Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics

Comparing Sequential and Cooperative Erasure Repairing

(Dutch title: De Vergelijking tussen Verlies Sequentieel en Coöperatief Herstellen)

A thesis submitted to the
Delft Institute of Applied Mathematics
in partial fulfillment of the requirements

for the degree

BACHELOR OF SCIENCE
in
APPLIED MATHEMATICS

by

Erik ten Hagen

Delft, the Netherlands
September 2018

BSc thesis APPLIED MATHEMATICS

"Comparing Sequential and Cooperative Erasure Repairing"

(Dutch title:"De Vergelijking tussen Verlies Sequentieel en Coöperatief Herstellen")

ERIK TEN HAGEN

Delft University of Technology

Supervisor

Dr.ir. J.H. Weber

Thesis committee

Dr. D.C. Gijswijt Drs. E.M. van Elderen

September, 2018 Delft

Preface

You are reading the thesis 'Comparing Sequential and Cooperative Erasure Repairing' which has been written as part of my bachelor in Applied Mathematics at Delft University of Technology. I have been working on this project between April and August of 2018.

The reason I chose to work on this subject for my thesis is because I liked the course "Applied Algebra: Codes en Cryptosystems". Here we had a introduction in coding theory, which I enjoyed and therefore wanted to work on it during my thesis.

I would like to thank J. Weber for his supervision of this project and to thank him for his feedback. I also would like to thank the rest of the thesis committee for reviewing my thesis.

Abstract

Information is spatially distributed over data servers and for many services online it has to be available at all times, but those servers are not always available. If we store the information in a smart way, we might be able to still get our information even if we can not reach the servers.
There are two factors we have to keep in mind when we restore the servers and information. Those are the repair bandwidth, this is the amount of data you need to download to repair a failed server, and the repair degree, this is the amount of other servers you have to access before you can repair your server. We will look at two methods for restoring information with focusing on the repair degree, which means to access the least amount of other servers.
First we discuss how we can repair one and multiple failures or erasures using the cooperative and sequential repairing method. Then we discuss the parameters for a sequential locally repairable code and its locality or repair degree. Next we will discuss the parameters for the Hamming code and the extended Hamming code and their locality for which we have constructed a function to calculate the generalized Hamming weight for $\kappa \leq 3$. Now we can compare the sequential locally repairable code with the Hamming code for two erasures and we can compare the sequential locally repairable code with the extended Hamming code for three erasures. The result is that the sequential locally repairable code has a much lower locality than the Hamming code and the extended Hamming code, but they have a higher information rate.

Contents

1 Introduction

In a distributed storage system, the data of a single file is spatially distributed over multiple nodes or storage units. It could be possible that a node or storage unit is temporary unavailable because too many people try to access the node, because the node may crash or restart or because the node is under maintenance.

In [1] where they studied the Facebook cluster, they found that they had a median of 50 nodes that were unavailable per day. To repair or access the data we have to use other nodes. Before we check other nodes we have to keep in mind two important factors. First is the repair bandwidth; that is the amount of data you need to download to repair a failed node and the second is the repair degree, which is the amount of other nodes you have to access before you can repair your node [2].

A simple idea for storing information against a failure is using backups, duplicating every node. Now if one node fails we can repair it with its duplicate. If multiple failures happen we can sometimes repair them and sometimes not. If the failures happen in two distinct nodes we can repair them just like with only one failure bit if the two failures happen, one in a node and one in its backup we can not repair them. We see here that each node can be repaired with one other node, but this is generally not the case.

In this thesis we have looked at codes that can repair multiple erasures and the amount of nodes/symbols you need to repair them.
We will discuss the cooperative method of erasure repairing and the sequential method of erasure repairing and discuss codes that can repair two and three erasures using these two methods.

In Chapter 2 we will give some information about linear codes. In Chapter 3 we will construct a function which we will later use to calculate the size of repair sets for the cooperative method. Chapter 4 will provide an introduction in erasure repairing for both one and multiple erasures. Chapters 5 and 6 will elaborate on two codes, one code to repair erasures sequential and the other code to repair erasures cooperative for both two and three erasures. In Chapter 7 we will compare them with each other. Finally in Chapter 8 we will look at a linear code where you can only repair erasures with two symbols.

2 Linear Codes

When we are talking about codes, there are a few properties we are interested in. First of all the length of a code and the information it contains. For linear codes we have length n and the information it has is called dimension k and because we are working binary we have $F_2^k \subset F_2^n$. This gives our code structure.
The information or message we send looks like $\boldsymbol{u} = (u_1, ..., u_k)$ with $u_i \in F_2$ and the information is protected against erasures with $n-k$ bits. To protect our message we use a function $f : F_2^k \to F_2^n$, this gives us $f(\boldsymbol{u}) = \boldsymbol{v}$ with $\boldsymbol{v}$ is the code word corresponding to $\boldsymbol{u}$. All the possible code words gives us our code.

2.1 Generator and Parity-check matrix

Because we are working in a linear space each u_i in $\boldsymbol{u}$ is linked with a vector $\boldsymbol{g_i}$ and depending on which u_i are ones our code word $\boldsymbol{v}$ is the sum of $\boldsymbol{g_i}$. When we put these vectors in a matrix we get G and $\boldsymbol{v} = \boldsymbol{u}G$. Now we have G is a $k \times n$ matrix with $\boldsymbol{g_i}$ as row vectors. G is called the generator matrix of this code and in this paper the generator matrix are all in standard form, meaning that they they look like

$$G = [I_k, X]$$

We are also interested in the parity-check matrix. This is a $(n-k) \times n$ matrix and for every code word $\boldsymbol{c}$ in C we get $H \cdot c^\top = 0$.
Because G is in standard form we can construct H as follows

$$H = \left[X^\top, I_{n-k}\right]$$

Example 1. If we look at a $[5, 4]-$code. Then its generator matrix is a 4×5 matrix with a I_4

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

and the parity-check matrix is a $1 \times n$ matrix with I_1.

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

This code is known as the parity code in which every row vector has an even amount of ones.

2.2 Weight, Distance, Information Rate and Redundancy

A code word $\boldsymbol{c}$ for a $[n, k, d]$ linear code C has ones in different positions. The support of a code word is a set existing of all the ones in $\boldsymbol{c}$.

$$\chi(\boldsymbol{c}) = \{i \in [1, .., n] | c_i = 1\}$$

The weight or Hamming weight of the code word $\boldsymbol{c}$ is the amount of ones it has. We denote this as $wt(\boldsymbol{c})$. This is equal to $|\chi(\boldsymbol{c})|$.
The distance between two code words $\boldsymbol{v}$ and $\boldsymbol{w}$ is the amount of positions in which they differ. If we want to determine the distance of a code C, we have to find the smallest distance between two code words in C. The distance of a linear code is equal to the minimum weight of a nonzero code word in C.

Example 2. If we look at $\boldsymbol{v} = (0, 0, 1)$ and $\boldsymbol{w} = (1, 1, 1)$, then we see that the weight of $\boldsymbol{v}$ is one and the weight of $\boldsymbol{w}$ is three. When we want to know the distance between two code words we can count the amount of positions they differ or we can calculate the weight of $\boldsymbol{v}-\boldsymbol{w}$. $\text{wt}(\boldsymbol{v}-\boldsymbol{w}) = wt(110) = 2$.

