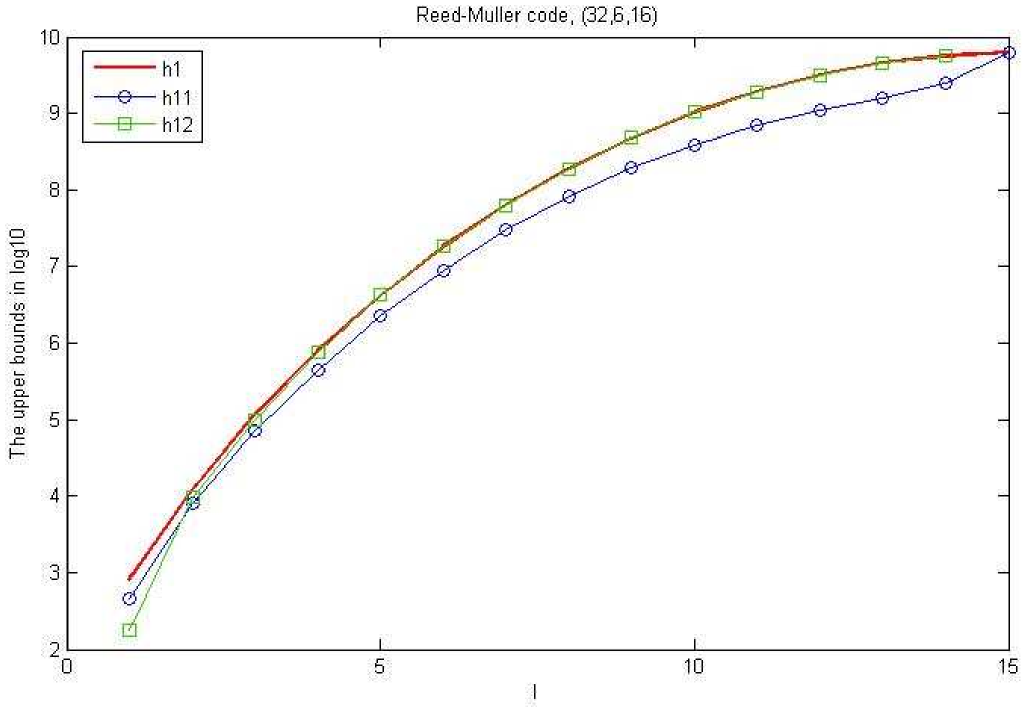
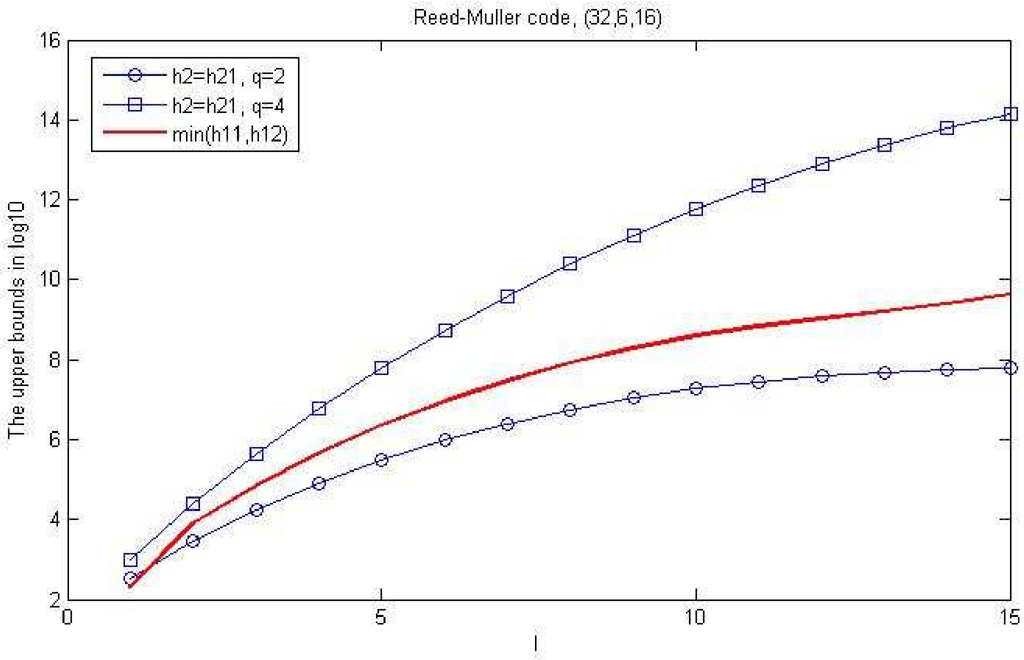


