Skew-Detecting and Skew-Tolerant Codes

- A Survey -

Diplomarbeit
Fakultät für Mathematik
Universität Bielefeld

vorgelegt von Beatrice Kienitz

Bielefeld, April 1998

Contents

Chapter 1

Introduction

1.1 Motivation and Background

Problem description

The main purpose of this thesis is to describe and analyse a special communication system called parallel asynchronous communication scheme. To model this system one sender and one receiver are connected by a channel consisting of n parallel subchannels. In this case by parallel I mean that each subchannel connects sender and receiver. The transmission of data is possible only in one direction, which is from sender to receiver; this means one studies communication without feedback.
Because of this restriction and the properties of the underlying channel a characteristic error pattern arises in this model. A major part of the analysis of this communication scheme is to describe this error pattern. The error is a kind of delay during transmission of the messages. This delay defines the error pattern named *skew* in a channel. This is the reason for that the receiver is not able to decode the messages which are sent without knowing special channel properties.
After analysing skew in this model one can construct certain codes that detect or even tolerate skew provided the channel's properties are not too bad. The meaning of 'too bad' will be specified in Chapter 3.

Goals

After a detailed description of the channel and the resulting error pattern conditions for skew detecting and skew tolerant codes are given. Once having found these conditions, which are distance properties between the code words, one is able to construct codes fulfilling the derived conditions. I will discuss two kinds of constructions. The first uses existing codes which already have the desired properties. The other one is straightforward.
If one wants to construct an 'efficient' code one has to assure that there is small redundancy, ease of encoding, membership test and decoding. Having this in mind, bounds and optimality issues for the described code constructions are examined.

Illustration

To get an idea of the model one can think of a number of n parallel tracks in a marble system. In this system transmission of a message is accomplished by rolling marbles along some of these tracks starting at the same time. Further, assume that the tracks differ in their lengths and their conditions, so that the amount of time it takes a marble to travel from sender to receiver is unknown and may even vary from track to track and from roll to roll. Thus, transmitting via this channel is *parallel asynchronous communication*. Notice that if for example two messages were sent, where for the second message one has used some of the shorter tracks, then this scheme allows for a certain number of marbles from the second batch to arrive before reception of the current message is complete.

Applications

As an example of this scheme, one can consider a board with several chips. The subchannels represent wires connecting the chips where differences in the lengths of the wires vary the propagation delay in the wires. This might result in asynchronous reception.
Since there is no communication between receiver and sender between messages, the sender does not wait for acknowledgement before sending the next message. This makes transmission faster, because the waiting period between the messages is shorter.
Potential applications are in on-chip, on-board and board to board communications enabling pipelined communication.

Relations

Relations to other problems of delay-insensitive data communication is presented in [1]. Verhoeff describes the possible models of parallel data transmission and proves the requirements for delay-insensitive codes.
There are two different approaches to solving this problem which is very common and arises in every system that incorporates transmission of information over parallel lines.
The first is based upon the idea of making the type synchronous by using time units. These units are known for both, the transmitter and the receiver, so that the receiver knows in which intervalls the messages were sent.
For the second, the asynchronous type, one possibility is to use a handshake mechanism. Namely, the transmitter sends the following message only after getting the acknowledgement that the current message has been completely received by the receiver. For this scheme it will be necessary for the receiver to know when reception of a message is complete, so he must be able to identify its end. Verhoeff has solved this problem by using a so-called *unordered code*, where the code words are unordered as vectors, as shown in Chapter 3. The disadvantage of this solution is that the channel is not fully utilized, because there is only one batch of marbles in the tracks at any time. The scheme considered in the following will make no use of the handshake mechanism, hence pipelined communication is possible.

Expression of thanks

First of all let me thank Prof. Ahlswede who led me to the study of skew codes. Furthermore I am grateful to B. Balkenhol and U. Tamm. They often spent their time for discussions and reading of the earlier versions of the thesis. As well I thank my best friends Regina and Jörg for everything.

1.2 Organization

The chapters

The chapters are organized as follows: Chapter 1 can be seen as a preparatory chapter. Notations and definitions of Coding Theory are shortly summarized. Basic codes especially those which are fault-tolerant are introduced. They are of use in Chapter 4 since all constructions presented there are based on

an error-correcting code.
Chapter 3 introduces the parallel asynchronous communication scheme. The definitions of skew and skew-detecting and skew-tolerant codes are given and explained. Their necessary and sufficient conditions are formulated in theorems and proved.
With these conditions one is able to construct codes which fulfill the properties needed for a skew code. In Chapter 4 three main methods to reach this are proposed. Two of them are based on known codes.
Bounds for the first method, Construction A, are given in Chapter 5.
In the last chapter optimality issues for Construction B and Construction C are presented.
All references are mentioned in the introductions of the chapters.

The appendix

In Appendix A the decoding algorithms for skew-detecting and skew-tolerant codes are formulated. They are needed to prove the theorems in Chapter 3.
In Appendix B the missing proofs of Chapter 5 are given.

The bibliography

The bibliography is ordered by subjects:

[1]-[2] refer to parallel communication and their codes,
[3]-[5] are literature about the theory of error-correcting codes,
[6]-[7] introduce the skew codes and their constructions,
[8]-[24] are an extract of the wide literature about t-EC/AUED codes, and
[25]-[34] are needed to present bounds of the codes.

Chapter 2

Basic Codes

This chapter can be seen as a preparation for the main part of the thesis, the study of the parallel asynchronous communication scheme. Some fundamental notions and definitions of Coding Theory which are needed below are shortly summarized in this chapter.
Especially in Section 2.2 I describe the idea of fault-tolerant codes. I clearly distinct between random, asymmetric and unidirectional errors. Then I briefly review the error correcting or detecting capabilities of binary block codes according to the different errors. This is mainly taken from [13]. These codes are of special interest in the model to be analyzed. The Hamming code will be introduced more precisely. This is required in view of the code constructions given in Chapter 4 which are based upon these codes. A much more detailed description can be found in [3], [4] or [5].

2.1 Definitions and Preliminaries

2.1.1 The Communication Scheme

In the context of this thesis there will always be two parties involved in the transmission of information, the sender of the messages and the receiver.The medium over which the messages are sent is called the channel.
The sender is composed of a source with *message alphabet* $\mathcal{A}$ and an encoder with *input alphabet* $\mathcal{X}$. The encoder transforms the messages of $\mathcal{A}$ into signals acceptable to the channel. The output signals from an *output alphabet* $\mathcal{Y}$ arrive at the decoder, who makes a decision concerning which message was

sent and delivers this message to the sink. The *sink alphabet* $\mathcal{B}$ may differ from $\mathcal{A}$. The decoder and the sink are associated with the receiver.
If one has no exact knowledge of the channel's properties, the input alphabet $\mathcal{X}$ of the channel is not necessarily the same as the output alphabet $\mathcal{Y}$. For our intention let the input and output symbols of a channel belong to the same alphabet $\mathcal{Q} = \{1, \dots q\}$, i. e. $|\mathcal{Q}| = q$.

A communication scheme in which no transmission from receiver to sender is possible is called communication *without feedback*.

Since at the moment one is not able to define errors which may occur in this scheme one can first mention that a channel is called *noisy* if during the transmission any kind of a possible changement of the message occurs. In Section 2.2 these errors will be described.

2.1.2 Types of Codes

Linear codes

Let n be fixed, then the set of $\mathcal{Q}$-ary n-tuples is denoted by $\mathcal{Q}^n$.

Definitions 2.1 A *q-ary block code $\mathcal{C}$ of length n* is any nonempty subset of $\mathcal{Q}^n$.

A *group code* $\mathcal{C}$ is a subgroup of $\mathcal{Q}^n$, where $\mathcal{Q}$ is a group.

If $|\mathcal{C}| = 1$ the code is called *trivial*.

The elements of $\mathcal{C}$ are called code words. One main criterium for a code is the number of messages, i. e. the source symbols, that can be encoded into code words of $\mathcal{C}$.
If the set of messages is $\mathcal{A} = \{1, \dots M\}$ one says that the *cardinality of* $\mathcal{C}$ is M. If the set of messages is $\mathcal{A} = \mathcal{Q}^k$ one says that $\mathcal{C}$ has *dimension k*.

The object of coding is to introduce redundancy so that even if some of the message bits are lost or corrupted, it will still be possible to recover the message at the receiver. Often the idea is to combine the information bits of a message with a certain number of extra bits so that each extra bit checks whether there is an error in some subset of the information bits. Therefore the dimension k is not necessarily equal to the length n of the code $\mathcal{C}$, because the redundant bits extend the length of the code.
The elements of $\mathcal{A}$ are the so-called *information vectors* or *message vectors*.

In the next section one will be able to see that to maximize error-protection one needs code words to have sufficient mutual distance. The number of coordinates, where two code words $\boldsymbol{x}$ and $\boldsymbol{y}$ differ, is called the *Hamming distance*. It is obviously a metric. If the Hamming distance is denoted by d_H, the *minimum distance* d of a non-trivial code $\mathcal{C}$ is the minimum Hamming distance between its code words:

$$d \quad \triangleq \quad \min_{\substack{x,y\in\mathcal{C} \\ x\neq y}} d_H(\boldsymbol{x}, \boldsymbol{y}). \tag{2.1}$$

Instead of stating that $\mathcal{C}$ is a code of length n, minimum distance d and cardinality M, one just says that $\mathcal{C}$ is a (n, M, d) code.

Linear block codes

If one assumes the alphabet $\mathcal{Q}$ and the code $\mathcal{C}$ to have some internal structure, one can construct and analyse codes much more easily than in the absence of such structure.
In the parallel asynchronous communication model which we want to analyze it is $\mathcal{Q} = \{0, 1\}$ as will be explained in chapter 3. As known $\mathcal{Q} = \{0, 1\}$ may be equipped with the structure of the *Galois field* $\mathbb{F}_2$. To generalize the following assume from now on that $\mathcal{Q}$ will always have the structure of the Galois field $\mathbb{F}_q$, where as a consequence $q = p^m$ for some prime p and $m \in \mathbb{N}$. Now let $\mathcal{Q} = \mathbb{F}_q$ one can associate the set of words $\mathcal{Q}^n$ with an n-dimensional vector space over $\mathbb{F}_q$. It will be denoted by $\mathcal{V}_n(q)$. For $\mathcal{V}_n(2)$ one shall simply write $\mathcal{V}_n$.

The *Hamming weight* $\omega(\boldsymbol{x})$ is the number of non-zero coordinates in $\boldsymbol{x}$. So

$$\omega(\boldsymbol{x}) = d_H(\boldsymbol{x}, \boldsymbol{0}),$$

where $\boldsymbol{0} = (0, \cdots, 0)$ denotes the all-zero vector of lenght n. Now that $\mathcal{Q}^n$ has the structure of the vector space $\mathcal{V}_n(q)$, one can define the most important general class of codes.

Definition 2.2 A *linear code* $\mathcal{C}$ *of length* n is any nonempty linear subspace of $\mathcal{V}_n(q)$.

If $\mathcal{C}$ has dimension k and minimum distance d one says that $\mathcal{C}$ is an $[n, k, d]$ code.

Note that a q-ary (n, M, d) code $\mathcal{C}$ has cardinality M, while a q-ary $[n, k, d]$ code $\mathcal{C}$ is linear and has cardinality q^k. The parameter d in the notations (n, M, d) or $[n, M, d]$ is sometimes omitted.
To determine the minimum distance of a code $\mathcal{C}$ with cardinality M and block length n one has to compute the distance between the $\binom{M}{2}$ pairs of code words. The next proposition shows the advantage of the structure of linear codes.

Proposition 2.3 The minimum distance of a linear code $\mathcal{C}$ is equal to the minimum non-zero weight in $\mathcal{C}$.

Proof: Since $\mathcal{C}$ is linear, with $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$ also $\boldsymbol{x} - \boldsymbol{y} \in \mathcal{C}$. The proposition now follows from the two observations:

$$\begin{aligned} d_H(\boldsymbol{x}, \boldsymbol{y}) &= d_H(\boldsymbol{x} - \boldsymbol{y}, \mathbf{0}) = \omega(\boldsymbol{x} - \boldsymbol{y}), \\ \omega(\boldsymbol{x}) &= d_H(\boldsymbol{x}, \mathbf{0}), \end{aligned}$$

which state that the distance between any two distinct code words is equal to the weight of some non-zero code words and vice versa. □

So, to determine the minimum distance of a linear code, one only has to find the lowest weight of all $M - 1$ non-zero code words.

The following corollary will especially be needed in Chapter 4 where for given k and d the length n of a code $\mathcal{C}$ has to be calculated. The proof can be found in [4].

Corollary 2.4 *The Griesmer bound*
The smallest value of n for which an $[n, k, d]$ code $\mathcal{C}$ exists satisfies the inequality

$$n \geq \sum_{i=0}^{k-1} \left\lceil \frac{d}{2^i} \right\rceil .$$

2.1.3 Description of Linear Codes

There are two standard ways of describing a k-dimensional linear subspace, i. e. a linear code $\mathcal{C}$: one by means of k independent basis vectors, the other uses $n - k$ linearly independent equations.
Remember that an $[n, k]$-code $\mathcal{C}$ is a linear subspace of $\mathcal{V}_n$ of dimension k.

The generator matrix

Definition 2.5 A *generator matrix* G for a linear code $\mathcal{C}$ is a $(k \times n)$-matrix for which the rows are a basis of $\mathcal{C}$. One says the rows of G generate $\mathcal{C}$.

Since the dimension of $\mathcal{C}$ is k, the number of rows of G, which equals the rank of G is k, because the rows must be linearly independent.
If G is a generator matrix for $\mathcal{C}$, then

$$\mathcal{C} = \{\boldsymbol{a}G : \boldsymbol{a} \in \mathcal{V}_k\}. \tag{2.2}$$

The code words $\boldsymbol{x} \in \mathcal{C}$ are linear combinations of the rows of G. If any two combinations were equal, there would be a dependence relation among rows of G. Thus, each distinct linear combination gives a distinct code word.
Also one could say that the code word $\boldsymbol{x} = \boldsymbol{a}G$ in (2.2) is the result of the encoding algorithm "muliplication by G" applied to the information vector $\boldsymbol{a} \in \mathcal{V}_k$.
One says that G is in *standard form* if $G = (I_k\, P)$, where I_k is the $(k \times k)$-identity matrix and $P \in \mathrm{M}(\mathrm{k} \times (\mathrm{n} - \mathrm{k}); \{0, 1\})$. If G is in standard form the first k coordinates of a code word $\boldsymbol{x} = \boldsymbol{a}G$ produce the information vector itself. For this reason these coordinates are called *information symbols*. The last $n-k$ coordinates are added to the k information symbols to make error-correction possible. The quantity $r = n - k$ is called the *redundancy* of the code.
Non-linear codes of cardinality q^k sometimes also have the quantity described above. This leads to the following

Definition 2.6 A *systematic code* $\mathcal{C}$ is a code of cardinality q^k which has the property that on the first k coordinates all possible (information) sequences occur.
Since one can separate information symbols and redundant symbols these codes are also called *separable*.

Notice, that a systematic code must not necessarily be linear and vice versa. But their algebraic structure shows the helpful relationship; one calls two codes $\mathcal{C}$ and $\mathcal{C}'$ *equivalent* - as far as error-correcting capability is concerned - if $\mathcal{C}'$ is obtained by applying a fixed permutation of the symbols to all the code words of $\mathcal{C}$. In other words, if $\mathcal{C}$ is a row space of a matrix G, then $\mathcal{C}'$ is the row space of a matrix G' that is obtained from G by rearranging columns. If one

matrix can be obtained from another by a combination of row operations and column permutations the two matrices are called *combinatorially equivalent.* Since the generator matrix of a linear code has full rank, it is quite obvious that any linear code is equivalent to a linear code that does have a generator matrix in standard form.
Also, the following proposition is clear.

Proposition 2.7 A linear code $\mathcal{C}$ is equivalent to a systematic code $\mathcal{C}'$.