Instead of looking at the support of one vector we can look at the support of $D \subseteq C$.

$$\chi(D) = \{i : \exists(x_1, x_2, ..., x_n) \in D, x_i = 1\}$$

Definition 1. For a $[n, k, d]$ linear code C and $1 \leq \kappa \leq k$, the κ^{th} generalized Hamming weight is

$$d_\kappa(C) = |\chi(D)|$$

with D a subcode of C with rank κ.

There is a small difference between the usual definition of generalized Hamming codes [3] and ours. Normally the generalized Hamming weight is the smallest $|\chi(D)|$ but we are interested in using the generalized Hamming weight for κ independent vectors. We will have D is the span of κ independent vectors.
For $\kappa = 1$ we get the weight of one vector and this the same as the weight of a code word.

The information rate and redundancy of a code tells us something about the information a code contains and the extra bits of a code. The information rate is the ratio between the dimension of a code and its length. The redundancy is the difference between the length of a code and its dimension.

Example 3. If we have the $[5, 4]$-code then the information rate of the code is $\frac{k}{n} = \frac{4}{5} = 0.8$ and the redundancy of the code is $n - k = 5 - 4 = 1$.

3 Analysis of the Generalized Hamming Weight

In this Chapter we analyse the generalized Hamming weight for $\kappa \leq 3$. We will need this to measure the size of repair sets in Chapter 6. For one vector we can calculate the amount of positions that are one using the weight of the vector, but for multiple vectors we can not easily calculate all the positions. Therefore we will construct a function that can calculate the generalized Hamming weight for three or less vectors.

Definition 2. If $\boldsymbol{a}, \boldsymbol{b}$ and $\boldsymbol{c}$ are vectors in F_2^n, then $Z(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) = \{i \in \{1, .., n\} | a_i = 1 \vee b_i = 1 \vee c_i = 1\}$.

For the generalized Hamming weight we are only interested in $|Z(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})|$

Lemma 1. *Let a,b and c be vectors in F_2^n, then*

$$|Z(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})| = \frac{wt(\boldsymbol{a}) + wt(\boldsymbol{b}) + wt(\boldsymbol{c}) + wt(\boldsymbol{a}+\boldsymbol{b}) + wt(\boldsymbol{a}+\boldsymbol{c}) + wt(\boldsymbol{b}+\boldsymbol{c}) + wt(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})}{4}.$$

Proof. Let $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ be vectors in F_2^n and define $A = \chi(\boldsymbol{a})$ and $B = \chi(\boldsymbol{b})$ and $C = \chi(\boldsymbol{c})$.
Notice that $\mathrm{wt}(\boldsymbol{a}) = |A|$ and $\mathrm{wt}(\boldsymbol{b}) = |B|$ and $\mathrm{wt}(\boldsymbol{c}) = |C|$.

Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be vectors in F_2^n and $U = \chi(\boldsymbol{u})$ and $V = \chi(\boldsymbol{v})$ and define $S_1(U, V) = U \cup V \backslash \{U \cap V\}$. Then we have $wt(\boldsymbol{u} + \boldsymbol{v}) = |S_1(U, V)|$ and

$$S_1(U, V) = \{i \in \{1, .., n\} | (u_i = 1 \wedge v_i = 0) \vee (u_i = 0 \wedge v_i = 1)\}$$

Define a second set

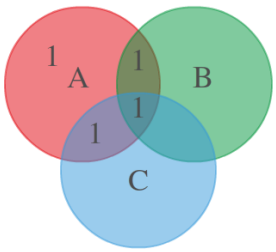
$$S_2(A, B, C) = \{(A \cup B \cup C) \backslash \big\{(A \cap B) \backslash C \cup (A \cap C) \backslash B \cup (B \cap C) \backslash A\big\}\}$$

This set is equivalent with

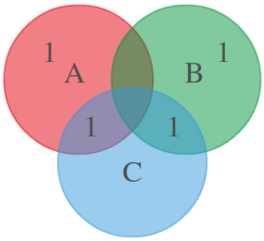
$$S_2(A, B, C) = \left\{ i \in \{1, .., n\} \middle| \begin{matrix} (a_i = 1 \wedge b_i = 1 \wedge c_i = 1) \vee (a_i = 1 \wedge b_i = 0 \wedge c_i = 0) \vee \\ (a_i = 0 \wedge b_i = 1 \wedge c_i = 0) \vee (a_i = 0 \wedge b_i = 0 \wedge c_i = 1) \end{matrix} \right\}$$

Notice that $wt(\boldsymbol{a} + \boldsymbol{b} + \boldsymbol{c}) = |S_2(A, B, C)|$

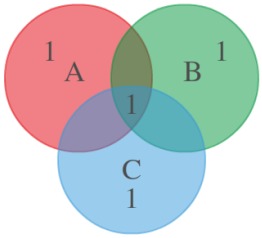
The sets counted in $wt(\boldsymbol{a})$ are the ones that are only in A. Those are the ones in A and not B and C and the ones in A and B but not in C and the ones in A and C and not in B and the ones in A,B and C. Which we can see in Figure 3.1a.
In Figure 3.1b we see the sets of ones we count in $wt(\boldsymbol{a} + \boldsymbol{b})$ and in Figure



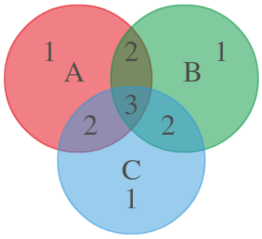
(a) The sets counted in $wt(\boldsymbol{a})$



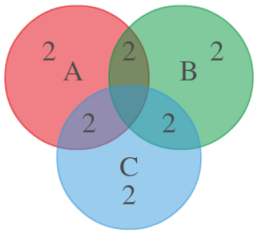
(b) The sets counted in $wt(\boldsymbol{a}+\boldsymbol{b})$



(c) The sets counted in $wt(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})$



(d) The sets counted in $wt(\boldsymbol{a})+wt(\boldsymbol{b})+wt(\boldsymbol{c})$



(e) The sets counted in $wt(\boldsymbol{a}+\boldsymbol{b})+wt(\boldsymbol{a}+\boldsymbol{c})+wt(\boldsymbol{b}+\boldsymbol{c})$

Figure 3.1: The visual representation of the sets we count in the weight of one or more vectors.

3.1c we see the sets of ones we count in $wt(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})$.

If we count the sets $wt(\boldsymbol{a})$ and $wt(\boldsymbol{b})$ and $wt(\boldsymbol{c})$, we count some sets twice or even thrice. Those are the ones in the position that are in $\boldsymbol{a}$ and $\boldsymbol{b}$ or in $\boldsymbol{a}$,$\boldsymbol{b}$ and $\boldsymbol{c}$.
If we add the sets of ones in Figure 3.1c with the sets of ones in Figure 3.1d and with the sets of ones in Figure 3.1e. We count each set four times. We get

$$4\cdot|Z(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})| = wt(\boldsymbol{a})+wt(\boldsymbol{b})+wt(\boldsymbol{c})+wt(\boldsymbol{a}+\boldsymbol{b})+wt(\boldsymbol{a}+\boldsymbol{c})+wt(\boldsymbol{b}+\boldsymbol{c})+wt(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})$$

$$|Z(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})| = \frac{wt(\boldsymbol{a})+wt(\boldsymbol{b})+wt(\boldsymbol{c})+wt(\boldsymbol{a}+\boldsymbol{b})+wt(\boldsymbol{a}+\boldsymbol{c})+wt(\boldsymbol{b}+\boldsymbol{c})+wt(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})}{4} \tag{1}$$