Figure 5.7 h_2, h_{21} and $\min\{h_{11}, h_{12}\}$ of Reed-Muller code (32,6,16) in log10.



5.3 LDPC (20,7,6)

Low-density parity-check (LDPC) codes are the codes which achieve an error performance only a fraction of a decibel away from the Shannon limit, [2]. The main feature of this class of codes is that parity-check matrices have a small density of non-zero symbols. This class of code is applied to many communication and digital storage systems where high reliability is required.

Here, we consider (20,7,6) LDPC code.

Because $d = 6$, l can receive values from 1 to 5.

Table 5.5 The upper bounds for LDPC code (20,7,6).

l	1	2	3	4	5
h_1	240	2090	11400	43605	124032
h_{11}	112	1538	7848	30293	124032
h_{12}	97	1650	9780	41877	
$h_2 = h_{21}, q = 2$	91	377	1092	2379	4095
$h_2 = h_{21}, q = 4$	247	2821	22126	126373	543361

Figure 5.8 h_1, h_{11} and h_{12} of LDPC code (20,7,6) in log10.

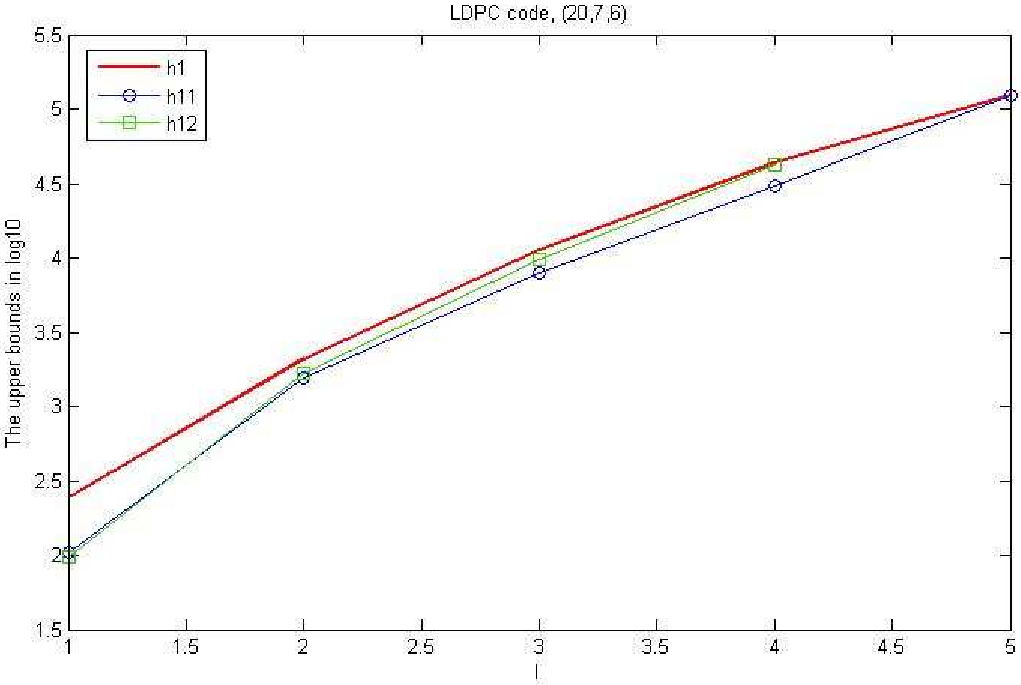
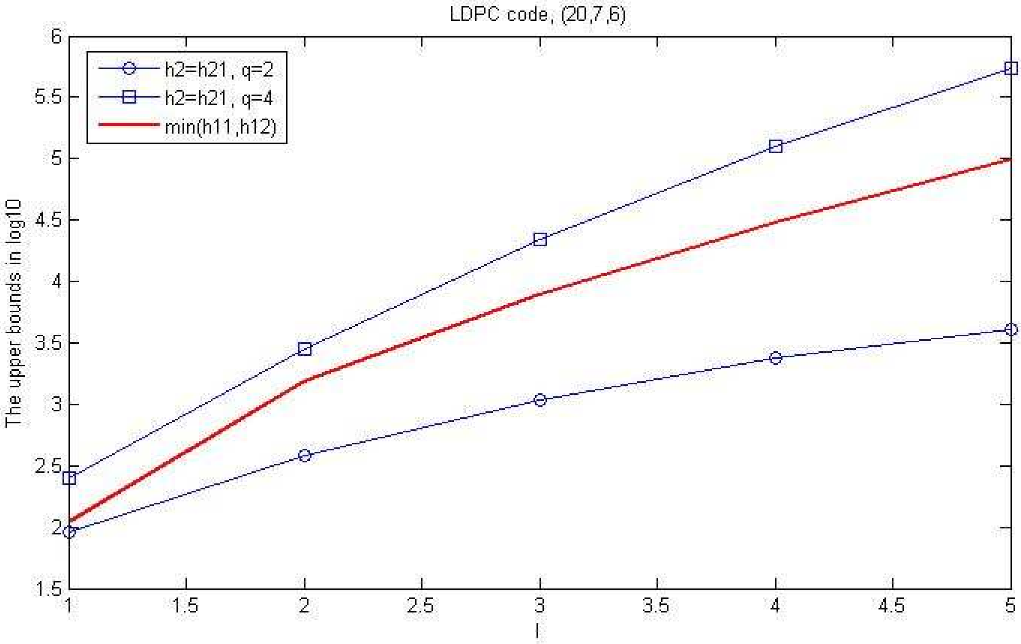