The parity check matrix

Definition 2.8 A *parity check matrix* H of an $[n,k]$ code $\mathcal{C}$ is an $((n-k)\times n)$-matrix, satisfying

$$\boldsymbol{x} \in \mathcal{C} \Leftrightarrow H\boldsymbol{x}^T = 0^T. \tag{2.3}$$

In other words $\mathcal{C}$ is the null space of the $n-k$ linearly independent equations. Also one can describe H as follows: If $\mathcal{C}$ is a subspace of $\mathcal{V}_n$ of dimension k and generator matrix H, then H is the parity check matrix of the null space of $\mathcal{C}$ which is an $(n-k)$-dimensional subspace $\mathcal{C}^\perp$, $\mathcal{C}^\perp = \{\boldsymbol{a} \in \mathcal{V}_n : (\boldsymbol{a},\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in \mathcal{C}\}$. Both the vector space $\mathcal{C}$ and its null space $\mathcal{C}^\perp$ are linear subspaces of the space of all n-tuples over $\{0,1\}$, and therefore, both are linear codes. They are called *dual codes.* If $\mathcal{C}$ is an $[n,k]$ code, $\mathcal{C}^\perp$ is an $[n,n-k]$ code. If a code is a row space of a matrix, its dual is the null space, and vice versa.
In more detail one can describe it in the following way: A code word $\boldsymbol{x}$ is in $\mathcal{C}$ if and only if it is orthogonal to every row of H. If $\boldsymbol{x} = (x_1,\dots x_n)$ and an element in the ith row and the jth column of H is denoted h_{ij}, then (2.3) implies that for each i, $1 \le i \le n-k$, i. e. for each row of H

$$\sum_j x_j h_{ij} = 0. \tag{2.4}$$

Thus, (2.3) means that the components of $\boldsymbol{x}$ must satisfy a set of $n-k$ independent equations. These equations are called generalized *parity checks*, since in the binary case they are simply checks for even parity on certain sets of symbols in the code word. That is, for each row of H, the number of 1's in $\boldsymbol{x}$ that corresponds to 1's in that row of H is even if and only if $\boldsymbol{x}$ satisfies (2.4).

Equation (2.3) holds for every vector $\boldsymbol{x} \in \mathcal{C}$. In particular, it holds for the k basis vectors of the matrix G. These k equations can be expressed as follows, with 0 denoting the $k \times (n-k)$ all zero matrix:

$$GH^T = 0 \tag{2.5}$$

From (2.5) the following is quite obvious.

Proposition 2.9 If $\mathcal{C}$ is the row space of the matrix $G = (I_k P) \in M(k \times n, \{0,1\})$, where I_k is a $(k \times k)$ identity matrix and P is a $(k \times (n-k))$ matrix, then $\mathcal{C}$ is the null space of $H = (-P^T I_{n-k}) \in M((n-k) \times n, \{0,1\})$.

2.1.4 Extention of Codes

Definition 2.10 Let $\mathcal{C}$ be a q-ary (n, M, d) code. The *extended* code $\mathcal{C}^{ext}$ of $\mathcal{C}$ is defined by

$$\mathcal{C}^{ext} = \{(x_1, x_2, \ldots, x_n, -\sum_{i=1}^{n} x_i : \boldsymbol{x} \in \mathcal{C}\}.$$

So, the words in $\mathcal{C}^{ext}$ are obtained from those of $\mathcal{C}$ by adding an overall parity check symbol.

Clearly, in the binary case, all words in $\mathcal{C}^{ext}$ will have even weight. This shows that if $\mathcal{C}$ is a binary $(n, M, 2e+1)$ code, $\mathcal{C}^{ext}$ will be a binary $(n+1, M, 2e+2)$ code.

If G and H are generator, respectively parity check matrix of an $[n, k, d]$ code $\mathcal{C}$, the generator matrix G^{ext} and parity check matrix H^{ext} of the extended code $\mathcal{C}^{ext}$ are given by:

$$G^{ext} = \left(\begin{array}{c|c} & -\sum_{j=1}^{n} g_{1j} \\ & -\sum_{j=1}^{n} g_{2j} \\ G & \vdots \\ & -\sum_{j=1}^{n} g_{nj} \end{array} \right). \tag{2.6}$$

and

$$H^{ext} = \left(\begin{array}{cccc|c} 1 & 1 & \ldots & 1 & 1 \\ \hline & & & & 0 \\ & & H & & \vdots \\ & & & & 0 \end{array} \right). \tag{2.7}$$

2.2 Fault Tolerant Codes

From now on let the input and output alphabet be $\mathcal{Q} = \{0, 1\}$ and $\mathcal{C} \subseteq \mathcal{V}_n$. Assume that a communication scheme consists of a noisy channel, so that during transmission of a code word $\boldsymbol{x} = (x_1, \ldots x_n)$ errors may occur. Let $\boldsymbol{x}' \in \{0, 1\}^n$ be the received vector. Then $\boldsymbol{e} \triangleq \boldsymbol{x} - \boldsymbol{x}' \bmod 2$, $\boldsymbol{e} \in \{0, 1\}^n$ denotes the *error vector*. By counting the number of coordinates of $\boldsymbol{e}$ which are 1, one is able to conclude the number of errors which have occured. Since in this scheme transitions of kind $0 \to 1$ and $1 \to 0$ are possible, one has to distinguish between different errors. In the following definitions one will refer to the transition $0 \to 1$ as 0-error and the transition $1 \to 0$ as 1-error.

Definitions 2.11 i.) If both 0-errors and 1-errors appear in a received word with equal probability then the channel is called *binary symmetric channel* and the errors are called *symmetric* type or *random errors.*

ii.) In an ideal *asymmetric channel* only one type of an error, i. e. either 0-errors of 1-errors, can occur and the error kind is known a priori.

iii.) If both, 1-errors and 0-errors, can occur in the received words, but in any particular word all errors are of one type, then one says that errors of *unidirectional* type have occured.

For further observation of errors one will use the following notation. Let $\boldsymbol{x}, \boldsymbol{y} \in \{0, 1\}^n$, $\boldsymbol{x} = (x_1, \ldots, x_n)$, $\boldsymbol{y} = (y_1, \ldots, y_n)$. The *support of a vector* $\boldsymbol{x}$, in short supp $\boldsymbol{x}$, is defined by the set of all non-zero coordinate positions of $\boldsymbol{x}$, i. e. supp $\boldsymbol{x} = \{i \in \{1, \cdots, n\} : x_i = 1\}$. The vector $\boldsymbol{x}$ is said to be *covered by* $\boldsymbol{y}$ if for all $1 \leq i \leq n$ it holds $x_i = 1$ then $y_i = 1$, i. e. supp $\boldsymbol{x} \subseteq$ supp $\boldsymbol{y}$.

Remark 2.12 If $\boldsymbol{x}$ is the transmitted word and $\boldsymbol{x}'$ the received one then ii.) states that either $\boldsymbol{x}$ is covered by $\boldsymbol{x}'$ for all $\boldsymbol{x} \in \mathcal{C}$ or $\boldsymbol{x}'$ is covered by $\boldsymbol{x}$ for all $\boldsymbol{x} \in \mathcal{C}$.
But iii.) means that for all $\boldsymbol{x} \in \mathcal{C}$ either $\boldsymbol{x}$ is covered by $\boldsymbol{x}'$, or $\boldsymbol{x}'$ is covered by $\boldsymbol{x}$.

In view of the previous definitions there are different kinds of fault-tolerant and fault-detecting codes. In the next three subsections I will introduce three of them. One can give deterministic conditions for them depending on their correcting or detecting capabilities in form of distance properties between

their code words. Therefore, for each error one needs a suitable notion of a distance. The main results are summarized in the propositions.
The codes introduced below will be of use in Chapter 4. It will be shown that certain combinations of these codes fulfill the same conditions as skew codes which are analysed in this thesis.

2.2.1 Error-Correcting and Error-Detecting Codes

Obviously, a code $\mathcal{C}$ which is capable to correct up to t random errors is called *t-error-correcting code*, in short t-EC code.
The *error correcting capability* c of a code $\mathcal{C}$ is defined by

$$c = \left\lfloor \frac{d-1}{2} \right\rfloor, \text{where } d \text{ is the minimum distance of } \mathcal{C}. \tag{2.8}$$

The reason for calling c error-correcting capability is quite obvious. If d is the minimum distance of a code $\mathcal{C}$ and if during the transmission of the code word $\boldsymbol{x}$ over the channel c errors at most have been made then the received word $\boldsymbol{x}'$ will still be closer to $\boldsymbol{x}$ than to any other codeword $\boldsymbol{y} \in \mathcal{C}$. So a maximum likelihood decoding algorithm applied to $\boldsymbol{x}'$ will result in $\boldsymbol{x}$.
However, codes can also be used for *error-detection*. Let f be some integer, $f \leq c$. Then it follows from the definition of d that no code word in $\mathcal{Q}^n$ can be at a distance at most f from one code word, while at the same time being at distance up to $d - f - 1$ from some other code word. This means that one can correct up to f errors, but also detect them if more than f errors have occured, as long as not more than $d - f - 1$ errors have occured.
Let $d \triangleq d(\mathcal{C})$ denote the minimum Hamming distance d of $\mathcal{C}$, then summarizingly one gets the following:

Proposition 2.13 A code $\mathcal{C}$ is capable of detecting e errors, $e \leq t$ if and only if $d \geq t + 1$.

Proposition 2.14 A code $\mathcal{C}$ is capable of correcting $\tilde{e}$ errors, $\tilde{e} \leq t$ if and only if $d \geq 2t + 1$.

Proposition 2.15 A code $\mathcal{C}$ is capable of correcting $\tilde{e}$ errors, $\tilde{e} \leq t$ and detecting up to e errors, $e \geq t$ if and only if $d \geq t + e + 1$.

The Hamming Code

Equation (2.5) can be interpreted as follows: a word $\boldsymbol{x}$ is a code word in a linear code $\mathcal{C}$ if and only if the coordinates of $\boldsymbol{x}$ give a dependency relation between the columns of the parity check matrix H of $\mathcal{C}$.
In particular, the minimum distance d of a linear code will satisfy $d \geq 2$ if and only if H does not contain the all-zero vector. Similary, $d \geq 3$ if and only if H does not contain two columns that are linearly dependent. In general, $\mathcal{C}$ will have minimum distance $d \geq w$ if and only if each $(w-1)$-tuple of columns is linearly independent. If at least one d-tuple of columns is linearly dependent, the minimum distance will be exactly equal to w.
In view of the above, one knows that the length of a q-ary $[n, k, 3]$ code is upper bounded by the maximum number of pairwise linearly independent vectors in $\mathcal{V}_r(q)$, where $r = n - k$ is the redundancy of $\mathcal{C}$. Now $\mathcal{V}_r(q)$ has $q^r - 1$ non-zero vectors. They can be devided into $(q^r - 1)/(q - 1)$ groups of size $q - 1$, each consisting of a non-zero vector in $\mathcal{V}_r(q)$ together with all its non-zero scalar multiples. The extreme case that $n = (q^r - 1)/(q - 1)$ leads to the following

Definition 2.16 The q-ary *Hamming code* of length $n = (q^r - 1)/(q - 1)$ and redundancy r is defined by the parity check matrix that consists of (a maximum set of) columns that are all pairwise linearly independent. It is denoted by $\mathcal{H}_r(q)$. Its minimum distance is equal to 3.

The binary Hamming code is most easily described in terms of its parity check matrix. Consider a matrix $H \in M(m \times (2^m - 1), \{0, 1\})$, the column vectors being all possible m-tuples except the zero m-tuple. Over $\mathbb{F}_2$ if two vectors add to zero, they must be equal. Therefore, no two columns of this matrix add to zero, and, since the only nonzero scalar is 1, no linear combination of two columns is zero. Therefore, the null space of H has minimum weight 3 and is capable of correcting all single errors. The code words have length $2^m - 1$, with m parity check symbols and therefore $2^m - 1 - m$ information symbols.

Definition 2.17 Let $\mathcal{C}$ be an $[n, k, d]$ code with parity check matrix H. The *syndrome* $\boldsymbol{s}$ of a vector $\boldsymbol{x}$ in $\mathcal{V}_n$ is the vector in $\mathcal{V}_{n-k}$ defined by $\boldsymbol{s} = H\boldsymbol{x}^T$.

Note that the syndrome $\boldsymbol{s}$ is a column vector, while the vectors related to vectors transmitted or received are row vectors.

If the syndrome of a received word $\boldsymbol{x}'$ is 0, then $\boldsymbol{x}$ is a code word and eihter no error has occured during the transmission or it is impossible to realise the error, because the number of errors which can be tolerated is exceeded. In general, if the code word $\boldsymbol{x}$ has been transmitted, and the word $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{e}$ has been received, where $\boldsymbol{e}$ is the error vector, one has

$$H\boldsymbol{x}'^T = H(\boldsymbol{x} + \boldsymbol{e})^T = H\boldsymbol{x}^T + H\boldsymbol{e}^T = H\boldsymbol{e}^T.$$

So, the syndrome of $\boldsymbol{x}'$ is completely determined by that of $\boldsymbol{e}$. If a code word $\boldsymbol{x}$ is transmitted and a single error occurs, the error vector $\boldsymbol{e}$ is a vector that has a single 1 in the position corresponding to the error. The syndrome $\boldsymbol{e}H^T$ is just the row of H^T corresponding to the error. Thus by comparing the syndrome with the matrix H^T, the position of the error can be found, and simply changing the symbol corrects the error.

2.2.2 Asymmetric Error-Correcting Codes

Since asymmetric and unidirectional errors are crossovers of type $0 \to 1$ or $1 \to 0$ it will be useful to define distances between the words of a code $\mathcal{C}$ by the number of coordinate positions in which the code words differ. For two code words $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C} \subseteq \{0,1\}^n$, $N(\boldsymbol{x}, \boldsymbol{y})$ denotes the number of coordinates in which $\boldsymbol{x}$ is 1 and $\boldsymbol{y}$ is 0, i. e. $N(\boldsymbol{x}, \boldsymbol{y}) = |\{i \in \{1, \cdots, n\} : x_i = 1 \text{ and } y_i = 0\}|$. It follows that $N(\boldsymbol{x}, \boldsymbol{y}) + N(\boldsymbol{y}, \boldsymbol{x}) = d_H(\boldsymbol{x}, \boldsymbol{y})$.
Rao and Chawla [9] and Anderson [8] have defined the *asymmetric distance* d_a with reference to the asymmetric error-correcting capabilities of binary block codes as follows: The asymmetric distance $d_a(\boldsymbol{x}, \boldsymbol{y})$ between two code words $\boldsymbol{x}, \boldsymbol{y}$ is defined by $d_a(\boldsymbol{x}, \boldsymbol{y}) \triangleq \max\{N(\boldsymbol{x}, \boldsymbol{y}), N(\boldsymbol{y}, \boldsymbol{x})\}$. In [9] it is shown that the asymmetric distance is a metric. Then the *asymmetric minimum distance* Δ of a non-trivial binary code $\mathcal{C}$ is the minimum asymmetric distance between its code words:

$$\Delta \triangleq \min_{\substack{x,y \in \mathcal{C} \\ x \neq y}} d_a(\boldsymbol{x}, \boldsymbol{y}). \tag{2.9}$$

Notice, that $d_a(\boldsymbol{x}, \boldsymbol{y}) = d_H(\boldsymbol{x}, \boldsymbol{y})$ exactly if $N(\boldsymbol{x}, \boldsymbol{y}) = 0$ or $N(\boldsymbol{y}, \boldsymbol{x}) = 0$. This means that either $\boldsymbol{x} \subseteq \boldsymbol{y}$ or $\boldsymbol{y} \subseteq \boldsymbol{x}$. If $\boldsymbol{x}, \boldsymbol{y}$ have the property that $\boldsymbol{x} \subseteq \boldsymbol{y}$ or $\boldsymbol{y} \subseteq \boldsymbol{x}$ then they are called an *ordered pair*; otherwise they are *unordered*.

As in the previous section, let $\Delta \triangleq \Delta(\mathcal{C})$ denote the minimum asymmetric distance of the code $\mathcal{C}$.

Proposition 2.18 A code $\mathcal{C}$ can correct any a asymmetric errors in a code word if and only if $\Delta \geq a + 1$.

Remark 2.19 By definition of d_H and d_a one obtains that $d_H(\boldsymbol{x}, \boldsymbol{y}) = b$ implies $d_a(\boldsymbol{x}, \boldsymbol{y}) \geq \lfloor \frac{b+1}{2} \rfloor$. Therefore, if $d \geq 2t + 1$ then $\Delta^{\mathcal{C}} \geq t + 1$.