□

Example 4. Lets check (1) for two vectors. Let $\boldsymbol{a} = (1,0,1,0,0,0,1,1,0,1,1)$ and $\boldsymbol{b} = (0,1,1,0,0,1,1,0,0,0,1)$, then $\boldsymbol{a}+\boldsymbol{b} = (1,1,0,0,0,1,0,1,0,1,0)$. We can easily check $Z(\boldsymbol{a},\boldsymbol{b},\boldsymbol{0}) = \{1,2,3,6,7,8,10,11\}$ and $|Z(\boldsymbol{a},\boldsymbol{b},\boldsymbol{0})| = 8$. Because $\boldsymbol{c} = \boldsymbol{0}$ we only need to know the weight of $\boldsymbol{a}$,$\boldsymbol{b}$ and $\boldsymbol{a}+\boldsymbol{b}$ to check

(1). $wt(\boldsymbol{a}) = 6$, $wt(\boldsymbol{b}) = 5$ and $wt(\boldsymbol{a} + \boldsymbol{b}) = 5$.

$$|Z(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{0})| = \frac{6 + 5 + 0 + 5 + 6 + 5 + 5}{4} = \frac{32}{4} = 8$$

Example 5. For three vectors look at $\boldsymbol{a} = (1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1)$ and $\boldsymbol{b} = (0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1)$ and $\boldsymbol{c} = (1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0)$, then $\boldsymbol{a} + \boldsymbol{b} = (1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0)$, $\boldsymbol{a} + \boldsymbol{c} = (0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1)$, $\boldsymbol{b} + \boldsymbol{c} = (1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1)$ and $\boldsymbol{a} + \boldsymbol{b} + \boldsymbol{c} = (0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0)$. Now we have $Z(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) = \{1, 2, 3, 4, 6, 7, 8, 9, 10, 11\}$ and $|Z(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})| = 10$. To check the formula (1) we need the weight of all these vectors, those are $wt(\boldsymbol{a}) = 6$, $wt(\boldsymbol{b}) = 5$, $wt(\boldsymbol{c}) = 5$, $wt(\boldsymbol{a} + \boldsymbol{b}) = 5$, $wt(\boldsymbol{a} + \boldsymbol{c}) = 5$, $wt(\boldsymbol{b} + \boldsymbol{c}) = 8$, $wt(\boldsymbol{a} + \boldsymbol{b} + \boldsymbol{c}) = 6$.

$$|Z(a, b, c)| = \frac{6 + 5 + 5 + 5 + 5 + 8 + 6}{4} = \frac{40}{4} = 10$$

The result is the same as counting the set $Z(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$.

4 Erasure Repairing

If you send a code word $\boldsymbol{c} = (c_1, ..., c_n)$, the receiver will not always get $\boldsymbol{c}$ but instead will get $\boldsymbol{v} = (v_1, .., v_n)$ with $v_i =?$ for the erasures on place $i \in \{1, .., n\}$ and $v_j = c_j$ on the other places. The set of the erasures exist of the indices where the erasures take place $E \subseteq \{1, ..., n\}$. The erasures can be repaired by combining some other symbols of the code word. Which symbols you can use depends on the code and the method of repairing. We will look at repairing one erasure and two methods of repairing multiple erasures. Those are cooperative [4] and sequential erasure repairing [5].

The amount of erasures we can solve depends on the distance of the code. We can always solve $t = d-1$ erasures. Although we may have a code word with more erasures that we can repair, this will not be the case for all code words in the code.

4.1 One Erasure Repairing

If we have one erasure this means we have received a the word $\boldsymbol{v} = (v_1, .., v_n)$ with on place i we have $v_i =?$. Thus we have $E = \{i\}$. To repair one erasure we have to look at the parity-check matrix for our code. For every vector $\boldsymbol{h}$ in the row space of the parity-check matrix we know $\sum_{j=1}^{n} c_j \cdot h_j = 0$. If $h_j = 0$ then c_j can be anything. So the only c_j that are important are the ones for which $h_j = 1$.
To solve one erasure we want to have a vector $\boldsymbol{h}$ in the row space of H such that $h_i = 1$. Because then we have $v_1 \cdot h_1 + ... + v_i \cdot h_i + ... + v_n \cdot h_n = 0$. Because we do not know v_i we get $v_i = h_1 \cdot v_1 + ... + h_{i-1} \cdot v_{i-1} + h_{i+1} \cdot v_{i+1} + ... + h_n \cdot v_n$. Now here we have $1 < i < n$. But the same principle works for $i = 1$ and $i = n$.
The repair set for v_i is $R = \{j \in \{1, ..., n\} \backslash \{i\} | h_j = 1\}$.

Definition 3. A $[n, k, d]$ linear code has locality r for one erasure if for every set E with $|E| = 1$ the repair set $|R| \leq r$.

Example 6. If we look at the $[7, 4, 3]$-code. This code is called a Hamming code and is generated by

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{2}$$

and its parity-check matrix is

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

If we receive $\boldsymbol{v} = (0, 0, 1, 1, ?, 1, 0)$, then $E = \{5\}$ and we need to find a $\boldsymbol{h}$ in the row space of H such that $h_5 = 1$. We can use $\boldsymbol{h} = (1, 1, 1, 0, 1, 0, 0)$. We know that $h_1 \cdot v_1 + ... + v_n \cdot h_n = 0$ and if we fill in the values of h_j we get $v_1 + v_2 + v_3 + v_5 = 0$ and thus we know $v_5 = v_1 + v_2 + v_3$. Now we check the values in v and we get $v_5 = 0 + 0 + 1 = 1$. So the code word was $\boldsymbol{v} = (0, 0, 1, 1, 1, 1, 0)$. The repair set is $R = \{1, 2, 3\}$ and for one erasure this code has locality 3.

4.2 Cooperative Erasure Repairing

For repairing erasures cooperative for the code word $\boldsymbol{c}$, we have to look at the row space of the parity-check matrix H. The vectors we want to use should have for each $j \in E$ and $l \in E \backslash \{j\}$ the values $c_j = 1$ and $c_l = 0$. If we name these vectors $\boldsymbol{w_j}$ and the i-th bit of this vector w_{j_i} then we can use them to restore c_j. If we define $R_j = \{i \in \{1, ..., n\} | i \neq j \wedge w_{j_i} = 1\}$, then R_j is the repair set for erasure j and $c_j = \sum_{i \in R_j} c_i$. The repair set for the erasures is $R = \cup_{j \in E} R_j$.

Definition 4. A $[n, k, d]$ linear code C has locality r_c if for every set E the repair set for cooperative erasure repairing has size r_c or less.

Because we can calculate the weight of two and three vectors, we can determine the size of the repair set. For two erasures we need a $\boldsymbol{a}$ and $\boldsymbol{b}$ in the row space of H, which satisfy the requirements to solve the erasures, then $r_c = |Z(\boldsymbol{a}, \boldsymbol{b}, 0)| - 2$. For three erasures we need $\boldsymbol{a}, \boldsymbol{b}$ and $\boldsymbol{c}$ in the row space of H, that satisfy the requirements. Then the repair set size is $r_c = |Z(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})| - 3$.