Figure 5.9 h_2, h_{21} and $\min\{h_{11}, h_{12}\}$ of LDPC code (20,7,6) in log10.



As in case of (32,6,16) Reed-Muller code, *method 2* gives better result in comparison with *method 1* just in case $l = 1$.

In case $q = 2$ or $q = 4$, $h_2 = h_{21}$.

Figure 5.9 illustrates that in case $q = 2$, h_2 is always smaller than $\min\{h_{11}, h_{12}\}$. Hence, the upper bound on the lth separating redundancy is h_2. From $q = 4$ to greater values of q, h_{21} is always greater than $\min\{h_{11}, h_{12}\}$. Hence, the upper bound on the lth separating redundancy is $\min\{h_{11}, h_{12}\}$.

5.4 Golay Code, (23,12,7)

In this part, we consider the (23,12,7) Golay code which is constructed by M. J. E. Golay in 1949. This code has many interesting structural properties and has been extensively studied by many coding theorists and mathematicians. It has been used in many real communication systems for error control. More information on this code can be found in [11].

Because $d = 7$, l can receive values from 1 to 6.

Table 5.6 The upper bounds for Golay code (23,12,7).

l	1	2	3	4	5	6
h_1	230	2277	14168	61985	201894	504735
h_{11}	135	1782	10248	42705	140329	504735
h_{12}	131	1989	12992	60809	200844	
$h_2 = h_{21}, q = 2$	66	231	561	1023	1485	1815
$h_2 = h_{21}, q = 4$	176	1661	10571	47993	160259	400829
$h_2, q = 8$	396	8481	121671	1230933	8995767	47819937
$h_{21}, q = 8$	396	8481	121671	1230933	8995767	42308277

Figure 5.10 h_1, h_{11} and h_{12} of Golay code (23,12,7) in log10.