2.2.3 Unidirectional Error-Correcting/Detecting Codes

The following propositions are proved in [13] for example.

Proposition 2.20 A code $\mathcal{C}$ is capable of detecting all unidirectional errors if and only if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$ it holds $\min\{N(\boldsymbol{x}, \boldsymbol{y}), N(\boldsymbol{y}, \boldsymbol{x})\} \geq 1$, i. e. $\boldsymbol{x}, \boldsymbol{y}$ are an unordered pair of code words.

Proposition 2.21 A code $\mathcal{C}$ is capable of correcting u unidirectional errors, $u \leq t$ if and only if the following condition holds: for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$

$$d_a(\boldsymbol{x}, \boldsymbol{y}) \geq \begin{cases} 2t + 1 & \text{for} \quad \boldsymbol{x}, \boldsymbol{y} \in \mathcal{C} \text{ an ordered pair} \\ t + 1 & \text{for} \quad \boldsymbol{x}, \boldsymbol{y} \in \mathcal{C} \text{ an unordered pair} \end{cases}.$$

From the previous fact one can define the *unidirectional distance* d_u of a code $\mathcal{C}$ as follows

$$d_u(\boldsymbol{x}, \boldsymbol{y}) \quad \triangleq \quad \begin{cases} d_a(\boldsymbol{x}, \boldsymbol{y}) = d_H(\boldsymbol{x}, \boldsymbol{y}) & \text{if} \quad \boldsymbol{x}, \boldsymbol{y} \in \mathcal{C} \text{ an ordered pair} \\ 2d_a(\boldsymbol{x}, \boldsymbol{y}) - 1 & \text{if} \quad \boldsymbol{x}, \boldsymbol{y} \in \mathcal{C} \text{ an unordered pair} \end{cases} \tag{2.10}$$

Then the *minimum unidirectional distance* δ of a non-trivial code $\mathcal{C}$ is defined as usual by:

$$\delta \quad \triangleq \quad \min_{\substack{x, y \in \mathcal{C} \\ x \neq y}} d_u(\boldsymbol{x}, \boldsymbol{y}). \tag{2.11}$$

Let $\delta \triangleq \delta(\mathcal{C})$ denote the minimum unidirectional distance of a code $\mathcal{C}$. Then Proposition 2.21 can be formulated as:

Proposition 2.22 A code $\mathcal{C}$ is capable of correcting u unidirectional errors, $u \leq t$ if and only if $\delta \geq 2t + 1$.

Chapter 3

Skew Codes

The aim of the thesis is to study parallel asynchronous communication without feedback. This chapter starts with its description and leads to the analysis of skew which is a special kind of an error during the transmission of messages. Skew-detecting and skew-tolerant codes have been developed to prevent erroneous decoding. The problem is to find deterministic conditions for these codes. It is therefore obvious that the occurence of skew must not be arbitrary. To define criteria for these codes an upper bound of possible delay in the channel is needed. Later one will see that skew depends on two fixed values t_1 and t_2 which cannot be exceeded. With this bound it is possible to determine necessary and sufficient conditions for codes which are able either to detect and stop the further transmission or to tolerate the occurence of skew and allow for continuous operation. These conditions are given by global distance properties between the code words of these codes.
First of all I define the communication scheme in Section 3.1. For further observation I introduce the notations to be used. In the following some definitions related to skew are given. The most important ones are the definitions of skew-detecting and skew-tolerant codes. The notations I adopted have been slightly modified from Blaum and Bruck,[6]. Also I have included the remarks to illustrate the definitions. For the same reason I discussed Example 3.8, also taken from [6], in more detail.
The main part of this chapter is Section 3.3. Deterministic conditions for skew codes are given and discussed. The proofs are based upon Decoding Algorithms which can be found in Appendix A. However, to my knowledge there is false reasoning in the decoding strategy for skew-tolerant codes given in [6]. I justify this opinion and propose a corrected version.

3.1 The Parallel Asynchronous Communication Scheme

The Model

This communication scheme consists of one sender and one receiver connected by n parallel asynchronous subchannels each starting at the sender and ending at the receiver. The n subchannels, labeled by numbers $1, \cdots n$, will in short be denoted as the channel.
The input alphabet of the channel is $\mathcal{Q} = \{0, 1\}$. Since the channel consists of n subchannels the code words to be conveyed via the channel are binary vectors of length n. Therefore, each subchannel represents one coordinate of the vector to be transmitted.
However, for further analysis of this scheme it will be helpful to regard the code words of a code $\mathcal{C}$ as subsets of $[n] \triangleq \{1, \ldots, n\}$ instead of binary vectors of length n and vice versa. This is possible by the isomorphism i which maps a vector on its support, i. e. $i : \{0,1\}^n \longrightarrow \mathcal{P}([n])$.
The isomorphism is formally given by the following.
Let $\mathcal{A} \subseteq \mathcal{P}([n])$ and

$$\begin{aligned} \mathcal{B} \triangleq \{x^n = (x_1, \ldots, x_n) \in \{0,1\}^n : & \text{ for all } \boldsymbol{x} \text{ there exits } X \in \mathcal{A} \\ & \text{ with } x_j = 1 \text{ for all } j \in X\} \end{aligned}$$

then $\mathcal{A} \hat{=} \mathcal{B}$.
Due to the isomorphism a code word of a code $\mathcal{C}$ will be denoted by $\boldsymbol{x} \in \mathcal{C}$, regardless whether it is a set or a vector, which will be obvious in the context. The reason for using both terms simultaneously will be explained next. The channel consists of n parallel asynchronous subchannels. Assume one wants to transmit a binary vector of length n. Each subchannel transmits simultaneously and can carry one coordinate of $\boldsymbol{x}$. In this model an electrical transition corresponds to a 1, while absence of a transition corresponds to a 0. Clearly, this means that the channel can only transmit the 1's of a vector and there is *no* symbol which assigns that a 0 was transmitted. The receiver notices transitions in those subchannels j, $j \in [n]$, where $x_j = 1$. Therefore, the receiver gets the support of the transmitted vector $\boldsymbol{x}$. Consequently, both the input and the output alphabet of the channel, can be seen as $\mathcal{A}$=[n].

The Error Pattern

First of all notice that of course each subchannel transmits simultaneously. Considering, however, that the subchannels are asynchronous the transitions of each subchannel may not arrive at the same time. The transmission time may differ in each subchannel and may even vary in each subchannel itself. This is the reason why the communication scheme allows for a certain number of transitions from a second message to arrive before the reception of the current message has been completed.
For example, let $\boldsymbol{x}$ be the code word sent first, followed by $\boldsymbol{y}$ and other possible code words. Then it may be possible that the receiver observes a transition in a subchannel $i \in \boldsymbol{y} - \boldsymbol{x}$, $i \in [n]$ before he has noticed transitions in all the subchannels belonging to the support of $\boldsymbol{x}$. This condition is called skew. The precise definition of skew is given in the next section.
Also it will be explained in the following that as a consequence of skew in general the receiver is not capable to decode the first sent codeword $\boldsymbol{x}$.

3.2 Definitions and Preliminaries

3.2.1 Notations

The most important property of the communication scheme described above is that the subchannels are asynchronous. This means that even if the subchannels transmit simultaneously the transitions may not arrive at the same time.
In the model with feedback this property can easily be handled. The sender transmits the following message only after he has got acknowledgement that the receiver has received the complete current message. From the coding point of view the main problem is to determine the end of a message. This model was first studied by Verhoeff [1]. He found out that the receiver is able to determine the end of a message by using a so-called *unordered code* $\mathcal{C}$, i. e. if $\mathcal{C} \in \mathcal{P}([n])$ is unordered, then $(\mathcal{C}, \subseteq)$ is an antichain; more precisely, an *antichain* is a set $\mathcal{X}$ on which a reflexive, antisymmetric and transitive relation $\preccurlyeq$ is defined ($(\mathcal{X}, \preccurlyeq)$ is called *partially ordered set*), but no two elements are comparable.
In our context this means that no code word $\boldsymbol{x}$ is a subset of another code word $\boldsymbol{y}$, for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}, \boldsymbol{x} \neq \boldsymbol{y}$. Then $\mathcal{C}$ is an antichain.

In the following we consider only the communication model without feedback. Assume that the vector $\boldsymbol{x}$ is transmitted, followed by other vectors. The receiver observes the transitions which arrive in the subchannels $1, \ldots, n$. After the m-th transition has arrived, the receiver gets the sequence
$\hat{Z}_m \triangleq z_1, \ldots, z_m = (z_i)_{i=1}^m$, $z_i \in [n]$, $i = 1, \ldots, m$,
where z_i represents the number of the channel in which the i-th transition has arrived, i. e. the i-th transition has been received in channel z_i.
If $\hat{Z}_m$ denotes a sequence as defined above, one denotes $Z_m \triangleq \mathrm{supp}\hat{Z}_m$ which is the support of the set containing all elements of the sequence $\hat{Z}$.
Furthermore, $\hat{Z} \triangleq (z_i)_{i \in \mathbb{N}}$ and $Z \triangleq \mathrm{supp}\hat{Z}$.
In the case where the subchannels are synchronous all transitions of a message arrive at the same time. This means that the received sequence $\hat{Z}_m$ starts with the support of the vector transmitted first, followed by the support of the second etc. For example, assume that the message $\boldsymbol{x} = \{x_1, \ldots, x_k\}$, $x_i \in [n]$, $1 \leq i \leq k$, is transmitted, followed by other messages. Then the receiver obtains a sequence $\hat{Z} = z_1, \ldots, z_i, \ldots$, where the first k transitions arrive in the channels $z_i \in \boldsymbol{x}$, for all $1 \leq i \leq k$, no matter in what order they arrive.
However, this is not the case when the channel consists of asynchronous subchannels. In addition one has to take into consideration that the amount of time it takes to convey a coordinate of the information vector from the sender to the receiver can vary in each subchannel itself.
Now one can imagine that it is possible that transitions of a second message may arrive before reception of the current vector is complete.
In order to precisely define skew one first has to introduce the following notations. At the moment one uses the notion of skew as a kind of delay as described above.
Let $\boldsymbol{x} \in \mathcal{C}$ be a code word and $\hat{Z}$ the received sequence. Define

$$m(\boldsymbol{x}; \hat{Z}) \triangleq \min\{j \in \mathbb{N}_0 : (Z_j \subseteq \boldsymbol{x}) \wedge (z_{j+1} \notin \boldsymbol{x} \vee z_{j+1} \in Z_j)\}. \tag{3.1}$$

Thus, $m(\boldsymbol{x}; \hat{Z})$ denotes the index of the last transition in $\boldsymbol{x}$ before the occurence of skew, i.e. the last transition of $\boldsymbol{x}$ before the arrival of either a transition not in $\boldsymbol{x}$ or a second arrival in one of the subchannels $1, \ldots n$. Furthermore, define

$$r(\boldsymbol{x}; \hat{Z}) \triangleq \min\{j \in \mathbb{N}_0 : \boldsymbol{x} \subseteq Z_j\}; \tag{3.2}$$

$r(\boldsymbol{x}; \hat{Z})$ denotes the index of the last arrival in $\boldsymbol{x}$.

Remarks 3.1 a.) If there is no skew, $m(\boldsymbol{x}; \hat{Z}) = r(\boldsymbol{x}; \hat{Z})$.
b.) If $\boldsymbol{x}$ is sent first and $\hat{Z} = z_1, \ldots,$ where $z_1 \notin \boldsymbol{x}$, then $m(\boldsymbol{x}; \hat{Z}) = 0$.

3.2.2 Definitions

A Bound of Skew

For further analysis it will be necessary to have fixed real numbers which define the occurence of skew between a code word $\boldsymbol{x}$ and all possible outcomes at the receiving end. If one knows these numbers for all code words of a code $\mathcal{C}$, one gets an upper bound for the skew, so that the channel is fully characterized.

Definition 3.2 Let $\boldsymbol{x} \in \mathcal{C}$ be a code word, $\hat{Z} = z_1, \ldots, z_i, \ldots$ the received sequence, $\hat{Z}_i = z_1, \ldots, z_i$, $Z_i = \text{supp}\hat{Z}_i$, where $z_t \in [n]$, t $\in \mathbb{N}$,
$m \triangleq m(\boldsymbol{x}; \hat{Z})$, $r \triangleq r(\boldsymbol{x}; \hat{Z})$.
The skew of $\boldsymbol{x}$ *with respect to* $\hat{Z}$ *is equal to* $(l_1, l_2) = (l_1(\boldsymbol{x}), l_2(\boldsymbol{x}))$, short $\mathcal{S}(\boldsymbol{x}; \hat{Z}) \triangleq (l_1, l_2))$,
if and only if
$l_1 = |(Z_r - Z_m) \cap \boldsymbol{x}|$ and $l_2 = r - m - l_1$.

In other words, it is helpful to describe critera of skew by counting the number of elements of a sent code word $\boldsymbol{x}$ in the part of the sequence $\hat{Z}$ where there is a delay in the transmission. So, l_1 gives the number of elements in $\boldsymbol{x}$ which are in the so-called *delay part* of the sequence. This part begins with the first component z_i not being element of $\boldsymbol{x}$ or being a repeated arrival, i.e. at least a second transition in a subchannel $z_i \in \boldsymbol{x}$.
Otherwise, it is easy to see that l_2 is the number of errors until the complete transmission of all elements of $\boldsymbol{x}$.

Remarks 3.3 a.) Let $\boldsymbol{x} = \{x_1, \ldots, x_k\}$, so that $|\boldsymbol{x}| = k$, then $k = m + l_1$, where m is defined in (3.1). Note that the repeated arrivals, which can be seen as elements of $\boldsymbol{x}$, are not counted by l_1.
b.) Let $\boldsymbol{x} \in \mathcal{C}$ be the code word transmitted first and $\hat{Z}$ the received sequence with $\mathcal{S}(\boldsymbol{x}; \hat{Z}) = (l_1, l_2)$, $m = m(\boldsymbol{x}; \hat{Z})$, $r = r(\boldsymbol{x}; \hat{Z})$. Then the first m elements of $\hat{Z}$ are elements of $\boldsymbol{x}$. At the $m + 1$th item, the delay part of the sequence containing $l_1 + l_2$ elements begins with $z_{m+1} \notin \boldsymbol{x}$ or $z_{m+1} \in Z_m$.

In the following, an *error* between $\boldsymbol{x}$ and the obtained sequence denotes either an arrival in a channel not contained in $\boldsymbol{x}$ or a repeated arrival. With this definition one is able to give an upper bound of skew in a certain communication channel.

Definition 3.4 *The skew of an n-parallel asynchronous channel* (n-PAC) *does not exceed* (t_1, t_2), denoted as

$$\overline{\mathcal{S}} \quad \triangleq \quad \max_{x \in \{0,1\}^n} \mathcal{S}(\boldsymbol{x}; \hat{Z}) = \max_{x \in \{0,1\}^n} (l_1(\boldsymbol{x}), l_2(\boldsymbol{x})) = (\overline{l_1}, \overline{l_2}) \leq (t_1, t_2),$$

if $\overline{l_1} \leq t_1$ and $\overline{l_2} \leq t_2$.