Example 7. We look at the $[7, 4, 3]$ binary Hamming code with generator matrix (2) and parity-check matrix (3).
If we receive the word $\boldsymbol{v} = (0, 1, ?, ?, 0, ?, 1)$. Then we can solve the erasures cooperative. First notice $E = \{3, 4, 6\}$. Now we have to find three vectors such that one vector has $c_3 = 1, c_4 = 0, c_6 = 0$ and one vector has $c_3 = 0, c_4 = 1, c_6 = 0$ and one vector that has $c_3 = 0, c_4 = 0, c_6 = 1$.
We can find these vectors in the row space of (3). The row space consist of the following vectors.

$$
\begin{aligned}
\boldsymbol{h_1} &= (1,1,1,0,1,0,0)\\
\boldsymbol{h_2} &= (1,1,0,1,0,1,0)\\
\boldsymbol{h_3} &= (1,0,1,1,0,0,1)\\
\boldsymbol{h_1}+\boldsymbol{h_2} &= (0,0,1,1,1,1,0)\\
\boldsymbol{h_1}+\boldsymbol{h_3} &= (0,1,0,1,1,0,1)\\
\boldsymbol{h_2}+\boldsymbol{h_3} &= (0,1,1,0,0,1,1)\\
\boldsymbol{h_1}+\boldsymbol{h_2}+\boldsymbol{h_3} &= (1,0,0,0,1,1,1)
\end{aligned}
$$

The three vectors we will use are $\boldsymbol{h_1}$, $\boldsymbol{h_1}+\boldsymbol{h_3}$ and $\boldsymbol{h_1}+\boldsymbol{h_2}+\boldsymbol{h_3}$. These vectors gives us the following equations.

$$
\begin{aligned}
c_3 &= c_1+c_2+c_5\\
c_4 &= c_2+c_5+c_7\\
c_6 &= c_1+c_5+c_7
\end{aligned}
$$

The repair set is $R=\{c_1,c_2,c_5,c_7\}$. If we check what those values are in our received word we can conclude our erased symbols are

$$
\begin{aligned}
c_3 &= 0+1+0=1\\
c_4 &= 1+0+1=0\\
c_6 &= 0+0+1=1
\end{aligned}
$$

So the code word was $(0,1,1,0,0,1,1)$.
In this situation those three vectors where the only possible vectors to repair these erasures. To repair the erasures we have used a total of four symbols, therefore the repair set is $R=\{c_1,c_2,c_5,c_7\}$. For small codes it is easy to check the size of the repair set, but for larger codes this could be time consuming. We could have used (1) to calculate the amount of symbols in the repair set.

4.3 Sequential Erasure Repairing

To solve erasures sequential for a word $\boldsymbol{c}$ we have to look at the row space of the parity-check matrix. The vectors should satisfy the following criteria. First vector should have for $j_1 \in E$ the value $c_{j_1}=1$ and for all $l_1 \in E\backslash\{j_1\}$

the value $c_{l_1} = 0$, then you can use this vector to solve the first erasure. For the second vector we need $j_2 \in E\backslash\{j_1\}$ with $c_{j_2} = 1$ and for all $l_2 \in E\backslash\{j_1, j_2\}$ the value $c_{l_2} = 0$, then you can solve the second erasure. Now for the i-th vector you have $j_i \in E\backslash\{j_1, ..., j_{i-1}\}$ with $c_{j_i} = 1$ and for all $l_i \in E\backslash\{j_1, ..., j_i\}$ with $c_{l_i} = 0$, then you can use that vector to solve the i-th erasure. If we have found the vector $\boldsymbol{w}$ for the j_i erasure and let w_b be the b-th bit of $\boldsymbol{w}$ and lets define $R_{j_i} = \{b \in \{1, ..., n\} | b \neq j_i \wedge w_b = 1\}$, then we can calculate that $c_{j_i} = \sum_{b \in R_{j_i}} c_b$.

Definition 5. A $[n, k, d]$ linear code C has locality r_s for sequential erasure repairing if for every erasure $j \in E$ the repair set R_j has size r_s or less.

Notice that for sequential erasure repairing the locality r_s only depends on the repair set of one erasure and for cooperative erasure repairing the locality r_c depends on the repair set of all erasures.

Example 8. Lets look at the code generated by (2) and the parity-check matrix (3). If we receive the word $\boldsymbol{v} = (0, 1, ?, ?, 0, ?, 1)$, then we can repair this with a few different repair sets.
The first vector we can choose is one of the three vectors in Example 7. Lets use $\boldsymbol{h_1}$. now we can solve the third symbol. $c_3 = c_1 + c_2 + c_5 = 0 + 1 + 0 = 1$
For the second erasure we can use $\boldsymbol{h_1} + \boldsymbol{h_3}$ just like with cooperative, but we can also use $\boldsymbol{h_3}$, this gives us $c_4 = c_1 + c_3 + c_7 = 0 + 1 + 1 = 0$.
For the last erasure we can use the rows $\boldsymbol{h_2}$, $\boldsymbol{h_1} + \boldsymbol{h_2}$, $\boldsymbol{h_1} + \boldsymbol{h_3}$, $\boldsymbol{h_1} + \boldsymbol{h_2} + \boldsymbol{h_3}$. If we use $\boldsymbol{h_2}$ it gives us $c_6 = c_1 + c_2 + c_4 = 0 + 1 + 0 = 1$ and we find that the code word was $(0, 1, 1, 0, 0, 1, 1)$.
Notice that for each of the three vectors we have used by cooperative, we can use two vectors to solve the second erasure and four vectors to solve the fourth erasure. So there are $3 \cdot 2 \cdot 4 = 24$ ways to repair these three erasures. Also because all the vectors in the row space of (3) has a weight of 4, the repair sets all have size 3.

As mentioned in Section 4, we can always solve $t = d - 1$ erasure. For the code we just mentioned we can solve $t = d - 1 = 3 - 1 = 2$ erasures. But we just saw in Example 7 and in Example 8 that it could solve 3 erasures. This only happens because we used the right code words, it is not always possible to repair three erasures. For example if we have received $v = (0, 0, ?, 0, ?, 0, ?)$. When we look at the generator matrix (2) we see on the third row the vector $(0, 0, 1, 0, 1, 0, 1)$. This is a possible code word for v, but the vector $(0, 0, 0, 0, 0, 0, 0)$ is also in our code. So we can not always repair three erasures.

For sequential erasure repairing we can use three different kind of vectors for repairing two erasures. The first vector has on the position of the first

erasure $c_{E_1} = 0$ and on the position of the second erasures $c_{E_2} = 1$. The second kind of vector has $c_{E_1} = 1$ and $c_{E_2} = 0$ and the third kind of vector has $c_{E_1} = 1$ and $c_{E_2} = 1$. For each of these sort of vectors there are two vectors with these properties.
This gives us three possible combination to solve the erasures. First one is we use the first kind of vectors with the second kind. Then you have a vector with $c_{E_1} = 0$ and $c_{E_2} = 1$ and also a vector with $c_{E_1} = 1$ with $c_{E_2} = 0$. Here we can solve the erasures both at the same time, just like cooperative.
We can also use the first kind of vectors with the third kind of vectors, this gives us a vector with $c_{E_1} = 0$ and $c_{E_2} = 1$ and also a vector with $c_{E_1} = 1$ and $c_{E_2} = 1$. Now we have to solve the second erasure first and then the first erasure.
Last option we can use is the second kind of vectors with the third kind of vectors, this gives us a vector with $c_{E_1} = 1$ and $c_{E_2} = 0$ and also a vector with $c_{E_1} = 1$ and $c_{E_2} = 1$. Now we have to solve the first erasure first and then the second erasure.