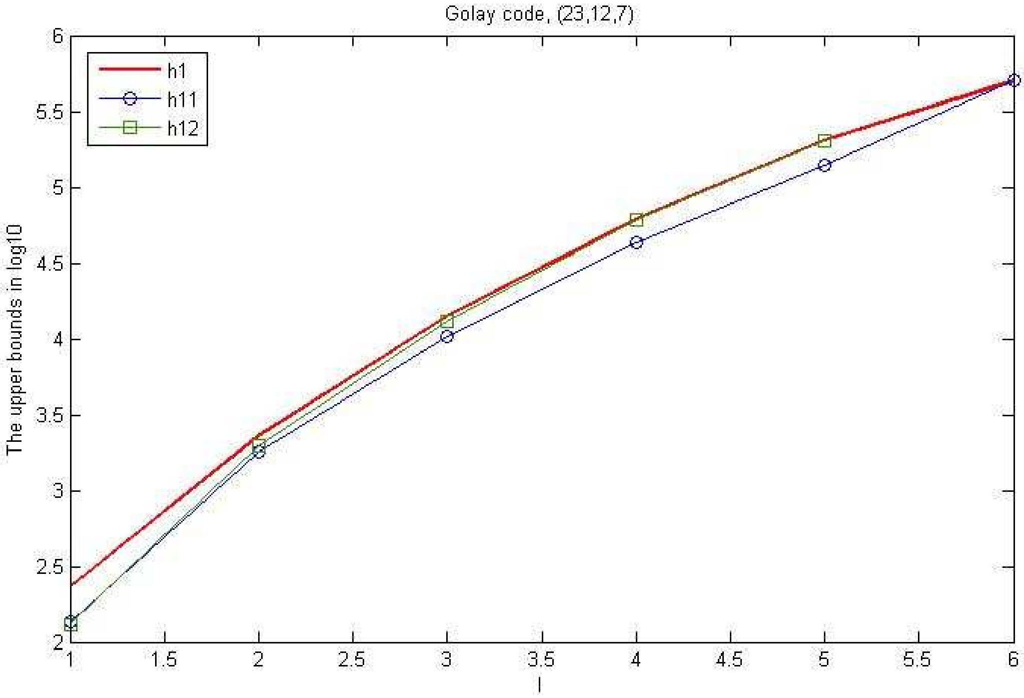
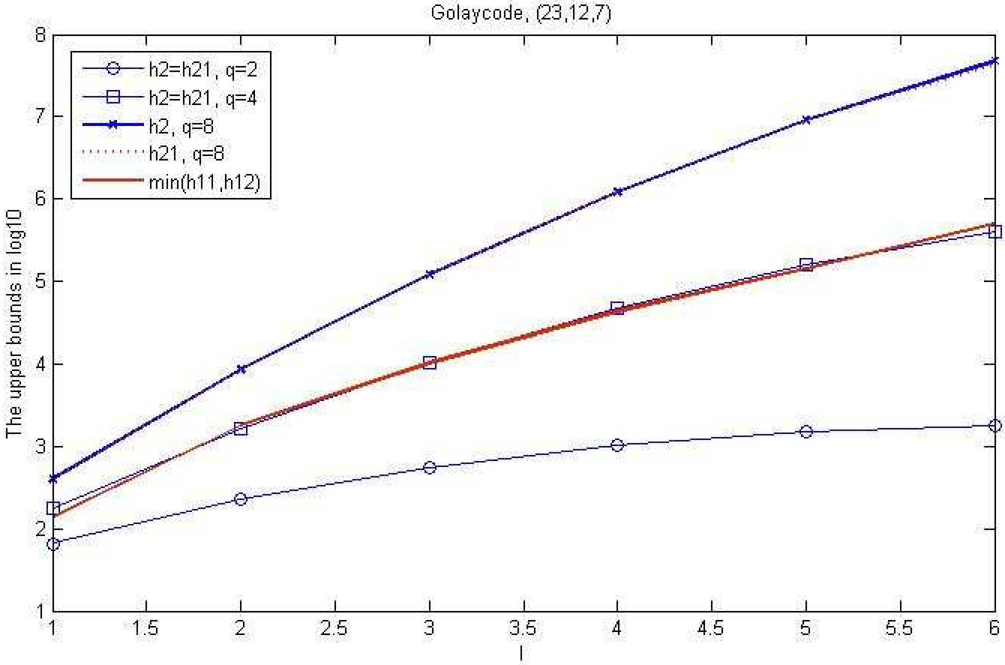


Figure 5.11 h_2, h_{21} and $\min\{h_{11},h_{12}\}$ of Golay code (23,12,7) in log10.