Remarks 3.5 a.) The condition $\overline{\mathcal{S}} = (\overline{l_1}, \overline{l_2}) \leq (t_1, t_2)$ is actually too strong, because it suffices to take the maximum for all elements of the code $\mathcal{C}$ which is a subset of $\{0,1\}^n$. But for the formulation of the deterministic conditions of skew codes one needs to have the fixed values of t_1 and t_2. Then one chooses a code $\mathcal{C}$ which fullfils the conditions.

b.) Let $\boldsymbol{x} \in \mathcal{C}$ be the code word transmitted first, $|\boldsymbol{x}| = k$ and $\mathcal{S}(\boldsymbol{x}; \hat{Z}) \leq (t_1, t_2)$ hold. Then, $\hat{Z}$ can not be arbitrary. If s is given by $s \triangleq \max\{0, k - t_1\}$, then $m(\boldsymbol{x}; \hat{Z}) \geq s$, i.e. before the occurence of skew at least the first s transitions must arrive in channels belonging to the set $\boldsymbol{x}$.
On the other hand, the number of errors until the complete transmission of $\boldsymbol{x}$ must not exceed t_2, therefore $r - k \leq t_2$. In other words, if one has a communication channel and a code $\mathcal{C}$ with $\mathcal{S}(\boldsymbol{x}; \hat{Z}) \leq (t_1, t_2)$ for all $\boldsymbol{x} \in \mathcal{C}$, then t_1 is the maximal number of elements of $\boldsymbol{x}$ lying in a delay part of any received sequence $\hat{Z}$. Accordingly, t_2 is the maximal number of errors during transmission of a code word $\boldsymbol{x}$.

c.) Notice that fundamentally one has to distinguish if the condition $\mathcal{S}(\boldsymbol{x}, \hat{Z}) \leq (t_1, t_2)$ in Definition 3.4 is meant locally or globally in the sequence $\hat{Z}$. This means that if the condition is only local then it holds for the sequence $\hat{Z}$ which changes after each decoding, because after the decoder has found a code word $\boldsymbol{x}$, the transitions of $\boldsymbol{x}$ will be deleted in the sequence, now denoted $\hat{Z}'$ for instance. Then the condition $\mathcal{S}(\boldsymbol{x}, \hat{Z}') \leq (t_1, t_2)$ for all $\boldsymbol{x} \in \mathcal{C}$ and possible sequence $\hat{Z}'$ holds anew. This difference becomes clear in connection with the decoding algorithm in the next section.

Definition 3.6 Let t_1 and t_2 be non-negative integers. The code $\mathcal{C}$ is called (t_1, t_2)-*skew-detecting* (SD) if, whenever a code word $\boldsymbol{x} \in \mathcal{C}$ is transmitted followed by other code words giving a received sequence $\hat{Z}$, then by examining $\hat{Z}$, the code will
(i) correctly decode $\boldsymbol{x}$ provided that $\mathcal{S}(\boldsymbol{x}; \hat{Z}) = (0, 0)$ (i.e. no skew) and
(ii) detect the occurence of skew if there is a repeated arrival as long as $(0, 0) \leq \mathcal{S}(\boldsymbol{x}; \hat{Z}) \leq (t_1, t_2)$ (i.e. the skew does not exceed (t_1, t_2)).

The definition states that in the worst situation, the detection of skew will occur after n arrivals.

Definition 3.7 Let t_1 and t_2 be nonnegative integers.
The code $\mathcal{C}$ is called (t_1, t_2)- *skew-tolerant* (ST) if, whenever a code word $\boldsymbol{x} \in \mathcal{C}$ is transmitted followed by other code words, and $\hat{Z}$ is the received sequence, then by examining $\hat{Z}$, the code will correctly decode $\boldsymbol{x}$ as long as $\mathcal{S}(X; \hat{Z}) \leq (t_1, t_2)$ (i.e. the skew does not exceed (t_1, t_2)).

Example 3.8 Consider a code $\mathcal{C}$ for a 5-PAC containing two code words, $\mathcal{C} = \{\boldsymbol{x}, \boldsymbol{y}\}$, where $\boldsymbol{x} = (10000)$, i.e. supp $\boldsymbol{x} = \{1\}$ and $\boldsymbol{y} = (01111)$, i.e. supp $\boldsymbol{y} = \{2, 3, 4, 5\}$. In particular $\boldsymbol{x} \cap \boldsymbol{y} = \emptyset$.
The claims are that the code $\mathcal{C}$ is

a.) (3, 3)-SD and

b.) (1, 2)-ST.

The decoder checks for the arrivals, and when either $\boldsymbol{x}$ or $\boldsymbol{y}$ arrive they are decoded and the process is restarted. For each arrival z_j, the receiver checks if the received sequence Z_j coincides with either $\boldsymbol{x}$ or $\boldsymbol{y}$. To establish a.) and b.) consider the two cases when (i) $\boldsymbol{x}$ is transmitted first, followed by $\boldsymbol{y}$ and then other transmissions and (ii) $\boldsymbol{y}$ is transmitted first followed by $\boldsymbol{x}$ and other transmission giving respectively the received sequence $\hat{Z}$.

Proof of a.) (i) If the first transition arrives in channel 1, then $\mathcal{S}(\boldsymbol{x}; \hat{Z}) = (0, 0)$ and $\boldsymbol{x}$ can be decoded correctly.
If the first transition does not arrive in this channel, then $m(\boldsymbol{x}; \hat{Z}) = 0$. Since $\mathcal{S}(\boldsymbol{x}; \hat{Z}) \leq (3, 3)$ the maximal number of errors till the end of the complete transmission of $\boldsymbol{x}$ must not exceed 3, so that there are 3 transitions at most arriving in the channels unequal channel number 1 until there is a transition in channel number 1 which forms the code word $\boldsymbol{x}$. However, this means that

one can not erroneously decode $\boldsymbol{y}$, because $|\text{supp } \boldsymbol{y}| = 4$. In the worst case the decoder has not yet decoded a single code word at the sixth arrival, and he notices repeated arrival. As a result of this he stops further transmission.
(ii) Here it cannot happen either that $\boldsymbol{x}$ will be decoded erroneously at first, because this would imply that the first transition arrives in channel 1. This, however, cannot be the case, since $m(\boldsymbol{y}; \hat{Z}) \geq s = \max\{0, 4-3\} = 1$. Therefore, the first transition must arrive in one of the subchannels contained in $\boldsymbol{y}$. So, if skew occurs during transmission of $\boldsymbol{x}$, it will not be possible to decode any code word of $\mathcal{C}$. The decoder stops when he notices repeated arrival.

Proof of b.) (i) If $\mathcal{S}(\boldsymbol{x}; \hat{Z}) \leq (1, 2)$ one of the first three transitions arrives in channel 1, because the maximal number of errors during transmission of $\boldsymbol{y}$ must not exceed 2.
(ii) Since $t_1 = 1$, $m(\boldsymbol{y}; \hat{Z}) \geq s = 4 - 1 = 3$ the first three transitions must arrive in channels belonging to $\boldsymbol{y}$. Therefore, the receiver has a clear decoding strategy: he observes the arrival of the first three transitions of $\hat{Z}$.
If $j \in \{1, 2, 3\}$ with $z_j \in \boldsymbol{x}$ exists then he decodes $\boldsymbol{x}$.
If for all $j \in \{1, 2, 3\}$ $z_j \notin \boldsymbol{x}$ then he decodes $\boldsymbol{y}$.

3.3 Necessary and Sufficient Conditions

3.3.1 (t_1, t_2)-Skew-Detecting Codes

In this section, (t_1, t_2)-SD codes will be studied. By giving characterizations for such codes in terms of distance properties between their code words one can prove that these conditions are not only sufficient but even necessary.

Theorem 3.9 For $t_1, t_2 \in \mathbb{N}_0$ let $t \triangleq \min\{t_1, t_2\}$ and $T \triangleq \max\{t_1, t_2\}$.
The code $\mathcal{C}$ is (t_1, t_2)-skew-detecting, if and only if
for any two different code words $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$ at least one of the following two conditions occurs:

$$a.) \quad \min \quad \{N(\boldsymbol{x}, \boldsymbol{y}), N(\boldsymbol{y}, \boldsymbol{x})\} \geq t + 1 \tag{3.3}$$

or

$$b.) \quad \min \quad \{N(\boldsymbol{x}, \boldsymbol{y}), N(\boldsymbol{y}, \boldsymbol{x})\} \geq 1 \text{ and} \tag{3.4}$$

$$\max \quad \{N(\boldsymbol{x}, \boldsymbol{y}), N(\boldsymbol{y}, \boldsymbol{x})\} \geq T + 1. \tag{3.5}$$

Proof: "$\Rightarrow$" The proof is given by a contradiction to the assumption that neither condition a.) nor b.) must be fulfilled if a code $\mathcal{C}$ is (t_1, t_2)-SD.
Assume that for a given (t_1, t_2)-SD code $\mathcal{C}$ none of the conditions is satisfied. Namely, there exist code words $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$ such that

$$a'.)\quad \min \quad \{N(\boldsymbol{x}, \boldsymbol{y}), N(\boldsymbol{y}, \boldsymbol{x})\} \leq t \tag{3.6}$$

and

$$b'.)\quad \min \quad \{N(\boldsymbol{x}, \boldsymbol{y}), N(\boldsymbol{y}, \boldsymbol{x})\} = 0 \text{ or} \tag{3.7}$$

$$\max \quad \{N(\boldsymbol{x}, \boldsymbol{y}), N(\boldsymbol{y}, \boldsymbol{x})\} \leq T. \tag{3.8}$$

First one can show that for a code $\mathcal{C}$ being skew-detecting, there are no two code words $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$ such that $\min\{N(\boldsymbol{x}, \boldsymbol{y}), N(\boldsymbol{y}, \boldsymbol{x})\} = 0$, because this means that the code is a chain. Consequently the receiver cannot distinguish which code word was sent first. Assume therefore without loss of generality $N(\boldsymbol{x}, \boldsymbol{y}) = 0$, then $\boldsymbol{x} \subseteq \boldsymbol{y}$. Let $V = \boldsymbol{y} - \boldsymbol{x}$. Given a set $W \in \mathcal{P}([n])$, let $\hat{W}$ denote a sequence with elements of W in some order. The code words are also sets in the following. Assume that the following sequence is received:

$$\hat{Z} = \boldsymbol{x}, \hat{V}, \boldsymbol{y}, \ldots$$

Notice, that $\hat{Z}$ is a concatenation of elements of the described sets. The order of elements in the sets is arbitrary.
One can see that there is neither skew between $\boldsymbol{x}$ nor $\boldsymbol{y}$ with respect to the sequence $\hat{Z}$, i.e. $\mathcal{S}(\boldsymbol{x}; \hat{Z}) = \mathcal{S}(\boldsymbol{y}; \hat{Z}) = (0, 0)$. But one is not able to see which code word, $\boldsymbol{x}$ or $\boldsymbol{y}$, was sent first. This makes the assumption in (3.7) impossible. One has to suppose that

$$0 < \min\{N(\boldsymbol{x}, \boldsymbol{y}), N(\boldsymbol{y}, \boldsymbol{x})\} \leq t \text{ and} \tag{3.9}$$

$$\max\{N(\boldsymbol{x}, \boldsymbol{y}), N(\boldsymbol{y}, \boldsymbol{x})\} \leq T. \tag{3.10}$$

Without loss of generality, assume that $N(\boldsymbol{x}, \boldsymbol{y}) \leq t_1$ and $N(\boldsymbol{y}, \boldsymbol{x}) \leq t_2$. Let $U = \boldsymbol{x} \cap \boldsymbol{y}$, $V = \boldsymbol{y} - \boldsymbol{x}$ and $W = \boldsymbol{x} - \boldsymbol{y}$. Assume that the following is the received sequence:

$$\hat{Z} = \hat{U}, \hat{V}, \hat{W}, \hat{U}, \ldots.$$

Since $\boldsymbol{y} = (\boldsymbol{x} \cap \boldsymbol{y}) \cup (\boldsymbol{y} - \boldsymbol{x})$, there is no skew between $\boldsymbol{y}$ and $\hat{Z}$, i.e. $\mathcal{S}(\boldsymbol{y}; \hat{Z}) = (0, 0)$, and the receiver will conclude, after arrival of the last transition in $\hat{V}$, that $\boldsymbol{y}$ was transmitted.
However, it also holds

$$\mathcal{S}(\boldsymbol{x}; \hat{Z}) = (|W|, |V|) = (N(\boldsymbol{x}, \boldsymbol{y}), N(\boldsymbol{y}, \boldsymbol{x})) \leq (t_1, t_2).$$

Therefore, it could have happened that $\boldsymbol{x}$ was transmitted first and the skew with respect to the received sequence does not exceed (t_1, t_2), contradicting the fact that $\mathcal{C}$ is (t_1, t_2)-SD.

"$\Leftarrow$" The second part of the proof shows that these conditions are also sufficient for a code to be (t_1, t_2)-SD. The proof is based on showing that given a code $\mathcal{C}$ which satisfies the conditions, a decoding strategy exists which decodes the received sequence correctly if there is no skew, and detects an error provided that the skew does not exceed (t_1, t_2). The decoding algorithm is formally given in Appendix A.1. I will present the main ideas of it in form of a decoding strategy. Given a received sequence $\hat{Z} = z_1, z_2, \ldots, z_j, \ldots$ the receiver examines each arrival z_j. He checks if the set Z_j is a code word, i. e. $Z_j = \text{supp}\hat{Z}_j = \boldsymbol{x} \in \mathcal{C}$. If there is a repeated arrival, a skew error is detected and an error-detecting protocol is invoked. If a code word is found, it is produced as output and the process is restarted. In the proof it will be shown that it is impossible for the receiver to decode a code word erroneously which is wrongly placed in the message by the skew.

Now, let $t_1, t_2 \in \mathbb{N}_0$, and t and T as defined. Let $\mathcal{C}$ be a code satisfying at least one condition a.) or b.).

In the following the sequence $\hat{Z}$ one wants to observe always starts at z_1. Once the decoder has found a code word $\boldsymbol{x} \in \mathcal{C}$, the next arrival is denoted as z_1.

Assume that $\boldsymbol{x} \in \mathcal{C}$ has been transmitted. If $\mathcal{S}(\boldsymbol{x}; \hat{Z}) = (0, 0)$ the decoding algorithm will correctly decode $\boldsymbol{x}$. If $z_j \in Z_{j-1}$, then z_j is a repeated arrival and further transmission will be stopped. Now, let $Z_j = \{z_1, z_2, \ldots, z_j\}$, $j \geq 1$. One has to show that, if $\mathcal{S}(\boldsymbol{x}; \hat{Z}) \leq (t_1, t_2)$ and $Z_j \neq \boldsymbol{x}$, then $Z_j \notin \mathcal{C}$ since the code is unordered.

Assume therefore that $\boldsymbol{x}$ and Z_j are unordered and $Z_j \in \mathcal{C}$ for some $j \geq 1$. Since $\mathcal{S}(\boldsymbol{x}; \hat{Z}) \leq (t_1, t_2)$, $N(\boldsymbol{x}, Z_j) \leq t_1$ and $N(Z_j, \boldsymbol{x}) \leq t_2$, contradicting conditions a.) and b.).

Since the decoding strategy correctly decodes $\boldsymbol{x}$ when $\mathcal{S}(\boldsymbol{x}; \hat{Z}) = (0, 0)$ and never outputs a code word when $0 < \mathcal{S}(\boldsymbol{x}; \hat{Z}) \leq (t_1, t_2)$, then a repeated arrival indicates the occurence of skew. In the worst case, after n transitions in different channels have arrived, the $(n + 1)$-th transition will necessarily be a repeated arrival. Hence, $\mathcal{C}$ is (t_1, t_2)-SD. $\square$

The next two corollaries follow immediately from the theorem:

Corollary 3.10 A code $\mathcal{C}$ is (t_1, t_2)-SD, if and only if it is also (t_2, t_1)-SD.

Corollary 3.11 A code $\mathcal{C}$ is (t, t)-SD, if and only if, for any pair of distinct code words $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$, it holds

$$\min\{N(\boldsymbol{x}, \boldsymbol{y}), N(\boldsymbol{y}, \boldsymbol{x})\} \geq 1 \text{ and} \tag{3.11}$$
$$\max\{N(\boldsymbol{x}, \boldsymbol{y}), N(\boldsymbol{y}, \boldsymbol{x})\} \geq t + 1. \tag{3.12}$$

3.3.2 (t_1, t_2)-Skew-Tolerant Codes

The structure of this section is similar to the previous one: To characterize (t_1, t_2)-ST codes one uses distance properties between their code words. With the help of these conditions one can show that the receiver can successfully decode each transmitted message provided that the skew does not exceed (t_1, t_2). Codes that can correct and not only detect skew have the advantage of allowing continuous operation: The system does not need to be halted in presence of skew. Therefore, the skew is not noticed by the user. However, it will be no surprise to the reader that ST-codes require more redundancy than SD codes.