These are the three possible combinations to repair two erasures. We like to give each of the vectors its own letter to count all the possible repair sets. For the first kind of vector we can check in Section 7 there are two vectors with this property, call them $\boldsymbol{a_1}$ and $\boldsymbol{a_2}$. For the second kind of vector there are also two vectors, call them $\boldsymbol{b_1}$ and $\boldsymbol{b_2}$ and for the third vector there are also two vectors with this property, call them $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$. Then the possible combinations are $(\boldsymbol{a_1}, \boldsymbol{b_1})$, $(\boldsymbol{a_1}, \boldsymbol{b_2})$, $(\boldsymbol{a_1}, \boldsymbol{c_1})$, $(\boldsymbol{a_1}, \boldsymbol{c_2})$, $(\boldsymbol{a_2}, \boldsymbol{b_1})$, $(\boldsymbol{a_2}, \boldsymbol{b_2})$, $(\boldsymbol{a_2}, \boldsymbol{c_1})$, $(\boldsymbol{a_2}, \boldsymbol{c_2})$, $(\boldsymbol{b_1}, \boldsymbol{c_1})$, $(\boldsymbol{b_1}, \boldsymbol{c_2})$, $(\boldsymbol{b_2}, \boldsymbol{c_1})$ and $(\boldsymbol{b_2}, \boldsymbol{c_2})$. So we got 12 different repair sets for 2 erasures.

5 Codes for Sequential Erasure Repairing

In this section we will discuss a certain family of sequential locally repairable codes, which is $(n, k, r, t) - SLRC$. This is a linear code of length n, dimension k, locality r and can solve always up to t erasures. This is introduced in [5] and we will call it SLRC or SLR-code. A key component of this family of SLRC is that they all have a code rate of at least $\frac{r}{r+t}$.
First we will discuss this code for two erasures and for three erasures,then we explain why the lower bound is interesting and then compare the size of the repair sets for these two codes.

5.1 Construction of the Code

For two and three erasures we have that the length

$$n = r^m \sum_{s=0}^{t} \frac{1}{r^{|supp_m(s)|}} \tag{4}$$

with the dimension $k = r^m$. The length and dimension depends on r,t and m. Here we have t as the amount of erasures the code can repair. m is any positive integer such that $t \leq 2^m - 1$. For both $t = 2$ and $t = 3$ we have $2 \leq m$. the $supp_m(s)$ is the set of ones in the binary representation of s with length m. So the only difference between two and three erasures is the value of t in

$$\sum_{s=0}^{t} \frac{1}{r^{|supp_m(s)|}}$$

For two erasures this is

$$\begin{aligned} &\sum_{s=0}^{2} \frac{1}{r^{|supp_m(s)|}} \\ =&1 + \frac{1}{r^{|supp_m(1)|}} + \frac{1}{r^{|supp_m(2)|}} \\ =&1 + \frac{1}{r} + \frac{1}{r} \\ =&1 + \frac{2}{r} = \frac{r+2}{r} \end{aligned} \tag{5}$$

For three erasures we get

$$\begin{aligned}
&\sum_{s=0}^{3} \frac{1}{r^{|supp_m(s)|}} \\
=&1 + \frac{1}{r^{|supp_m(1)|}} + \frac{1}{r^{|supp_m(2)|}} + \frac{1}{r^{|supp_m(3)|}} \\
=&1 + \frac{1}{r} + \frac{1}{r} + \frac{1}{r^2} \\
=&\frac{r^2 + 2 \cdot r + 1}{r^2} \\
=&\left(\frac{r+1}{r}\right)^2
\end{aligned} \tag{6}$$

In Table 5.1 we see a list of parameters for the codes that exist for two and three erasures. Because the code is constructed depending on r we know for each code how big the repair set is.

		SLRC for two erasures		SLRC for three erasures	
Size of r	value for m	n	k	n	k
2	2	8	4	9	4
	3	16	8	18	8
	4	32	16	36	16
3	2	15	9	16	9
	3	45	27	48	27
	4	135	81	144	81
4	2	24	16	25	16
	3	78	52	100	64
5	2	35	25	36	25
6	2	48	36	49	36
7	2	63	49	64	49
8	2	80	64	81	64
9	2	99	81	100	81
10	2	120	100	121	100

Table 5.1: A list of parameters of existing SLR-code for two and three erasures with $2 \leq r \leq 10$

5.2 The Upper Bound and the Lower Bound

For cooperative erasure repairing we have the upper bound

$$\frac{k}{n} \leq \frac{r}{r+t} \tag{7}$$

according to Rawat, Mazumdar, and Vishwanath in [6].
But for the family of our sequential erasure repairing code in Section 5.1 we have the lower bound

$$\frac{k}{n} \geq \frac{r}{r+t} \tag{8}$$

The reason to why the SLR-code has a higher information rate then what you would expect for the cooperative repairable code is that the SLR-code is constructed in a different way than the normal cooperative codes.

Our information rate is

$$\frac{k}{n} = \frac{r^m}{r^m \sum_{s=0}^{t} \frac{1}{r^{|supp_m(s)|}}} = \frac{1}{\sum_{s=0}^{t} \frac{1}{r^{|supp_m(s)|}}} \tag{9}$$

If we combine (9) with what we have learned for $t = 2$ in (5) we get $\frac{k}{n} = \frac{r}{r+2}$. So for $t = 2$ we get the lower bound. Instead of looking at the lower bound $\frac{r}{r+t}$ we are interested in $\frac{1}{\frac{r+t}{r}}$ because then we can compare $\frac{r+t}{r}$ with $\sum_{s=0}^{t} \frac{1}{r^{|supp_m(s)|}}$.
If t increases by one then $\frac{r+t}{r}$ increases with $\frac{1}{r}$, however if t increases for $\sum_{s=0}^{t} \frac{1}{r^{|supp_m(s)|}}$, the sum only increases with $\frac{1}{r^{|supp_m(t)|}}$ and because $|supp_m(t)| \geq 1$ we know that its always same or less then $\frac{1}{r}$. So $\frac{r+t}{r} \geq \sum_{s=0}^{t} \frac{1}{r^{|supp_m(s)|}}$ for $t \geq 2$ and thus we get

$$\frac{r}{r+t} \leq \frac{1}{\sum_{s=0}^{t} \frac{1}{r^{|supp_m(s)|}}}$$

and this proves (8).

5.3 The Difference between the Locality for Two and Three Erasures

When we rewrite the information rate for two respectively three erasures, $\frac{k}{n} = \frac{r}{r+2}$ and $\frac{k}{n} = \left(\frac{r}{r+1}\right)^2$ such that it only depends on $\frac{k}{n}$, we can see that the locality r changes for codes with a high redundancy against codes with a low redundancy. For two erasures we get

$$r = \frac{2 \cdot \frac{k}{n}}{1 - \frac{k}{n}} \tag{10}$$

for three erasures we get

$$r = \frac{\frac{k}{n} + \sqrt{\frac{k}{n}}}{1 - \frac{k}{n}} \tag{11}$$

The difference between the size of the repair sets for the variable $\frac{k}{n}$ is

$$\frac{\frac{k}{n} + \sqrt{\frac{k}{n}}}{1 - \frac{k}{n}} - \frac{2 \cdot \frac{k}{n}}{1 - \frac{k}{n}} = \frac{\sqrt{\frac{k}{n}} - \frac{k}{n}}{1 - \frac{k}{n}}$$

we know that $0 < \sqrt{\frac{k}{n}} < 1$ and therefore we know $\sqrt{\frac{k}{n}} - \frac{k}{n} < 1 - \frac{k}{n}$ and thus

$$0 < \frac{\sqrt{\frac{k}{n}} - \frac{k}{n}}{1 - \frac{k}{n}} < 1 \tag{12}$$

Because the difference between the SLR-code for two erasures and the SLR-code for three erasures is smaller than one, we know that for codes with the same information rate that the repair set size is at most one larger. In Figure 5.1 we can see how the size of the repair set changes for codes with a low and high information rate.