Because the values of n and d of this code are similar to (20,7,6) LDPC code, again, except $l=1$, *method 1* always gives better results in comparison with *method 2*. Figures 4.8 and 4.10 have the same features.

Table 5.6 shows that in case $q=2$ and $q=4$, $h_2=h_{21}$. From $q=8$, the values of h_{21} and h_2 differ in great values of l.

From Figure 5.11, we can see that in case $q=2$, $h_2<\min\{h_{11},h_{12}\}$. The upper bound is given by h_2. However, in case $q=4$, we should compare the values of h_2 with $\min\{h_{11},h_{12}\}$ to find the upper bound on the lth separating redundancy.

From $q=8$ to greater values of q, the upper bound is always determined by $\min\{h_{11},h_{12}\}$.

5.5 BCH Code, (63,45,7)

The Bose, Chaudhuri, and Hocquenghem (BCH) codes form a practical class of powerful random error-correcting codes, particularly if the expected number of errors is small compared with the length. For extensive information, we refer to [11], [12].

Here, we choose the (63,45,7) BCH code to examine the new upper bounds on the separating redundancy of a code with long length but low Hamming distance.

Because $d=7$, l can receive values from 1 to 6.

Table 5.7 The upper bounds for BCH code (63,45,7).

l	1	2	3	4	5	6
h_1	1071	31248	595665	8339310	91375011	815346252
h_{11}	447	27660	509937	7062798	78809415	815346252
h_{12}	783	29328	582435	8324022	91358631	
$h_2=h_{21}, q=2$	171	987	4047	12615	31179	63003
$h_2=h_{21}, q=4$	477	7821	90441	784449	5295501	28495197
$h_2=h_{21}, q=8$	1089	41073	1090653	21662421	333667569	4.0777e+009

Figure 5.12 h_1, h_{11} and h_{12} of BCH code (63,45,7) in log10.