Theorem 3.12 For $t_1, t_2 \in \mathbb{N}_0$ let $t = \min\{t_1, t_2\}$.
The code $\mathcal{C}$ is (t_1, t_2)-skew-tolerant, if and only if
for any pair of distinct code words $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$ at least one of the following conditions occurs:

$$a.) \quad \min \quad \{N(\boldsymbol{x}, \boldsymbol{y}), N(\boldsymbol{y}, \boldsymbol{x})\} \geq t + 1 \tag{3.13}$$

or

$$b.) \quad \min \quad \{N(\boldsymbol{x}, \boldsymbol{y}), N(\boldsymbol{y}, \boldsymbol{x})\} \geq 1 \text{ and} \tag{3.14}$$
$$\max \quad \{N(\boldsymbol{x}, \boldsymbol{y}), N(\boldsymbol{y}, \boldsymbol{x})\} \geq t_1 + t_2 + 1. \tag{3.15}$$

Proof: "$\Rightarrow$" As in the proof of Theorem 3.9 assume that for a given (t_1, t_2)-ST code $\mathcal{C}$ the conditions a.) and b.) are not satisfied.
W. l. o. g., let $N(\boldsymbol{x}, \boldsymbol{y}) \leq N(\boldsymbol{y}, \boldsymbol{x})$, so negation of a.) and b.) gives

$$N(\boldsymbol{x}, \boldsymbol{y}) \leq t$$

and

$$N(\boldsymbol{x}, \boldsymbol{y}) = 0 \text{ or } N(\boldsymbol{y}, \boldsymbol{x}) \leq t_1 + t_2.$$

If $N(\boldsymbol{x}, \boldsymbol{y}) = 0$, the code $\mathcal{C}$ cannot be unordered which contradicts the fact that $\mathcal{C}$ is ST. It follows that the assumption has to be

$$1 \leq N(\boldsymbol{x}, \boldsymbol{y}) \leq t \text{ and} \tag{3.16}$$
$$N(\boldsymbol{y}, \boldsymbol{x}) \leq t_1 + t_2. \tag{3.17}$$

Since $N(\boldsymbol{y}, \boldsymbol{x}) = |\boldsymbol{y} - \boldsymbol{x}| \leq t_1 + t_2$, one can assume that $\boldsymbol{y} - \boldsymbol{x} = A_1 \cup A_2$, where $A_1 \cap A_2 = \emptyset$, $|A_1| \leq t_1$ and $|A_2| \leq t_2$.
As before, by $\hat{W}$ one denotes a sequence with elements of a given set W in some order. Let $U = \boldsymbol{x} \cap \boldsymbol{y}$ and $W = \boldsymbol{x} - \boldsymbol{y}$. Assume that we receive a sequence $\hat{Z}$ as

$$\hat{Z} = \hat{U}, \hat{A}_2, \hat{W}, \hat{A}_1, \hat{U}, \ldots.$$

According to Definition 3.2 of skew, one gets

$$\mathcal{S}(\boldsymbol{x}; \hat{Z}) = (|W|, |A_2|) \leq (t, t_2) \leq (t_1, t_2),$$

where the first inequality follows from the assumption (3.16) that $|W| = N(\boldsymbol{x}, \boldsymbol{y}) \leq t$. Therefore, since the code $\mathcal{C}$ is (t_1, t_2)-ST, $\boldsymbol{x}$ will be decoded as the transmitted code word.
On the other hand, one also gets

$$\mathcal{S}(\boldsymbol{y}; \hat{Z}) = (|A_1|, |W|) \leq (t_1, t) \leq (t_1, t_2).$$

Therefore, since code $\mathcal{C}$ is (t_1, t_2)-ST, $\boldsymbol{y}$ could also be decoded as the transmitted code word. This is a contradiction.

Before proving that the necessary conditions in Theorem 3.12 are also sufficient for a code to be (t_1, t_2)-ST, I will give the main ideas of the Decoding Algorithm formally given in Appendix A.2. As in the previous section the proof is based upon the existence of such a decoding algorithm. To my knowledge, there is a false reasoning in the Decoding Algorithm given in [6] which I will explain later on. First I will give a description of the corrected strategy. Obviously, it is more complicated than for the (t_1, t_2)-SD code, where the decoder only has to check for each arrival if the received set is a code word. Now one has to look into a window in the immediate past and eliminate possible sets of arrivals that could have been caused by skew. Skew may appear in two possible ways: through arrivals of transitions that do not belong in the current code word, and through repeated arrivals.
If skew occurs during transmission of a code word $\boldsymbol{x} \in \mathcal{C}$, there is a delay

part with maximum length of $t_1 + t_2$ transitions at the end of the complete transmission of $\boldsymbol{x}$. This delay part consists of at most t_1 elements of the current code word and at most t_2 errors caused by the skew. The task now is to find those l_2 elements, l_2 as defined in Definition 3.2, $0 \leq l_2 \leq t_2$, which are errors between the sent code word $\boldsymbol{x}$ and the received sequence $\hat{Z}$. This set of size l_2 has to be eliminated to get $\boldsymbol{x}$. Since the decoder does not know which code word was sent first, obviously, it knows nothing about either the beginning or the end of the delay part. So each arrival z_j could be the beginning of this part. The decoder observes the last $t_1 + t_2$ arrivals, i. e. $z_{j-(t_1+t_2)+1}, \ldots, z_{j-1}, z_j$. Denote these arrivals as sets consisting of one element $W_1, \ldots, W_{t_1+t_2}$ respectively and changing with each step of the decoding strategy. In each step one has to treat $W_1, \ldots, W_{t_1+t_2}$ as if they cover the delay part. The part ends with the last missing element belonging in the code word $\boldsymbol{x}$. This is the reason for looking at the sets $W_1, \ldots, W_{t_1+t_2-1}$ only, because if it is really the delay part one does not to eliminate $W_{t_1+t_2}$.
Since one does not know if there is an error or not, one has to check if the elimination of subsets of $W_1 \cup \ldots \cup W_{t_1+t_2-1}$ of sizes $0, \ldots, t_2$ produces a code word $\boldsymbol{x} \in \mathcal{C}$. More precisely, this means the following: suppose the strategy was done for Z_{j-1} without finding a code word of $\mathcal{C}$. z_j is the next arrival. So, observe $Z_j \triangleq Z_{j-1} \cup \{z_j\}$ and $W_{t_1+t_2} \triangleq \{z_j\}, W_{t_1+t_2-1} \triangleq \{z_{j-1}\}$, etc.
Then, for each

$$A \subseteq \bigcup_{s=1}^{t_1-1+i} W_{t_2-i+s} = W_{t_2-i+1} \cup \ldots \cup W_{t_1+t_2-1}$$

of size $|A| = i$, and for each $0 \leq i \leq t_2$ the decoder has to check if $Z_j - A \in \mathcal{C}$. This means for example for the biggest possible number of errors, i. e. the biggest possible set A, $|A| = t_2$, must be chosen from $W_1 \cup \ldots W_{t_1+t_2-1}$. (Notice, that for $1 \leq s \leq t_1 + t_2 - 2$ with $t_2 - i + s \geq t_1 + t_2 - 1$ it is $W_{t_2-i+s} \cup W_{t_1+t_2-1} = \emptyset$.)
If $Z_j - A \in \mathcal{C}$ the decoder will output this code word.

The main differences between the algorithm I will propose in the appendix and the one given in [6] are the following.
First, in [6] it was mentioned that repeated arrivals are ignored in the decoding of the current code word. They only have to be stored because they belong in another code word. Therefore, the authors propose to store all repeated arrivals in a set B. The cardinality of the set is bounded by t_2, because this is the maximum number of errors which may occur. They only

check the cardinality of this set without taking the repeated arrivals into further consideration.
The problem of this method is that they disregard that a repeated arrival belongs in the delay part. A repeated arrival is definitely *one* of the elements to be eliminated. (Therefore, it is correct that this element is of no use for the decoding.) If one puts this element in a storage set and contiues with the next arrival the delay part increases by one element. In the model where $\mathcal{S}(\boldsymbol{x}; \hat{Z}) \leq (t_1, t_2)$ is a global condition this may result in false decoding. Therefore, all arrivals must be entered in the windows.
The second problem of this is that in [6] the algorithm stops at the moment when a third arrival in one of the channels is observed, i. e. a second repetition.
This is not the case if one allows a set to have equal elements.
Another advantage of taking the repeated arrivals not out of consideration is the following:
After finding a code word $\boldsymbol{x} \in \mathcal{C}$ one has to eliminate the elements of $\boldsymbol{x}$ in the sequence. If there are elements in B which are now deleted in the sequence they are not longer repeated arrivals and one has to insert this elements at the right place of the sequence. This is not necessary if all arrivals are treated equivalently.
For the same reasons as above for the repeated arrivals I propose not to eliminate all elements of the decoded word $\boldsymbol{x}$, but the first $m = m(\boldsymbol{x}; \hat{Z})$. The others belong into the delay part of the next code word.
Besides one can label the decoded items of the sequence by a 0 to exclude them from further decoding.

Proof of Theorem 3.12: "$\Leftarrow$" One wants to show that a given code $\mathcal{C}$ fulfilling at least one of the conditions a.) or b.) is (t_1, t_2)-ST provided that the skew does not exceed (t_1, t_2). Suppose that $\boldsymbol{x} \in \mathcal{C}$ is the code word transmitted first and $\hat{Z}$ is the received sequence. Let $m = m(\boldsymbol{x}; \hat{Z})$, $r = r(\boldsymbol{x}; \hat{Z})$ and $\mathcal{S}(\boldsymbol{x}; \hat{Z}) = (l_1, l_2) \leq (t_1, t_2)$.
The proof that the code $\mathcal{C}$ is (t_1, t_2)-ST consists of two parts. In the first part it must be shown that $\boldsymbol{x}$ is one possible outcome of the decoding algorithm. Second, it must be garanteed that the decoder does not produce any code word different from $\boldsymbol{x}$ for any arrival z_j, $j \leq r$.
First it will be shown that $\boldsymbol{x}$ will be decoded when Z_r arrives. Assume w. l. o. g. that $\hat{Z}_r$ does not contain repeated arrivals because the repeated arrivals are not relevant for decoding. Furtherupon, let $C = Z_r - \boldsymbol{x}$, i. e.

$\boldsymbol{x} = Z_r - C$. Since there are no repeated arrivals $|C| = l_2$, i.e. the number of elements in Z_r not belonging in $\boldsymbol{x}$. One has to demonstrate that $Z_r - C$ is a possible outcome, i.e. $C \subseteq \bigcup_{s=1}^{t_1-1+l_2} W_{t_2-l_2+s}$. By definition of these sets, $W_{t_1+t_2} = \{z_r\}$ etc., one gets

$$\bigcup_{s=1}^{t_1-1+l_2} W_{t_2-l_2+s} = \{z_{r-t_1-l_2+1}, z_{r-t_1-l_2+2}, \ldots z_{r-1}\}.$$

Since $C \subseteq Z_{r-1} - Z_m = \{z_{m+1}, z_{m+2}, \ldots, z_{r-1}\}$, it is enough to show that $Z_{r-1} - Z_m \subseteq \{z_{r-t_1-l_2+1}, zr - t_1 - l_2 + 2, \ldots, z_{r-1}\}$, i.e.

$$m + 1 \geq r - t_1 - l_2 + 1.$$

The last inequality is equivalent to $r - m \leq t_1 + l_2$. But this is true, since $r - m = l_1 + l_2$ and $l_1 \leq t_1$.

To show that $\boldsymbol{x}$ is also the only possible outcome till the $r - th$ transition has arrived, one has to prove that for any $j \leq r$ and $Z_j - A \neq \boldsymbol{x}$, where A is defined as in the decoding strategy, $Z_j - A \notin \mathcal{C}$.
If $j \leq m$ it contradicts the fact that the code $\mathcal{C}$ is unordered, because $Z_j - A \subseteq Z_j \subseteq Z_m \subseteq \boldsymbol{x}$ holds.
Assume there exists j with $m + 1 \leq j \leq r$. When transition z_j arrives one can find a subset $A \subseteq \{z_{j-t_1-i+1}, z_{j-t_1-i+2}, \ldots, z_{j-1}\}$ where $|A| = i \leq t_2$ and the decoder produces a code word different from X, i. e. $Z_j - A \neq \boldsymbol{x}$ and $Z_j - A \in \mathcal{C}$.
To demonstrate that this assumption contradicts the conditions in the theorem one computes the values of $N(Z_j - A, \boldsymbol{x})$ and $N(\boldsymbol{x}, Z_j - A)$. Notice, that always $Z_m \subseteq A, Z_j, \boldsymbol{x} \subseteq Z_r$ holds.
Let $A \triangleq A_1 \cup A_2$, where $A_1 \triangleq A \cap Z_m$ and $A_2 \triangleq A - A_1$. Let $|A_1| = i_1$ and $|A_2| = i_2$, so that $i = i_1 + i_2$.
Observe first $N(Z_j - A, \boldsymbol{x})$. It holds

$$\begin{aligned} N(Z_j - A, \boldsymbol{x}) \leq N(Z_j, \boldsymbol{x}) &\leq N(Z_r, \boldsymbol{x}) \\ &= |Z_r - \boldsymbol{x}| = |C| = l_2 \leq t_2, \end{aligned}$$

where C is the set of errors without repeated arrivals as above.
From the decoding strategy one knows that $A \subseteq \{z - j - t_1 - i + 1, \ldots, z_{j-1}\}$ and by definition of A_1 one gets $A_1 \subseteq \{z_{j-t_1-i+1}, \ldots, z_m\}$.
Therefore, it holds

$$i_1 = |A_1| \leq |\{z_{j-t_1-i+1}, \ldots, z_m\}| = m + t_1 + i - j$$

and respectively by definition of i

$$j - m \leq t_1 + i_2 \Leftrightarrow -i_2 \leq t_1 + j + m$$

With the last equation and since $|Z_m| \leq |\boldsymbol{x}|$ and $|A_2| \leq |A|$ one can conclude

$$\begin{aligned} N(Z_j - A, \boldsymbol{x}) = |(Z_j - A) - \boldsymbol{x}| &= |(Z_j - X) - A| \\ &\leq |(Z_j - Z_m) - A_2| \\ &= j - m - i_2 \\ &\leq j - m + t_1 + j + m = t_1. \end{aligned}$$

Therefore, one gets

$$\begin{aligned} N(Z_j - A, \boldsymbol{x}) &\leq t_1 \text{ and} \\ N(Z_j - A, \boldsymbol{x}) &\leq t_2. \end{aligned}$$

If $t \triangleq \min\{t_1, t_2\}$, then $N(Z_j - A, \boldsymbol{x}) \leq t$.

Second, for $N(\boldsymbol{x}, Z_j - A)$, it holds

$$(\boldsymbol{x} - (Z_j - A)) \subseteq (\boldsymbol{x} - (Z_m - A)) \subseteq ((\boldsymbol{x} - Z_m) \cup A) \subseteq ((\boldsymbol{x} \cap \{Z_r - Z_m\}) \cup A).$$

And, since $|(Z_r - Z_m) \cap \boldsymbol{x}| = l_1 \leq t_1$, where the equality follows from definition, and the inequality is given by the explained assumption. Also, one knows that $|A| = i \leq t_2$ by definition of the set A in the decoding strategy. So, one obtains

$$N(\boldsymbol{x}, Z_j - A) \leq t_1 + t_2.$$

But the inequalities $N(Z_j - A, \boldsymbol{x}) \leq t$ and $N(\boldsymbol{x}, Z_j - A) \leq t_1 + t_2$ contradict the assumption. This shows that $\boldsymbol{x} = Z_r - C$ is the only possible outcome of the stategy. This completes the proof. $\square$

The following corollary is clear from Theorem 3.12.

Corollary 3.13 A code $\mathcal{C}$ is (t_1, t_2)-ST, if and only if it is also (t_2, t_1)-ST.