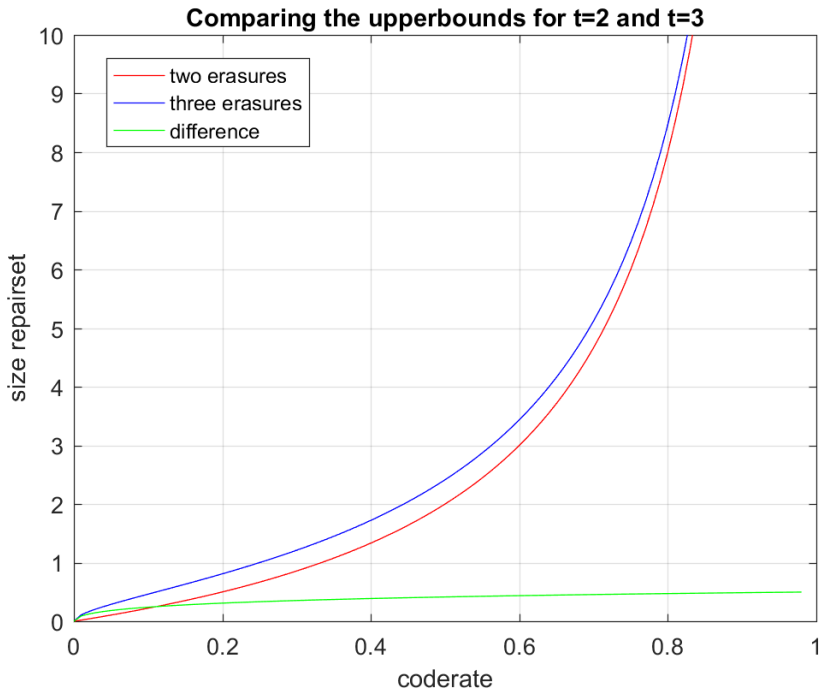


Figure 5.1: The upper bound for two and three erasures and the difference between them in one figure.

6 Codes for Cooperative Erasure Repairing

For cooperative erasure repairing we discuss the Hamming code because this code has a distance of 3 and therefore can repair up to two erasures. We will also look at the extended Hamming code because this code has a distance of 4 and therefore can repair three erasures.

6.1 Hamming Codes

The binary Hamming code is a code with length $n = 2^m - 1$ and dimension $2^m - 1 - m$ for $m \geq 2$.
The columns of the parity check matrix exist of $2^m - 1$ non-zero distinct vectors in F_2^m. Because of this all the row vectors have a weight of 2^{m-1}.
Because of [7] we know that the locality of the code is $r_c = 3 \cdot 2^{m-2} - 2$ for repairing two erasures. Our method for calculating r_c gives us

$$\begin{aligned} r_c =& Z(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{0}) - 2 \\ =& \frac{wt(\boldsymbol{a}) + wt(\boldsymbol{b}) + wt(\boldsymbol{0}) + wt(\boldsymbol{a}+\boldsymbol{b}) + wt(\boldsymbol{b}+\boldsymbol{0}) + wt(\boldsymbol{a}+\boldsymbol{0}) + wt(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{0})}{4} - 2 \\ =& \frac{2^{m-1} + 2^{m-1} + 0 + 2^{m-1} + 2^{m-1} + 2^{m-1} + 2^{m-1}}{4} - 2 \\ =& \frac{6 \cdot 2^{m-1}}{4} - 2 \\ =& 3 \cdot 2^{m-2} - 2 \end{aligned} \tag{13}$$

and because of his method we have come up with (1) that can calculate the weight of two and three vectors, which we will use to calculate r_c for extended Hamming codes.

In Table 6.1 we see a list of parameters for the Hamming codes and the amount of symbols it takes to repair two erasures.

6.2 Extended Hamming Codes

As mentioned in [8] when you extend a code C with a $k \times n$ generator matrix G you get C^* with a $k \times (n+1)$ generator matrix G^*. The generator matrix looks like $G^* = [G, \boldsymbol{b}]$ with $\boldsymbol{b}$ is created such that each row of G^* has an even

weight. If H is the parity-check matrix for C then H^* is the parity-check matrix for C^*.

$$H^* = \begin{bmatrix} H & 0 \\ j & 1 \end{bmatrix}$$

with $\boldsymbol{j}$ a $1 \times n$ row of ones. If the distance d of C is odd, then the distance of C^* will be $d + 1$.

Because the Hamming code is a $[2^m - 1, 2^m - 1 - m, 3]$ code we get $[2^m, 2^m - 1 - m, 4]$ which is the extended Hamming code. For c_i with $i \in \{1, .., n\}$ the amount of symbols for repairing an erasure does not change, because for the vector $\boldsymbol{h}$ in the row space of H we used to repair c_i we can now use the vector $\boldsymbol{h'} = [\boldsymbol{h}, 0] \in H^*$. We see that $wt(\boldsymbol{h}) = wt(\boldsymbol{h'})$ and therefore r_c does not change.

For c_{n+1} we see a vector $\boldsymbol{h^*}$ in H^* with only ones, so c_{n+1} can be repaired with n symbols. If we take a $\boldsymbol{h}$ in the row space of H^* with $c_{n+1} = 0$, then $wt(\boldsymbol{h^*} + \boldsymbol{h}) = wt(\boldsymbol{h^*}) - wt(\boldsymbol{h}) = n - wt(\boldsymbol{h}) = 2^m - 2^{m-1} = 2^{m-1}$. The vector $\boldsymbol{h_*} + \boldsymbol{h}$ has weight 2^{m-1} and $c_{n+1} = 1$, therefore c_{n+1} can be repaired with $2^{m-1} - 1$ other symbols. Because this works for every $\boldsymbol{h}$ with $c_{n+1} = 0$ we can always find a vector of weight 2^{m-1} to solve c_{n+1}. So for the extended Hamming code one erasure can be repaired with $2^{m-1} - 1$ symbols.

Because the dimension of this code is 4 we know that there are three vectors $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ in the row space of H^* such that it can solve three erasures. We can use (1) to calculate $r_c = |Z(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})| - 3$. Because $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ have weight 2^{m-1} and every combination also have weight 2^{m-1} we get r_c is

$$\begin{aligned} r_c =& |Z(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})| - 3 \\ =& \frac{7 \cdot 2^{m-1}}{4} - 3 \\ =& \frac{7 \cdot 2^{m-1}}{2^2} - 3 \\ =& 7 \cdot 2^{m-3} - 3 \end{aligned} \tag{14}$$

In Table 6.1 we see a list of parameters of the extended Hamming codes and the amount of symbols it takes to restore three erasures.

value for m	Hamming Code			Extended Hamming Code		
	n	k	r_c	n	k	r_c
3	7	4	4	8	4	4
4	15	11	10	16	11	11
5	31	26	24	32	26	25
6	63	57	46	64	57	53
7	127	120	94	128	120	109

Table 6.1: A list of parameters for the Hamming codes and the extended Hamming code including its length, dimension and the locality.