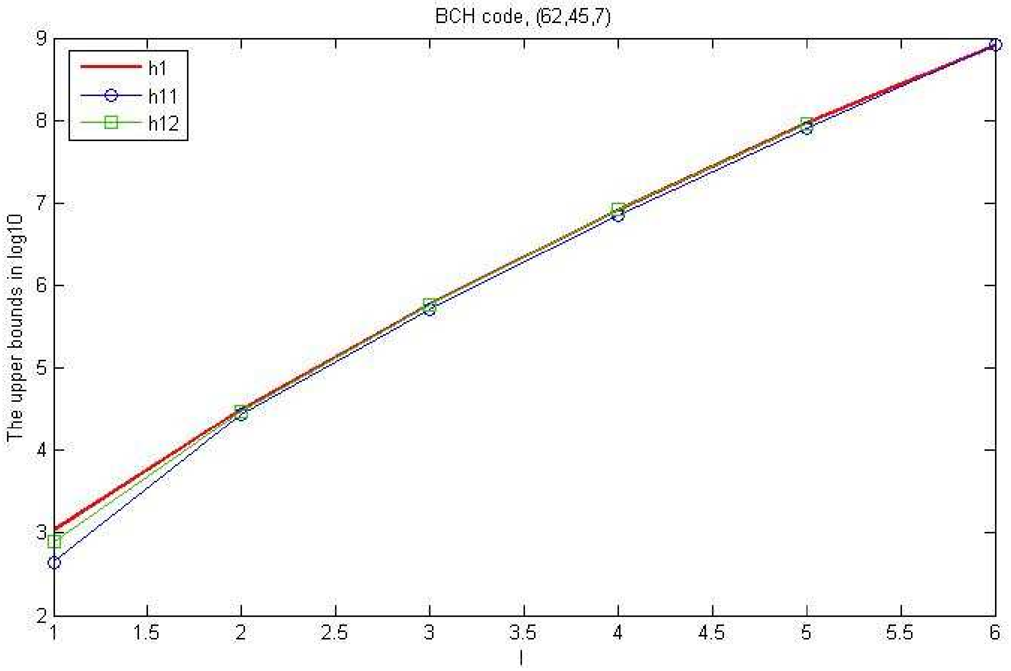
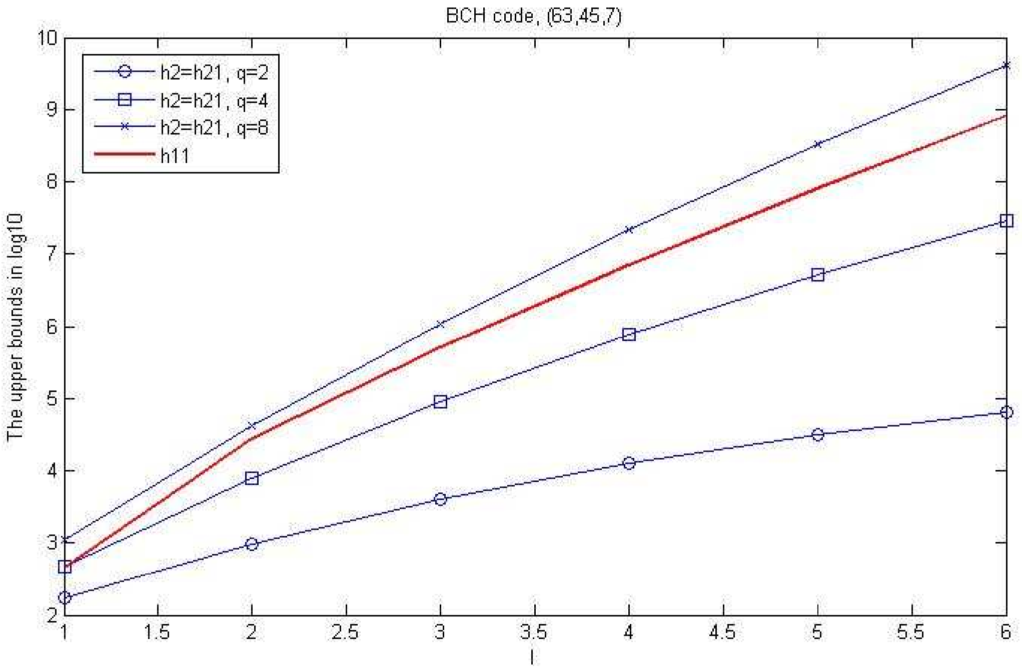


Figure 5.13 h_2, h_{21} and $\min\{h_{11}, h_{12}\}$ of BCH code (63,45,7) in log10.



We can see that the efficiencies of *method 1* and *method 2* depend on the relation between the values of l and n. In this case, *method 1* always gives better results in comparison with *method 2* but it does not show strong efficiency as in (32,6,16) Reed-Muller code due to the great value of n with respect to l. This is illustrated in Figure 5.12.

Table 5.7 shows that in case $q=2$, $q=4$ and $q=8$, $h_2 = h_{21}$. On the other hand, in case $q=2$ and $q=4$, $h_2 < h_{11}$. Hence, the upper bound is given by h_2. From $q=8$ to greater values of q, the upper bound is always determined by h_{11}.

Similarly to all cases above, the role of h_{21} on the separating redundancy is meaningless.

5.6 Conclusions

From the data above, we can conclude that for these codes:

- Except some codes in case $l=1$, *method 1* always gives the smaller upper bounds in comparison with *method 2*.
- The new upper bound in the second construction does not lead to the improvement because whenever the condition $(q-1)^{i-1} > \binom{n}{l}$ satisfies, the upper bound on the lth separating redundancy is always determined by the new upper bound in the first construction.