Chapter 4

Skew Code Constructions

In this chapter basically three code constructions for (t_1, t_2)- SD and (t_1, t_2)-ST codes are given. As found out in Chapter 3 the code words of these codes have to fulfill certain distance properties. Currently, there are two possibilities to get a construction of a skew code. The first one uses the fact that other known codes fulfill similar conditions for their code words. Hence, one can adopt their constructions. The other method is a direct one.
In the following two sections these methods are explained. In Section 4.1 two families of codes are introduced. The first one is well known in literature, the so-called t-error-correcting/all unidirectional error-detecting code. The other is the t-error correcting unordered code. It will be shown that the deterministic conditions of these codes cover those of a skew code. At most one has to take care that the code provides a certain minimum distance d.
In the second section another construction is given to fulfill merely the deterministic conditions of a skew code.
The constructions are summarized in Constructions A, B and C. Small derivations and improvements are labeled with a prime. All three constructions have in common that one starts with a standard random error-correcting code and then appends a tail in such a way that the code suffices all the conditions required. In any case the code has to be an antichain as demonstrated in Chapter 3. Nevertheless, not all constructions are leading to a systematic code which is desirable for simple encoding and decoding. The advantage of constructions built up on systematic t-EC codes is that the final code is also systematic.
In sections 4.3 and 4.4 Construction A, B and C will be compared. First by giving an example, second by calculating the redundancies.

4.1 Utilization of Other Codes

4.1.1 The t-Error-Correcting/All Unidirectional Error-Detecting Code

In this section I introduce the method of constructing a skew code by using the t-error-correcting/all unidirectional error-detecting code, short t-EC/AUED code. This type has been extensively discussed for improving the reliability of computer systems and communication networks. The problem of constructing t-EC/AUED codes has received wide attention in recent literature, see [11]-[23]. Generally, there are two different methods to construct t-EC/AUED codes: the group theoretic method and the method of appending a special tail to a linear code. In the literature the standard way is the last one which I have chosen to describe in more detail.

The first published work in the area of t-EC/AUED codes is reported in the paper of Pradhan, [11]. Pradhan and Bose [12] and later Rao, and [13], have developed a class of single error correcting/all unidirectional error detecting (SEC/AUED) codes. New constructions for this kind of code can be found, for instance, in [20] by Zhang and Tu. Some of the codes are derived by using Berger's technique [2], some by using the group theoretic method. In [18] systematic t-EC/AUED codes with easy encoding and decoding are presented by Wang and Jha. They are comparable in redundancy to the most efficient systematic codes in [15] and [16]. Currently, the most popular construction for the t-EC/AUED codes is to choose a t-EC code and then append a tail in such a way that the new code can detect more than t errors when they are unidirectional. This tail is a function of the weights of the code words.
t-EC/AUED codes are only one possibility to derive skew codes so I have followed only the developement of the above mentioned method. Blaum and van Tilborg first have described this construction in [17]. Since the construction is based upon the best available random error-correcting and asymmetric error-correcting codes, the construction will be improved if the codes are improved. From this several improvements are made to reduce the redundancy of the code, for instance in [19], [21], [22] and [23]. I have included most of them; at least the idea is shortly presented.

The section is organized as follows. After defining the code, Theorem 4.2 which was proved in [11] and [13] states its deterministic condition. It is given by a distance property in the same way as it was done in Chapter 3 for

the skew code. This makes it easy to establish the connection between these codes and skew codes. The following results taken from [17] are leading to Theorem 4.6 which describes a concatenation of the t-EC code words with a tail to obtain a t-EC/AUED and therefore a skew code as shown next. In Lemmata 4.7, 4.9 and 4.11 three methods for certain tail matrices are given which are needed in Construction A.
Different techniques to reduce the redundancy r in special cases were given in [19], [21] and [22]. All these techniques are mainly based upon modifying the t-EC code in such a way that the weight distribution of the original code is reduced. The method proposed by Bruck and Blaum in [19] leads to the Construction A'. Al-Bassam and Bose use in [21] the coset codes which have narrow weight distribution to reduce r in some cases. The main difference between the construction in [19] and [21] and the one given in [22] by Katti and Blaum is the replacement of the all-0 code word from the t-EC code with another code word. It will be explained precisely in the Construction A".
Two other methods to reduce r are given by Zhang and Tu in [20] and by Yang and Laih in [23]. I will sketch the ideas.
Good tables and comparisons of the parameters of t-EC/AUED codes can be found in [17] and [19].

I have already mentioned that another method, the group theoretic method, has been studied, for instance, in [10], [13] and [20]. There also exist some ad hoc methods for constructing EC/AUED codes in [28] which are not included.

Definition 4.1 A code $\mathcal{C}$ is called *t-error-correcting/all unidirectional error-detecting*, short t-EC/AUED, if it
a.) can correct up to t random errors and
b.) is capable of detecting all error patterns of the unidirectional type.

The following theorem gives necessary and sufficient conditions for a code to be t-EC/AUED.

Theorem 4.2 A code $\mathcal{C}$ is t-EC/AUED if and only if,
for any pair of distinct code words $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$ it holds $N(\boldsymbol{x}, \boldsymbol{y}) \geq t+1$.

Proof: "$\Leftarrow$" Clearly, the code $\mathcal{C}$ is t-error-correcting.
Moreover, if at most t errors have been made in a transmitted code word $\boldsymbol{x}$, the received word $\boldsymbol{x}'$ will still satisfy

$$\min\{N(\boldsymbol{x}', \boldsymbol{y}), N(\boldsymbol{y}, \boldsymbol{x}')\} \geq 1, \text{ for all } \boldsymbol{y} \in \mathcal{C}.$$

So, $\boldsymbol{x}'$ with $d_H(\boldsymbol{x}, \boldsymbol{x}') \leq t$ cannot also be the result of an unidirectional error pattern having occured in any code word $\boldsymbol{y} \in \mathcal{C}$.
"$\Rightarrow$" Assume there exist $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$ with $N(\boldsymbol{x}, \boldsymbol{y}) = s \leq t$. Let $\boldsymbol{x}'$ be the vector obtained from $\boldsymbol{x}$ by changing s coordinates, where $\boldsymbol{x}$ is 1 and $\boldsymbol{y}$ is 0, into 0. Then $d_H(\boldsymbol{x}, \boldsymbol{x}') = s$ and $\boldsymbol{x}'$ is at the same time the result of $N(\boldsymbol{x}, \boldsymbol{y})$ unidirectional errors in $\boldsymbol{y}$. So, $\mathcal{C}$ is not t-EC/AUED. □

This theorem states that an t-EC/AUED code satisfies the condition a.) for (t_1, t_2)-SD and ST codes if the skew between the transmitted code word and the received sequence does not exceed (t_1, t_2), and $t = \min\{t_1, t_2\}$. The next lemma follows immediately:

Lemma 4.3 Let $t_1, t_2 \in \mathbb{N}, t = \min\{t_1, t_2\}$. Let $\mathcal{C}$ be an t-EC/AUED code. Then $\mathcal{C}$ is (t_1, t_2)-SD and ST.

The property of an t-EC/AUED is that whenever more than t errors occur which are unidirectional, the received word does not fall into a sphere of radius t whose center is a code word.
The most common method to construct a systematic t-EC/AUED code is to choose an t-error-correcting (EC) code and append a tail such that the new code can detect more than t errors when they are unidirectional. The tail is a function of the weight of the code word.
I will now present this construction. In [17] it was proved that for larger dimensions it gives lower redundancy than the known constructions in [11], [12], [13], [14] and [15].
One starts by using a linear $[n',\, k,\, 2t+1]$ code $\mathcal{C}'$, which is t-EC, as a first step to encode the k information symbols. The next proposition shows what can be said about $N(\boldsymbol{x}, \boldsymbol{y})$ and $N(\boldsymbol{y}, \boldsymbol{x})$. The following notation will be used:

$$\begin{aligned} \lceil x \rceil &\triangleq j, \text{ where } j \in \mathbb{N} \text{ and } x \leq j < x+1, \\ \lfloor x \rfloor &\triangleq i, \text{ where } i \in \mathbb{N} \text{ and } x-1 < i \leq x. \end{aligned}$$

Proposition 4.4 Let $\mathcal{C}'$ be an $[n',\, k,\, 2t+1]$ code. Then, for any pair of distinct code words $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}'$ with $\omega(\boldsymbol{x}) = i \leq j = \omega(\boldsymbol{y})$ it holds

$$N(\boldsymbol{y}, \boldsymbol{x}) \geq t+1 \tag{4.1}$$

and

$$N(\boldsymbol{x}, \boldsymbol{y}) \geq \max\left\{0, t+1-\left\lceil\frac{j-i}{2}\right\rceil\right\}. \tag{4.2}$$

Proof: Without loss of generality $\boldsymbol{x}$ and $\boldsymbol{y}$ can be assumed to look like

$$\begin{array}{rcl} \boldsymbol{x} & = & \overbrace{11\ldots1}^{a}\ \overbrace{11\ldots1}^{b}\ \overbrace{00\ldots0}^{c}\ \overbrace{00\ldots0}^{d} \\ \boldsymbol{y} & = & 11\ldots1\ 00\ldots0\ 11\ldots1\ 00\ldots0. \end{array}$$

Of course, $a+b+c+d=n'$, but also it holds

$$a+b=i,\ a+c=j \tag{4.3}$$

$$b+c=d_H(\boldsymbol{x},\boldsymbol{y}) \geq 2t+1 \tag{4.4}$$

$$b=i-a \leq j-a=c \tag{4.5}$$

From (4.5) and (4.4) it follows that $2N(\boldsymbol{y},\boldsymbol{x})=2c \geq b+c \geq 2t+1$.
Since $N(\boldsymbol{y},\boldsymbol{x}) \in \mathbb{N}$ the first statement (4.1) of the proposition is proved.
But from (4.3) one also has $c=j-a=b+j-i$ and therefore $2b+j-i=b+c \geq 2t+1$. Hence,

$$N(\boldsymbol{x},\boldsymbol{y})=b \geq \left\lceil \frac{2t+1-j+i}{2} \right\rceil = t+1-\left\lceil \frac{j-i}{2} \right\rceil .$$

□

To construct now an t-EC/AUED code $\mathcal{C}$ from a $[n',\ k,\ 2t+1]$ code $\mathcal{C}'$, one needs to add a tail $\boldsymbol{t}_x$ to each code word $\boldsymbol{x} \in \mathcal{C}'$ with the property that for each $\boldsymbol{x},\boldsymbol{y} \in \mathcal{C}'$, $\boldsymbol{x} \neq \boldsymbol{y}$ it holds $N(\boldsymbol{t}_x,\boldsymbol{t}_y) \geq t+1-N(\boldsymbol{x},\boldsymbol{y})$.
Let $w(\boldsymbol{x})=i \leq j=w(\boldsymbol{y})$. According to Proposition 4.4 and since $N((\boldsymbol{x},\boldsymbol{t}_x),(\boldsymbol{y},\boldsymbol{t}_y))=N(\boldsymbol{x},\boldsymbol{y})+N(\boldsymbol{t}_x,\boldsymbol{t}_y)$ it would suffice if

$$\begin{array}{rcl} N(\boldsymbol{t}_x,\boldsymbol{t}_y) & \geq & t+1-\max\left\{0, t+1-\left\lceil \dfrac{j-i}{2} \right\rceil\right\} \\ & \geq & \min\left\{t+1, \left\lceil \dfrac{j-i}{2} \right\rceil\right\}. \end{array}$$

Since the tail to be added only depends on the weight of a code word $\boldsymbol{x}$, one can write $\boldsymbol{t}_i$ with $i=\omega(\boldsymbol{x})$, instead of $\boldsymbol{t}_x$.

Definition 4.5 An $(p \times q)$-matrix with entries in $\{0,1\}$ is called *descending tail matrix of strength* s with rows $\boldsymbol{t}_i$, $0 \leq i \leq p-1$, denoted by $T(p,q;s)$, if for all $0 \leq i \leq j \leq p-1$, it holds

$$N(\boldsymbol{t}_i,\boldsymbol{t}_j) \geq \min\left\{s, \left\lceil \frac{j-i}{2} \right\rceil\right\}.$$

In view of the preceeding, the following theorem is now obvious.

Theorem 4.6 Let $\mathcal{C}'$ be an $[n', k, 2t+1]$ linear systematic code and T be a descending tail matrix $T(n'+1, q; t+1)$ with rows $\boldsymbol{t}_i$, $0 \le i \le n'$. Then

$$\mathcal{C} \triangleq \{(\boldsymbol{x}, \boldsymbol{t}_{\omega(x)}) : \boldsymbol{x} \in \mathcal{C}'\}$$

is a t-EC/AUED code of length $n = n' + q$ which is systematic on k positions.

By Lemma 4.3 and Theorem 4.6 one gets the following first construction, developed by Blaum and van Tilborg.

Construction A

Assume one wants to construct an (t_1, t_2)-SD or ST code $\mathcal{C}$ of dimension k. Let $t = \min\{t_1, t_2\}$. Choose an $[n', k, 2t+1]$ EC code $\mathcal{C}'$. Consider a descending tail matrix $T(n'+1, q; t+1)$ with rows $\boldsymbol{t}_i$, $0 \le i \le n'$.
Let $\boldsymbol{u}$ be an information vector of length k. Then proceed as follows.

1. Encode $\boldsymbol{u}$ into a vector $\boldsymbol{v} \in \mathcal{C}'$.
2. Let $j \triangleq \omega(\boldsymbol{v})$ be the Hamming weight of $\boldsymbol{v}$. Then append to $\boldsymbol{v}$ the jth row of $T(n'+1, q; t+1)$.

The problem of constructing an t-EC/AUED code reduces to a problem of developing efficient descending tail matrices T. The following lemma which was also given in [17] shows one method. I supplied the proof of Lemma 4.7 by giving a complete induction.

Lemma 4.7 Let T_u be the $(2u \times u)$-matrix with entries in $\{0, 1\}$ defined recursively by $T_1 \triangleq \binom{1}{0}$

$$T_{u+1} \triangleq \left(\begin{array}{ccc|c} 1 & 1 & \cdots \; 1 & 1 \\ \hline & & & 0 \\ & & & 1 \\ & & & 0 \\ & T_u & & \cdot \\ & & & \cdot \\ & & & \cdot \\ & & & 1 \\ \hline 0 & 0 & \cdots \; 0 & 0 \end{array} \right). \tag{4.6}$$

Then T_u is a descending tail matrix $T(2u, u; s)$ for each value $s \in \mathbb{N} \setminus \{0\}$.