7 Differences Between the SLRC and Hamming Codes for Two and Three Erasures

7.1 Codes with the same Length

If we compare Table 5.1 with Table 6.1 for codes that can repair two erasures with the same length, then we find two Hamming codes and two SLRC with the same length and for three erasures we also find two Hamming codes and SLRC with the same length. As you can see in Table 7.1. If we look at the locality of the codes for two erasures we see a big difference. If the repair sets for SLRC are disjunct then you still need less symbols to repair all the erasures than the Hamming code.
$2 \cdot r_s < r_c$ with r_s as the locality of SLRC and r_c as the locality of Hamming codes. This is the same for three erasures. $3 \cdot r_s < r_c$.

We also have to look at the redundancy, this is the difference between the length of a code and the information it holds. So for the $[63, 57]$ Hamming code we see that the redundancy is 6 and for the $[63, 49]$ SLRC we see that the redundancy is 14, which is more than twice as much as that of the Hamming code. So the trade off between these two code is one code has can store more information but to repair erasures you need more bits to repair them against a code that can store less information but if there are erasures, you can restore those with less bits than the other.

Hamming code	locality r_c	redundancy $n-k$	SLRC	locality r_s	redundancy $n-k$
Two erasures					
[15,11]	10	4	[15,9]	3	6
[63,57]	46	6	[63,49]	7	14
Three erasures					
[16,11]	11	5	[16,9]	3	7
[64,57]	53	7	[64,49]	7	15

Table 7.1: the locality and redundancy of the SLRC and Hamming codes for both two and three erasures.

7.2 Codes with the same Information Rate

For codes with almost the same information rate we find two codes we can compare. First for two erasures we get a Hamming code $[31, 26]$. This code has a information rate of $\frac{k}{n} \approx 0.839$ and a repair set of $r_c = 24$. We can also find a SLR-code $[120, 100]$. This code has a information rate of $\frac{k}{n} \approx 0.833$ and a repair set of $r_s = 10$ as shown is Table 7.2.
We have also found two codes for three erasures to compare. Now we have the extended Hamming code $[32, 26]$ with an information rate of $\frac{k}{n} \approx 0.183$ and a repair set of $r_c = 25$. For the SLR-code we have $[100, 81]$ with an information rate $\frac{k}{n} = 0.81$ and a repair set of $r_s = 9$ as shown in Table 7.2.
For both two and three erasures we see that the Hamming code can get a high information rate with a small code. Also for two erasures the repair set for SLRC is always smaller than the Hamming code because $2 \cdot r_s < r_c$, but for three erasures we see for the first time that $3 \cdot r_s \nleq r_c$. So if we have three erasures in the $[100, 81]$ SLRC and all the repair sets for the individual erasures are disjunct we have a bigger repair set than the Hamming code. Otherwise the combined repair sets for SLRC is most likely to be still smaller than the Hamming code.

Erasures	Hamming or SLRC	Information rate	code parameters	locality
2	Hamming	0.839	$[31, 26]$	$r_c = 24$
2	SLRC	0.833	$[120, 100]$	$r_s = 10$
3	Extended Hamming	0.813	$[32, 26]$	$r_c = 25$
3	SLRC	0.81	$[100, 81]$	$r_s = 9$

Table 7.2: A list of parameters for codes with almost the same information rate.

7.3 Codes with the same Dimension

We have also looked at codes with the same or almost the same dimension. For two erasures we found the Hamming codes $[7, 4]$ and $[31, 26]$ and for the SLRC $[8, 4]$ and $[45, 27]$ and for three erasures we found the extended Hamming codes $[8, 4]$ and $[32, 26]$ and for SLRC $[9, 4]$ and $[48, 27]$. In Table 7.3 we see that the redundancy of Hamming codes and extended Hamming codes are smaller than their locality and for the SLRC we see that the locality of the code is smaller than their redundancy. We saw this already in Table 7.1 for codes with the same length. The only difference is with the $[8, 4]$ extended Hamming code and the $[9, 4]$ SLR-code we see that $3 \cdot r_s \nleq r_c$. So if the repair sets for the SLRC were all disjunct, they need more symbols

to repair all erasures in total than the cooperative methods.

Hamming or SLRC	code parameters	redundancy	locality
two erasures			
Hamming	$[7, 4]$	3	$r_c = 4$
SLRC	$[8, 4]$	4	$r_s = 2$
Hamming	$[31, 26]$	5	$r_c = 24$
SLRC	$[45, 27]$	18	$r_s = 3$
three erasures			
Extended Hamming	$[8, 4]$	4	$r_c = 4$
SLRC	$[9, 4]$	5	$r_s = 2$
Extended Hamming	$[32, 26]$	8	$r_c = 25$
SLRC	$[48, 27]$	21	$r_s = 3$

Table 7.3: A list of parameters for codes with almost the same dimension for two erasures.

8 The Influence of Limiting the Repair Set Size for One Erasures

In this section we will look first at the $[7, 3, 4]$ code where every erasure can only be repaired with two symbols. Second we will look at the same $[7, 3, 4]$ code but now each erasure can be repaired with two or more symbols. Then we look at the locality of this code if we want to repair multiple erasures.

8.1 $[7, 3, 4]$-Code with Limited Repair Set for One Erasure

Until now we have looked at the parity-check matrix to solve erasures, but we will take a difference approach to restore the erasures. The $[7, 3, 4]$ code is generated with

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \tag{15}$$

To repair the erasures for this code we will look at Figure 8.1. This figure is constructed such as the sum of the column vectors shown on each line (including the circle) is equal to zero. Just like with the parity-check matrix when one column vector is erased we can construct it with the other columns on one line. Each of these columns represent a position in the code words. For example the vector $(1, 1, 0)^\top$ correspond with c_6, the sixth position of the code.
If we have lost a bit in our code word we check which vector corresponds with that position, then we will look at the lines containing that vector and then we can know which other vectors and thus which position can repair our erasure. This figure was constructed and used in [9].

Example 9. If we receive $\boldsymbol{c} = (0, 0, 1, 1, ?, 0, 1)$ then we have lost the fifth bit. First we check (15) to see what the fifth column is. This is $(1, 0, 1)^\top$. Now we want to look at Figure 8.1 and check the lines which contains $(1, 0, 1)^\top$. There are three lines that contain $(1, 0, 1)^\top$, those lines are

$$L_1 = \left\{ \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \right\} \tag{16}$$

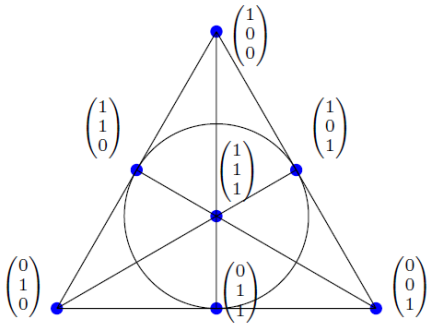


Figure 8.1: [7,3,4]-code repair triangle

$$L_2 = \left\{ \begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}1\\0\\1\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix} \right\} \tag{17}$$

$$L_3 = \left\{ \begin{pmatrix}1\\1\\0\end{pmatrix}, \begin{pmatrix}0\\1\\1\end{pmatrix}, \begin{pmatrix}1\\0\\1\end{pmatrix} \right\} \tag{18}$$

We can convert these vectors to the position in the generator matrix. This gives us

$$L_1 = \{c_2, c_7, c_5\} \tag{19}$$

$$L_2 = \{c_1, c_5, c_3\} \tag{20}$$

$$L_1 = \{c_6, c_4, c_5\} \tag{21}$$

Because the sum of the vectors in (16), (17) and (18) are zero we know from (19) that $c_5 = c_2 + c_7$ and from (20) that $c_5 = c_1 + c_3$ and from (21) that $c_5 = c_4 + c_6$. So c_5 has three different repair sets of size 2. Now if we take one of these repair sets, lets take $c_5 = c_2 + c_7$ then $c_5 = 0 + 1 = 1$.