Chapter 6

Conclusions

6.1 Contributions of the Thesis

Separating parity-check matrices are useful for decoding over channels causing both errors and erasures. The separating parity-check matrices with small number of rows reduce not only decoding complexity but also memory storage at the en/decoders. In this thesis, we proposed two methods which can give the improved upper bounds on the separating redundancy being the minimum number of rows in a separating matrix. Besides, we also presented a *covering design*. This design is not optimal but it can give a general upper bound on the *covering number* being the minimum size of a *covering design*. Applying this result to the upper bound obtained from *method 1*, we can reduce more than half the upper bound given by (2.1) of some codes, i.e. the Reed-Muller code (32,6,16). In some cases, *method 2* gives a better result in comparison with *method 1* when the number of erasures is one. We also computed the upper bounds on the separating redundancies of some famous practical codes.

6.2 Future directions

The new upper bound derived from the argument of calculating the number of useful rows in an l-separating matrix in the second construction does not lead to the improvement. The appropriate value of b for each l is still a question. Besides, there are still gaps between the upper and lower bounds. Hence, more research on bounding techniques for the separating redundancy is required. Our methods give the general upper bounds for all codes. However, the better bounds on the separating redundancy of codes with special characteristics can be achieved. In addition, the number of distinct rows in a separating matrix of a code relates to the weight distribution of its dual code. Therefore, the question of how to apply the weight distribution needs to be researched. Potential future work also includes the determination of the bounds for classes of codes of practical interest.

References

[1] K.A.S. Abdel-Ghaffar and J.H. Weber, “Separating erasures from errors for decoding,” Proceedings of the IEEE International Symposium on Information Theory, Toronto, Canada, pp. 215-219, July 6-11, 2008.

[2] J. H. Weber, *Lecture Notes: Error-Correcting Codes*, Delft University of Technology, 2007.

[3] Jeffrey H. Dinitz, Douglas R. Stinson, *Contemporary Design Theory, A Collection of Surveys*, A Wiley-Interscience Publication, 1992.

[4] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, “Finite-length analysis of low-density parity-check codes on the binary erasure channel,” *IEEE Trans. Inform. Theory*, vol. 48, no. 6, pp. 1570-1579, June 2002.

[5] J. Han and P. H. Siegel, “Improved upper bounds on stopping redundancy,” *IEEE Trans. Inform. Theory*, vol. 53, no. 1, pp. 90-104, January 2007.

[6] H. D. L. Hollmann and L. M. G. M. Tolhuizen, “On parity check collections for iterative erasure decoding that correct all correctable erasure patterns of a given size,” *IEEE Trans. Inform. Theory*, vol. 53, no. 2, pp. 823-828, February 2007.

[7] La Jolla Covering Repository, *http://www.ccrwest.org/cover.html*.

[8] R. M. Roth, *Introduction to Coding Theory*. Cambridge, UK: Cambridge University Press, 2006.

[9] M. Schwartz and A. Vardy, “On the stopping distance and the stopping redundancy of codes,” *IEEE Trans. Inform. Theory*, vol. 52, no. 3, pp. 922-932, March 2006.

[10] J. H. Weber and K. A. S. Abdel-Ghaffar, “Results on parity-check matrices with optimal stopping and/or dead-end set enumerators,” *IEEE Trans. Inform. Theory*, vol. 54, no. 3, pp. 1368-1374, March 2008.

[11] S. Lin and D. J. Costello, Jr. , *Error Control Coding*, Pearson Education International, 2004.

[12] F. J. Mac Williams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland Publishing Company, 1977.

[13] S. A. Vanstone and P. C. van Oorschot, *An Introduction to Error-Correcting Codes with Applications*, Norwell, MA: Kluwer, 1989.

Appendix

Some of the results presented in thesis have appeared in paper "*New Upper Bounds on the Separating Redundancy of Linear Block Codes*" of the 30th Symposium on Information Theory in the Benelux, which was held at the Eindhoven University of Technology at Eindhoven, The Netherlands, on May 28 and 29, 2009.