Proof: To justify that T_u with rows $\boldsymbol{t}_0, \boldsymbol{t}_1, \dots \boldsymbol{t}_{2u-1}$ is a $T(2u, u; s)$-matrix one has to prove that for all $0 \le i \le j \le 2u-1$ it holds

$$N(\boldsymbol{t}_i, \boldsymbol{t}_j) \ge \min\left\{s, \left\lceil \frac{j-1}{2} \right\rceil\right\}.$$

For $u = 1$ it is $D_1 = \binom{1}{0} = \binom{\boldsymbol{t}_0}{\boldsymbol{t}_1}$. Therefore, one has three values to look at, $N(\boldsymbol{t}_0, \boldsymbol{t}_0)$, $N(\boldsymbol{t}_0, \boldsymbol{t}_1)$ and$N(\boldsymbol{t}_1, \boldsymbol{t}_1)$. Since for every $s \in \mathbb{N}$ it holds

$$\begin{aligned}
N(\boldsymbol{t}_0, \boldsymbol{t}_0) &= N(1,1) = 0 \ge \min\{s, \lceil 0/2 \rceil\} = 0 \\
N(\boldsymbol{t}_0, \boldsymbol{t}_1) &= N(1,0) = 1 \ge \min\{s, \lceil 1/2 \rceil\} = 1 \\
N(\boldsymbol{t}_1, \boldsymbol{t}_1) &= N(0,0) = 0 \ge \min\{s, \lceil (1-1)/2 \rceil\} = 0
\end{aligned}$$

the assumption that T_1 is a $T(2,1;s)$ is verified.
Assume that T_{u-1} is a descending tail matrix of strength s with rows $\boldsymbol{t}'_0, \boldsymbol{t}'_1, \dots, \boldsymbol{t}'_{2(u-1)-1} \in \{0,1\}^{u-1}$. T_u has rows $\boldsymbol{t}_0, \boldsymbol{t}_1, \dots, \boldsymbol{t}_{2u-1} \in \{0,1\}^u$ as defined in (4.6), i. e.

$$\begin{array}{rcl}
\boldsymbol{t}_0 &= \overbrace{(11\dots11)}^{u} =& (\boldsymbol{t}'_0, 1) \\
\boldsymbol{t}_1 &= (11\dots10) =& (\boldsymbol{t}'_0, 0) \\
\boldsymbol{t}_2 &= \quad \dots \quad =& (\boldsymbol{t}'_1, 1) \\
& \vdots & \\
\boldsymbol{t}_{2u-2} &= (00\dots01) =& (\boldsymbol{t}'_{2(u-1)-1}, 1) \\
\boldsymbol{t}_{2u-1} &= (00\dots00) =& (\boldsymbol{t}'_{2(u-1)-1}, 0)
\end{array}.$$

Obviously $N(\boldsymbol{t}_i, \boldsymbol{t}_j)$ can not get smaller than $N(\boldsymbol{t}'_i, \boldsymbol{t}'_j)$. Exact calculation demonstrates that for $0 \le j \le 2u-1$ there are four possible cases:

$$N(\boldsymbol{t}_i, \boldsymbol{t}_j) = N((\boldsymbol{t}'_{i-1}, 0), (\boldsymbol{t}'_{j-1}, 0)) = N(\boldsymbol{t}'_{i-1}, \boldsymbol{t}'_{j-1}) \tag{4.7}$$
$$N(\boldsymbol{t}_i, \boldsymbol{t}_j) = N((\boldsymbol{t}'_{i-1}, 0), (\boldsymbol{t}'_{j-1}, 1)) = N(\boldsymbol{t}'_{i-1}, \boldsymbol{t}'_{j-1}) \tag{4.8}$$
$$N(\boldsymbol{t}_i, \boldsymbol{t}_j) = N((\boldsymbol{t}'_{i-1}, 1), (\boldsymbol{t}'_{j-1}, 0)) = N(\boldsymbol{t}'_{i-1}, \boldsymbol{t}'_{j-1}) + 1 \tag{4.9}$$
$$N(\boldsymbol{t}_i, \boldsymbol{t}_j) = N((\boldsymbol{t}'_{i-1}, 1), (\boldsymbol{t}'_{j-1}, 1)) = N(\boldsymbol{t}'_{i-1}, \boldsymbol{t}'_{j-1}). \tag{4.10}$$

Since T_{u-1} is a descending tail matrix it is

$$N(\boldsymbol{t}'_{i-1}, \boldsymbol{t}'_{j-1}) \ge \min\left\{s, \left\lceil \frac{(j-1)-(i-1)}{2} \right\rceil\right\} = \min\left\{s, \left\lceil \frac{j-i}{2} \right\rceil\right\}.$$

Therfore, the inequality to be proved holds in (4.7)-(4.10), and shows that T_u is a $T(2u, u; s)$-matrix for every positive value of s. □

Remark 4.8 There is no similar method for n' even.

T_u will be an efficient descending tail matrix only for very small values of n'. For larger values of n' one shall need more advanced constructions. Therefore, a further method will be described next, which is also taken from [17].

Lemma 4.9 Let $\mathcal{D}$ be an asymmetric error-correcting code of length l, cardinality $|\mathcal{D}|$ and asymmetric minimum distance Δ. Let A be a $(|\mathcal{D}| \times l)$-matrix whose rows are the code words of $\mathcal{D}$ in order of nonascending weight.
Then A is a descending tail matrix $D(|\mathcal{D}|, l; \Delta)$.

Proof: Compare the rows $\boldsymbol{a}_i$ and $\boldsymbol{a}_j, i < j$, in A. W. o. l. g. they look like

$$\begin{aligned} \boldsymbol{a}_i &= \overbrace{11\ldots1}^{a}\ \overbrace{11\ldots1}^{b}\ \overbrace{00\ldots0}^{c}\ \overbrace{00\ldots0}^{d} \\ \boldsymbol{a}_j &= 11\ldots1\ 00\ldots0\ 11\ldots1\ 00\ldots0. \end{aligned}$$

Since $\mathcal{D}$ has asymmetric minimum distance Δ, one knows that $\max\{b, c\} \geq \Delta$. But $i < j$, so $a + b = \omega(\boldsymbol{a}_i) \geq \omega(\boldsymbol{a}_j) = a + c$. Hence $b \geq c$ and thus $N(\boldsymbol{a}_i, \boldsymbol{a}_j) = b = \max\{b, c\} \geq \Delta$. So, by Definition 4.5 A is a descending tail matrix of strength Δ. $\square$

Remark 4.10 Since $N(\boldsymbol{a}_i, \boldsymbol{a}_{i+1}) \geq \Delta$ in Lemma 4.9, while by Definition 4.5 one only needs that $N(\boldsymbol{a}_i, \boldsymbol{a}_{i+1}) \geq 1$ the matrix given in Lemma 4.9 will not yield the best construction of t-EC/AUED codes.

The next lemma combines Lemma 4.7 and Lemma 4.9.

Lemma 4.11 Let $\mathcal{D}$ be an asymmetric error-correcting code of length l with minimum asymmetric distance $\Delta = t + 1$. Let A be a $(|\mathcal{D}| \times l)$-matrix whose rows $\boldsymbol{a}_i, 1 \leq i \leq |\mathcal{D}|$ are the code words of $\mathcal{D}$ in order of nonascending weight. Let $\boldsymbol{t}_j, 1 \leq j \leq 2u$, be the rows of T_u for some fixed value u (T_u as defined above). Define a $(2u|\mathcal{D}| \times (l + u))$-matrix M by the *external* (or *Kronecker*) *product* of A and T_u

$$M \triangleq A \times T_u = \begin{pmatrix} \boldsymbol{a}_1 \times T_u \\ \boldsymbol{a}_2 \times T_u \\ \vdots \\ \boldsymbol{a}_{|\mathcal{D}|} \times T_u \end{pmatrix}, \text{where } \boldsymbol{a}_i \times T_u \text{ stands for } \begin{pmatrix} \boldsymbol{a}_i, \boldsymbol{t}_1 \\ \boldsymbol{a}_i, \boldsymbol{t}_2 \\ \vdots \\ \boldsymbol{a}_i, \boldsymbol{t}_{2u} \end{pmatrix}.$$

Then M is a descending tail matrix $T(2u|\mathcal{D}|, l + u; t + 1)$.

Proof: Compare $(\boldsymbol{a}_i, \boldsymbol{t}_v)$ and $(\boldsymbol{a}_j, \boldsymbol{t}_w), 1 \le i \le j \le |\mathcal{D}|, 1 \le v < w \le 2u$.
If $i < j$, then $N(\boldsymbol{a}_i, \boldsymbol{a}_j) \ge t+1$.
If $i = j$ and $v < w$, then $N(\boldsymbol{t}_v, \boldsymbol{t}_w) \ge \lceil \frac{w-v}{2} \rceil \ge \min\{t+1, \lceil \frac{w-v}{2} \rceil\}$. □

When using Lemma 4.11 to find t-EC/AUED codes one, of course, wants to minimize the redundancy of the code. Therefore, one has to find an asymmetric t-error-correcting code $\mathcal{D}$ of length l and $|\mathcal{D}| = m$, and a tail matrix T_u such that $2um \ge n' + 1$ and $l + u$ is minimal (Use, for example, TABLE I in [17]). Then the descending tail matrix $T(n'+1, q; t+1)$ needed in Construction A is given by the matrix M in Lemma 4.11.
In [19] several improvements on the tail matrices can be found.

Since this construction depends on the best asymmetric error-correcting code available, Bruck and Blaum propose in [19] two further techniques to reduce the redundancy of t-EC/AUED codes. Those two methods can be used together. The first involves using t-EC codes that contain the all-1 vector (for instance, BCH codes and Golay codes have this property, see [5]). Second, when choosing a code word, one takes either a code word or its complement, according to which of the two has smaller weight. One has to pay one bit for this operation, but the weight distribution is reduced by half. Finally, one appends a tail as descibed in Construction A together with Lemma 4.11. Overall, one will often gain in redundancy. In [19] there are comparisons between the parameters of t-EC/AUED codes, $t = 1, \cdots, 4$ constructed by method A and A'. Construction A' ties the results from A in most cases, but quite often it also improves them.

Construction A'

Assume one wants to construct an (t_1, t_2)-ST or SD code $\mathcal{C}$ of dimension k. Let $t \triangleq \min\{t_1, t_2\}$. Choose an $[n', k+1, 2t+1]$ EC code $\mathcal{C}'$ containing the all-1 vector with n' as small as possible. Choose a $T(\lfloor n'/2 \rfloor + 1, q; t+1)$ descending tail matrix T with rows $\boldsymbol{t}_i, 0 \le i \le \lfloor n'/2 \rfloor$ and q as small as possible. Then proceed as in steps 1. and 2. in Construction A.

The code $\mathcal{C}$ of Construction A

$$\mathcal{C} \triangleq \{(\boldsymbol{x}, \boldsymbol{t}_{\omega(\boldsymbol{x})}) : \boldsymbol{x} \in \mathcal{C}', \omega(\boldsymbol{x}) \le n'/2\},$$

is t-EC/AUED since the subset of code words of weight less or equal $\lfloor n'/2 \rfloor$ is still a t-EC code. According to A, the tail makes it t-EC/AUED.

In order to shorten a code containing the all-1 vector in such a way that the shortended code also contains the all-1 vector, one can use the following

Lemma 4.12 Let $\mathcal{C}$ be an $[n,k,d]$ EC code with parity check matrix H. Assume that the all-1 vector is in $\mathcal{C}$. Let $\boldsymbol{x}$ be a code word in $\mathcal{C}$ such that its nonzero components are $i_1, i_2. \cdots .i_\omega$, $1 \leq i_1 < i_2 \cdots < i_\omega \leq n$. Let $\tilde{H}$ be the matrix obtained by deleting columns $i_1, i_2, \cdots, i_\omega$ from H. Let $\tilde{\mathcal{C}}$ be the $[n-\omega, k-\omega, d]$ code whose parity check matrix is $\tilde{H}$. Then the all-1 vector is in $\tilde{\mathcal{C}}$.

Proof: The all-1 vector is in $\tilde{\mathcal{C}}$ if and only if the sum of all the columns in $\tilde{H}$ gives the zero column.
Since the all-1 vector $\mathbf{1}$ is in $\mathcal{C}$, then $\mathbf{1}+\boldsymbol{c}$ is also in $\mathcal{C}$. This vector has zero components $i_1, i_2, \cdots i_\omega$. Summing the columns corresponding to the nonzero components, one obtains the zero column. But these columns correspond to the columns in $\tilde{H}$. $\square$

In [23] Yang and Laih present a further method to reduce the weight distribution of t-EC codes by using a $DC_m(k+d,k)$ code and the method of [19] simultaneously. Their method is proposed for constructing an efficient quasi-systematic t-EC/AUED code. $DC_m(n,k)$ denotes a balanced (or dc-free) code of length n and dimension k where $\lceil \frac{n-m}{2} \rceil \leq \omega(\boldsymbol{x}) \leq \lceil \frac{n+m}{2} \rceil$, for all $\boldsymbol{x} \in DC_m(n,k)$. All words in a balanced code have equal number of ones and zeros, see [24]. The basic idea of the proposed codes is based on that DC_m codes have a narrow weight distribution. If one chooses DC_m codes as the input of systematic t-EC codes, the number of different weights of t-EC codes can be reduced.

The construction described in [22] by Katti and Blaum depends on the replacement of the all-0 code word by a code word of weight $\lceil n'/2 \rceil$ in a EC code $\mathcal{C}'$.

<u>**Construction A”**</u>

Assume one wants to construct an (t_1,t_2)-ST or SD code $\mathcal{C}$ of dimension k. Let $t \triangleq \{t_1,t_2\}$. Choose an $[n', k+1, 2t+1]$ EC code $\mathcal{C}'$ containing the all-1 vector with n' as small as possible and a $T(\lceil n'/2 \rceil - 2t, q; t+1)$ descending tail matrix T with rows $\boldsymbol{t}_i$, $0 \leq i \leq n'/2 - 2t - 1$, and q as small as possible. If $\boldsymbol{u}$ is an information vector of length k, then $\boldsymbol{u}$ is encoded as follows:

1. If $\boldsymbol{u} \neq \mathbf{0}$, let $\boldsymbol{c} \in \mathcal{C}'$ be the encoding of $(\boldsymbol{u}, 0)$ in $\mathcal{C}'$. If $\omega(\boldsymbol{c}) \leq \lfloor n'/2 \rfloor$ then encode $\boldsymbol{u}$ as $\boldsymbol{c}$, otherwise encode it as $\boldsymbol{c} \leftarrow \mathbf{1} + \boldsymbol{c}$, (in words, as the complement of $\boldsymbol{c}$.)

2. If $\boldsymbol{u} = \mathbf{0}$, then encode $\boldsymbol{u}$ as a (fixed) $\boldsymbol{c} \in \mathcal{C}'$ of weight $\lceil n'/2 \rceil$ such that $\boldsymbol{c}$ does not belong in the set of code words obtained in step 1.

3. Append to the vector $\boldsymbol{c}$ obtained in steps 1 or 2 the tail $\boldsymbol{t}_{\omega(c)-2t-1}$, completing the encoding.

The replacement results in a $T(\lceil n'/2 \rceil - 2t, q; t+1)$ matrix instead of a $T(\lfloor n'/2 \rfloor + 1, q; t+1)$ matrix.
If n' is even, since one is using only half of the code words of $\mathcal{C}'$ in Construction A", then, in particular, half of the code words of weight $n'/2$ are not assigned to information vectors and one of them can be chosen to encode the all-0 vector.
If n' is odd then no code word of weight $\lceil n'/2 \rceil$ is assigned to an information vector and any code word with that weight can b chosen to represent the all-0 vector.

Zhang and Tu remark in [20] that in the above described construction the tail actually consits of two parts: the parity check bits of a linear code and the bits used to code the weights of the code words of the linear code. In their paper they discuss a new approach for constructing a t-EC/AUED code. The idea is to design the tail as a whole without paritioning into two parts. They partition the systematic part into the union of t-EC/AUED subcodes. The tail bits are used to code the indices of these t-EC/AUED subcodes so that, as a whole, the t-EC/AUED conditions are satified. They propose two ways to partion the message sequences: the group theoretic method and the linear code syndrome method. These methods has been shown to be more efficient in the cases of $t = 1$ and $t = 2$ when $k \leq 31$, see [20].

4.1.2 The t-Error-Correcting Unordered Code

The structure of this section is similar to the previous one. One wants to construct a skew code by using another code for which the construction is known. In this section the t-error-correcting unordered code will be introduced. First I will give as usual its definition. Then Lemma 4.16 shows the connection between these codes and skew codes. Construction B describes the use of an error-correcting unordered code to obtain a skew code. Lemma 4.18 justifies this method. These results are taken from [6] or [30] found out by Blaum and Bruck. Although the given construction is not globally optimal, it has the advantage that the resulting code is systematic if the error-correcting code used for the construction is systematic. In chapter 6 it will be proved that the code is optimal when the error correcting code is an extended Hamming code, and also for certain BCH codes.
In [30] the same authors propose two further constructions of ECU codes. One of them works for the case in which the minimum distance is either 3 or 4 and has the advantage that it is systematic. The third construction is based on Sperner's result and is more efficient for large n. Since it has the disadvantage that it is not systematic I only describe its idea.

Definition 4.13 A code $\mathcal{C}$ is called t *error-correcting unordered*, denoted by t-ECU, if it
a.) can correct t errors and
b.) for any pair of distinct code words $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$ it holds
$\min\{N(\boldsymbol{x}, \boldsymbol{y}), N(\boldsymbol{y}, \boldsymbol{x})\} \geq 1$, i. e. any two code words are unordered.

Remark 4.14 The condition in b.) states nothing but $\mathcal{C}$ is an antichain.

Definition 4.15 A code $\mathcal{C}$ is *error-correcting unordered* (ECU) with minimum distance d if for any code words $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$ it holds
a.) $d_H(\boldsymbol{x}, \boldsymbol{y}) = N(\boldsymbol{x}, \boldsymbol{y}) + N(\boldsymbol{y}, \boldsymbol{x}) \geq d$ and
b.) $\min\{N(\boldsymbol{x}, \boldsymbol{y}), N(\boldsymbol{y}, \boldsymbol{x})\} \geq 1$.

The connection between ECU codes and skew codes is given in the following:

Lemma 4.16 Let $t_1, t_2 \in \mathbb{N}, t = \min\{t_1, t_2\}$.
i.) Let $\mathcal{C}$ be an ECU with minimum distance $d \geq t_1 + t_2 + 1$. Then $\mathcal{C}_1$ is (t_1, t_2)-SD.
ii.) Let $\mathcal{C}$ be an ECU with minimum distance $d \geq t_1 + t_2 + t + 1$. Then $\mathcal{C}_2$ is (t_1, t_2)-ST.