Because every vector in Figure 8.1 is connected with three lines. Every position, just like c_5, has three repair sets. Because of this you can always solve three erasures, both cooperative and sequential. However, there is a difference between the size of the repair sets; for sequential it is always $r_s = 2$, but cooperative has $r_c = 3$ or $r_c = 4$ depending on E. So the locality for sequential erasure repairing is two and the locality for cooperative erasure repairing is four.

8.2 The Advantage of Sequential Erasure Repairing

For the $[7, 3, 4]$-code we can always solve three erasures, both cooperative and sequential. This is because every position has three repair sets of length 2. Only four erasures will not always be solved.

Example 10. If the erasures are on the position $\{1, 5, 6, 7\}$, then every line in figure 8.1 that might be used to solve the erasures has two erasures on it instead of one.
These are the lines corresponding to

$$L_1 = \{c_1, c_7, c_4\} \tag{22}$$

$$L_2 = \{c_1, c_5, c_3\} \tag{23}$$

$$L_3 = \{c_1, c_2, c_6\} \tag{24}$$

It is possible to have four erasures, which you could solve sequential and not cooperative. This happens for the erasures $\{1, 2, 5, 6\}$. We can solve $c_1 = c_4 + c_7$ and $c_2 = c_3 + c_4$ and $c_6 = c_3 + c_7$, but we can not solve c_5 cooperative because the only way to solve c_5 with repair set of size 2 is with:
$c_5 = c_1 + c_3$, but c_1 was erased.
$c_5 = c_2 + c_7$, but c_2 was erased.
$c_5 = c_4 + c_6$, but c_6 was erased.
But if we want to solve this sequential, we could get our code back, because now we restore first c_1 and c_2 and c_6 and then we can solve c_5.

If you have four erasures and the remaining symbols are all on one line in figure 8.1, then you can not solve the erasures cooperative nor sequential.

8.3 $[7, 3, 4]$-Code without Limiting the Repair Set

When we limited r in Section 8.1 we unknowingly used the parity-check matrix. The parity-check matrix of the $[7, 3, 4]$-code is

$$H = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{25}$$

and the lines in Figure 8.1 are exactly the rows in (25) with $wt(\boldsymbol{h}) = 3$. The row space of (25) consists of 15 vectors. There are seven vectors with weight 3, seven vectors with weight 4 and one vector with weight 7. The extra row vectors do not influence the repair set for sequential, but they can decrease the size of the repair set for repairing 3 erasures cooperative. The row vectors of weight 4 are the following seven

$$\begin{aligned}
\boldsymbol{h_1} &= (1,1,1,0,0,0,1)\\
\boldsymbol{h_2} &= (1,1,0,1,1,0,0)\\
\boldsymbol{h_3} &= (1,0,1,1,0,1,0)\\
\boldsymbol{h_4} &= (0,1,1,0,1,1,0)\\
\boldsymbol{h_5} &= (0,0,1,1,1,0,1)\\
\boldsymbol{h_6} &= (0,1,0,1,0,1,1)\\
\boldsymbol{h_7} &= (1,0,0,0,1,1,1)
\end{aligned}$$

Example 11. If we have received a code word with $E = \{2,3,7\}$ then if we wanted to repair this cooperative with Figure 8.1 it would give us $R = \{1,4,5,6\}$. But if we use the row space of (25) we can solve it with $R = \{1,5,6\}$. we can solve c_2 with c_1 and c_6 (one of the lines), solve c_3 with c_1 and c_5 (also one of the lines) and we can solve c_7 with c_1, c_5 and c_6 (using $\boldsymbol{h_7}$).

Although we can restrict the locality of a code for one erasure, this is counterproductive if we want to repair multiple erasures cooperatively. For sequential erasure repairing the locality does not change.

9 Conclusion and Future Work

In this thesis, we have looked at the locality of repairing erasures cooperative and sequential. First we have explained how we can repair one erasure and then how we can repair multiple erasures using the sequential erasure repairing method and how to repair multiple erasures using the cooperative erasure repairing method. Then we have discussed a sequential locally repairable code, what length it has and its dimension for a code that can repair two erasures and for a code that can repair three erasures.

We have also discussed the Hamming code and that it can repair two erasures cooperative and we have looked at the extended Hamming code, which can repair three erasures cooperative. For the Hamming code and the extended Hamming code we have made a function to calculate the generalized Hamming weight for $\kappa \leq 3$. This was useful to calculate the locality for both codes.
Next we have compared the SLR-code for two erasures with the Hamming code and if they have the same length, then the Hamming code can store more information but the repair set for the SLR-code needs has a lower locality. We saw the same thing happen with the SLR-code for three erasures and the extended Hamming code.
Then we looked at the information rate of the codes and for two erasures we found two codes (one Hamming and one SLRC) with almost the same information rate. The same is true for three erasures. To get the same information rate we needed a large SLRC with a relatively small Hamming code and extended Hamming code.
When comparing the dimension of the codes we saw that the trade off between the SLRC and the Hamming code and the extended Hamming code is that the SLRC has a low locality with a high redundancy and the Hamming code and the extended Hamming code have a low redundancy and a high locality.
At the end we saw an example of were we restricted the locality for one erasure resulting in a higher locality for repairing multiple erasures cooperative.

For future work it could be interesting to look at the locality for shortened extended Hamming codes, so codes with distance 4 but shorter than your regular Hamming codes. Because we had a big list of SLRC codes but only a small list of Hamming codes and extended Hamming codes to compare them with. It would also be interesting to look into erasure repairing for product codes since we know you can repair erasures sequential but what about cooperative.

Bibliography

[1] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur and K. Ramchandran, "A Solution to the Network Challenges of Data Recovery in Erasure-coded Distributed Storage Systems: A Study on the Facebook Warehouse Cluster". URL:https://www.usenix.org/system/files/conference/hotstorage13/hotstorage13-rashmi.pdf

[2] S.B. Balaji and P. V. Kumar, "Erasure Codes for Distributed Storage: Tight Boundsand Matching Constructions," URL:https://arxiv.org/abs/1806.04474

[3] I. Núñez, E. Ortiz and A. Urdapilleta, "Generalized Hamming Weights for Linear Codes", URL: http://www.uprh.edu/ simu/Reports2001/NOU.pdf

[4] K. A. S. Abdel-Ghaffar, J. H. Weber, "Bounds for Cooperative Locality Using Generalized Hamming Weights", *IEEE Int. Sym. Inf. Theory (ISIT)* June 2017.

[5] W. Song, K. Cai, C. Yuen, K Cai and G. Han, "On Sequential Locally Repairable Codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 5, may 2018.

[6] A. S. Rawat, A. Mazumdar and S. Vishwanath, "Cooperative Local Repair in Distributed Storage," *EURASIP J. Adv. Signal Process* dec. 2015.

[7] J. Bom. "Cooperative Locality of Shortened Hamming Codes". *TU Delft, Applied Mathematics B.Sc. Thesis* June 2017.

[8] D. R. Hankerson, D. G. Hoffman, D. A. Leonard, C. C. Lindner, K. T. Phelps, C. A. Rodger, and J. R.Wall, *Coding Theory and Crypthography: The Essentials, Second Edition, Revised and Expanded*

[9] A. Wang and Z. Zhang, "Repair Locality with Multiple Erasure Tolerance," *IEEE Trans. Inf. Theory* vol. 60 no. 11, June 2013