Proof: Let $t = \min\{t_1, t_2\}$ and $T = \max\{t_1, t_2\}$.
i.) Assume that for $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}_1$ the condition a.) of a code to be (t_1, t_2)-SD is violated, say $N(\boldsymbol{x}, \boldsymbol{y}) \leq t$. The code words are unordered, and also

$$\begin{aligned} N(\boldsymbol{y}, \boldsymbol{x}) &= d_H(\boldsymbol{x}, \boldsymbol{y}) - N(\boldsymbol{x}, \boldsymbol{y}) \\ &\geq t_1 + t_2 + 1 - t = T + 1. \end{aligned}$$

Hence, $\boldsymbol{x}$ and $\boldsymbol{y}$ satisfy condition b.) of Theorem 3.9 proving that the code is (t_1, t_2)-SD.
ii.) Assume that for $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}_2$ the condition a.) is violated, say, $N(\boldsymbol{x}, \boldsymbol{y}) \leq t$. The code words are unordered, and also

$$\begin{aligned} N(\boldsymbol{y}, \boldsymbol{x}) &= d_H(\boldsymbol{x}, \boldsymbol{y}) - N(\boldsymbol{x}, \boldsymbol{y}) \\ &\geq t_1 + t_2 + 1. \end{aligned}$$

Hence, $\boldsymbol{x}$ and $\boldsymbol{y}$ satisfy condition b.) of Theorem 3.12 proving that the code is (t_1, t_2)-ST. $\square$

Construction B

Assume one wants to construct an (t_1, t_2)-SD or ST code $\mathcal{C}$ of dimension k. Choose first an $[n', k, d]$ error-correcting code $\mathcal{C}'$, where
$d \geq t_1 + t_2 + 1$ for a code to be (t_1, t_2)-SD and
$d \geq t_1 + t_2 + t + 1$ for a code to be (t_1, t_2)-ST.
Let $\boldsymbol{u}$ be an information vector of length k. Then proceed as follows.

1. Encode $\boldsymbol{u}$ into a vector $\boldsymbol{v} \in \mathcal{C}'$.
2. Let $j \triangleq \omega(\boldsymbol{v})$ be the Hamming weight of $\boldsymbol{v}$. Then append to $\boldsymbol{v}$ the complement of the binary representation of $\lfloor j/d \rfloor$.

Remarks 4.17 i.) The code $\mathcal{C}$ has length $n = n' + \lceil \log_2 \lceil \frac{n'+1}{d} \rceil \rceil$, because there are $n' + 1$ possible Hamming weights for $\boldsymbol{v} \in \mathcal{C}'$.

ii.) The code $\mathcal{C}$ is systematic if the code $\mathcal{C}'$ is systematic.

iii.) The Berger construction corresponds to the special case in which $\mathcal{C}'$ is the $[k, k, 1]$ code.

iv.) One may sometimes make the construction more efficient when the all-1 vector is in $\mathcal{C}'$ by taking a coset of the code. the construction is analogous but one has less than $n' + 1$ different weights in the coset, see [19].

Lemma 4.18 The code $\mathcal{C}$ obtained in Construction B is ECU with minimum distance d.

Proof: Since the construction is based on a $[n', k, d]$ code the code $\mathcal{C}$ has minimum distance d.
It is left to show that the code $\mathcal{C}$ is an antichain. Let $\boldsymbol{v}, \boldsymbol{w} \in \mathcal{C}', \boldsymbol{v} \neq \boldsymbol{w}$ with $\omega(\boldsymbol{v}) = i$ and $\omega(\boldsymbol{w}) = j$, $i \leq j$, then $N(\boldsymbol{w}, \boldsymbol{v}) > 0$. This is clear for the case $i < j$. For the case $i = j$, assume $N(\boldsymbol{w}, \boldsymbol{v}) = 0$; the means that $\boldsymbol{w} \subseteq \boldsymbol{v}$. Since they have the same weight it follows $\boldsymbol{w} = \boldsymbol{v}$, but this is not possible.
Let $\boldsymbol{t}_v$, $\boldsymbol{t}_w$ be the tails of $\boldsymbol{v}$ and $\boldsymbol{w}$ obtained by using Construction B.
Obviously,

$$N((\boldsymbol{w}, \boldsymbol{t}_w), (\boldsymbol{v}, \boldsymbol{t}_v)) > 0.$$

But also it holds

$$N((\boldsymbol{v}, \boldsymbol{t}_v), (\boldsymbol{w}, \boldsymbol{t}_w)) > 0.$$

There are two cases to prove:
First case: $\lfloor \frac{i}{d} \rfloor = \lfloor \frac{j}{d} \rfloor$ and therefore $\boldsymbol{t}_v = \boldsymbol{t}_w$. Then $j - i \leq d - 1$.
Assume $\boldsymbol{v} \subseteq \boldsymbol{w}$ and therefore $(\boldsymbol{v}, \boldsymbol{t}_v) \subseteq (\boldsymbol{w}, \boldsymbol{t}_w)$. Then it holds

$$d_H(\boldsymbol{v}, \boldsymbol{w}) = N(\boldsymbol{v}, \boldsymbol{w}) + N(\boldsymbol{w}, \boldsymbol{v}) = j - i \leq d - 1.$$

This is a contradiction, because d is the minimum distance of $\mathcal{C}$.
Second case: $\lfloor \frac{i}{d} \rfloor \neq \lfloor \frac{j}{d} \rfloor$, and since $i \leq j$ it is $\lfloor \frac{i}{d} \rfloor < \lfloor \frac{j}{d} \rfloor$. Then the binary number of $\boldsymbol{t}_v$ is larger than the binary number of $\boldsymbol{t}_w$. But this means $N(\boldsymbol{t}_v, \boldsymbol{t}_w) > 0$.
Since we have assumed that $N(\boldsymbol{w}, \boldsymbol{v}) > 0$ one finally gets that

$$N((\boldsymbol{v}, \boldsymbol{t}_v), (\boldsymbol{w}, \boldsymbol{t}_w)) > 0.$$

This shows that $\mathcal{C}$ is an antichain. □

In [30] a simple method to obtain ECU codes with minimum distance $d = 3$ is given

<u>Construction B'</u>

Let $(u_0, u_1, \ldots, u_{k-1})$ be an information vector. then proceed as follows:

1. Add a parity bit $u_k = \sum\limits_{i=0}^{k-1} u_i$.

2. Let $\{i_1, i_2, \ldots, i_j\}$ be the set where $(u_0, u_1, \ldots, u_k)$ is zero (i. e., the complement of the support). Let $\boldsymbol{r}$ be the binary representation of $\sum_{l=1}^{j} i_l$. Appending $\boldsymbol{r}$ to $(u_0, u_1, \ldots, u_k)$ completes the encoding.

Remark 4.19 The code $\mathcal{C}$ obtained by the Construction B' has length $n = k = \lceil \log_2 k + \log_2(k+1) \rceil$.

Lemma 4.20 The code $\mathcal{C}$ obtained in Construction B' is ECU with minimum distance 3.

Proof: Consider the code words $\boldsymbol{v}_1 = (\boldsymbol{u}_1, \boldsymbol{r}_1)$ and $\boldsymbol{v}_2 = (\boldsymbol{u}_2, \boldsymbol{r}_2)$, where $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are the information vectors plus the extra parity check bits. In particular, $d_H(\boldsymbol{u}_1, \boldsymbol{u}_2) \geq 2$.
Assume first that $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are unordered. If $d_H(\boldsymbol{u}_1, \boldsymbol{u}_2) > 2$, we are done, so let $d_H(\boldsymbol{u}_1, \boldsymbol{u}_2) = 2$. Since in paricular, $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are also unordered, one has to show that $\boldsymbol{r}_1 \neq \boldsymbol{r}_2$, which will make $d_H(\boldsymbol{v}_1, \boldsymbol{v}_2) \geq 3$.
Since $d_H(\boldsymbol{u}_1, \boldsymbol{u}_2) = 2$ and $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are unordered, let j be the location where $\boldsymbol{u}_1$ is 0 and $\boldsymbol{u}_2$ is 1 and l be the location where $\boldsymbol{u}_1$ is 1 and $\boldsymbol{u}_2$ is 0. Without loss of generality, assume that $j > l$. Therefore, $\boldsymbol{r}_1$ will be larger than $\boldsymbol{r}_2$ as a binary number. In particular, they are different, so $d_H(\boldsymbol{v}_1, \boldsymbol{v}_2) \geq 3$.
Assume next that $\boldsymbol{u}_1$ and$\boldsymbol{u}_2$ are not unordered. Without loss of generality, assume that $\boldsymbol{u}_1 \leq \boldsymbol{u}_2$. In particular, the set of zeros of $\boldsymbol{u}_2$ is contained into the set of zeros of $\boldsymbol{u}_1$, thus $\boldsymbol{r}_1$ is larger than $\boldsymbol{r}_2$ as a binary number. Therefore $\boldsymbol{r}_1$ is not contained in $\boldsymbol{r}_2$, proving that $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are unordered. One can easily see that $d_H(\boldsymbol{v}_1, \boldsymbol{v}_2) \geq 3$. In effect, since $\boldsymbol{u}_1 \leq \boldsymbol{u}_2$, $d_H(\boldsymbol{u}_1, \boldsymbol{u}_2 \geq 2$, since, in particular, $\boldsymbol{r}_1 \neq \boldsymbol{r}_2$, this completes the proof. $\square$

The third construction decribed in [30] has the disadvantage that it is not systematic. The encoding is given by a look-up table, so this construction is not very practical when k is large. Essentially, one reverses the order of Construction B: one first encode the k information symbols into m symbols such that $2^k \leq \binom{m}{\lceil m/2 \rceil}$ and m as small as possible. Then one adds r redundant bits in such a way that one obtains an error-correcting code with minimum distance d. It is clear that the resulting construction gives an ECU code with distance d and length $m + r$.

4.2 Systematic (t_1, t_2)-Skew Codes

In the following one wants to construct a skew code directly without using other codes which have nearly the same properties as the code to be constructed. This has the advantage that one fulfill strictly the deterministic conditions of a skew code.
As done in the previous section one starts by using a t-EC code $\mathcal{C}'$. Then one adds a tail to the encoded vector of $\mathcal{C}'$ to fulfill the conditions a.) and b.) of Theorems 3.9 and 3.12. and finally one unorders the code. In the work of blaum, Bruck and Khachatrian, [7], it is described how to construct such tails. For this, Definitions 4.21 and 4.22 are needed. To the statement of Lemma 4.23 also given in [7] I added the complete proof. Corollary 4.26 states that the Construction C leads to a skew code; therefore Theorem 4.25 is needed.

Definition 4.21 Given two fixed values $m, p \in \mathbb{N}$ one denotes a $(p \times q)$-matrix, where q can be arbitrarily choosen, with rows $\boldsymbol{r}_0, \boldsymbol{r}_1, \dots \boldsymbol{r}_{p-1} \in \{0,1\}^q$ as $R(p, q; m)$, if for any two rows $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$, $0 \leq i < j \leq p-1$ at least one of the following conditions holds:

$$\begin{aligned} &a.) \quad N(\boldsymbol{r}_i, \boldsymbol{r}_j) \geq j - i \\ &\text{or} \\ &b.) \quad N(\boldsymbol{r}_j, \boldsymbol{r}_i) \geq m - (j - i). \end{aligned}$$

Definition 4.22 Let $\boldsymbol{w}_0, \boldsymbol{w}_1, \dots, \boldsymbol{w}_s \in \{0,1\}^s$ be binary vectors defined as follows:

$$\boldsymbol{w}_i = \overbrace{00\dots0}^{i}\overbrace{11\dots1}^{s-i}.$$

Let $p, s \in \mathbb{N}$. A $(p \times s)$-matrix is denoted as $B(p, s)$, if its rows $\boldsymbol{u}_0, \boldsymbol{u}_1, \dots, \boldsymbol{u}_{p-1} \in \{0,1\}^s$ are given by $\boldsymbol{u}_i \triangleq \boldsymbol{w}_j$, where $j = \langle i \rangle_{s+1} \triangleq i \bmod s+1, 0 \leq j \leq s$.

Lemma 4.23 $B(p, s)$ is a $R(p, s; s+1)$

Proof: To prove the statement one has to show that for any two rows of $B(p, s)$, $\boldsymbol{u}_i, \boldsymbol{u}_j$, $0 \leq i < j \leq p-1$ it holds

$$\begin{aligned} &a.) \quad N(\boldsymbol{u}_i, \boldsymbol{u}_j) \geq j - i \\ &\text{or} \\ &b.) \quad N(\boldsymbol{u}_j, \boldsymbol{u}_i) \geq s + 1 - (j - i). \end{aligned}$$

By definition of $B(p,s)$ each row $\boldsymbol{u}_i$, $0 \le i \le p-1$ is of the following kind

$$\boldsymbol{u}_i = \begin{cases} \boldsymbol{w}_0 = (11\ldots1) & \text{if} \quad i = m(s+1) \\ \boldsymbol{w}_1 = (01\ldots1) & \text{if} \quad i = m(s+1)+1 \\ \vdots & \\ \boldsymbol{w}_{s-1} = (0\ldots01) & \text{if} \quad i = m(s+1)+s-1 \\ \boldsymbol{w}_s = (0\ldots00) & \text{if} \quad i = m(s+1)+s \end{cases}, m \in \mathbb{N}.$$

Suppose now that for $0 \le i < j \le p-1$ it holds $i = k(s+1)+c$ and $j = l(s+1)+d$, for $c,d,k,l \in \mathbb{N}$ and $0 \le c,d \le s$. There are three cases to observe: $c = d, c < d$ and $c > d$.

1.) Suppose $c = d$. Then $\boldsymbol{u}_i = \boldsymbol{u}_j = \boldsymbol{w}_c$. This means that $N(\boldsymbol{u}_i, \boldsymbol{u}_j) = N(\boldsymbol{u}_j, \boldsymbol{u}_i) = 0$. Since $i < j$, but $\langle i \rangle_{s+1} = \langle j \rangle_{s+1} = c$ it follows that $k < l$ and therefore $k+1 \le l$ or $k+1-l \le 0$. Now one can see that

$$\begin{aligned} s+1-(j-i) &= s+1-(l(s+1)+c-(k(s+1)+c)) \\ &= s+1-(l-k)(s+1) \\ &= (s+1)(k+1-l) \quad \le 0. \end{aligned}$$

This shows that condition b.) holds, because

$$N(\boldsymbol{u}_j, \boldsymbol{u}_i) = 0 \ge s+1-(j-i).$$

2.) Suppose $c < d$. Then

$$\begin{aligned} \boldsymbol{u}_i &= (\underbrace{0\ldots0}_{c}\underbrace{1\ldots\ldots1}_{s-c}) \\ \boldsymbol{u}_j &= (\underbrace{0\ldots\ldots0}_{d}\underbrace{1\ldots1}_{s-d}) \end{aligned}$$

thereupon $N(\boldsymbol{u}_i, \boldsymbol{u}_j) = d-c$ and $N(\boldsymbol{u}_j, \boldsymbol{u}_i) = 0$.
Since $k(s+1)+c = i < j = l(s+1)+d$ and $c < d$ it follows that $k = l$ or $k < l$.
Provided $k = l$, one can see that

$$j-i \quad = \quad l(s+1)+d-(l(s+1)+c) = d-c.$$

This shows that condition a.) holds, because

$$N(\boldsymbol{u}_i, \boldsymbol{u}_j) = d-c = j-i.$$

Provided $k < l$, one can see that

$$\begin{aligned} s+1-(j-i) &= s+1-(l-k)(s+1)-(d-c) \\ &= \underbrace{(k+1-l)(s+1)}_{\leq 0} - \underbrace{(d-c)}_{>0} \qquad \leq\ 0. \end{aligned}$$

This shows that condition a.) holds, because

$$N(\boldsymbol{u}_j, \boldsymbol{u}_i) = 0 > s+1-(j-i).$$

3.) Suppose $d > c$. Then

$$\begin{aligned} \boldsymbol{u}_i &= (\underbrace{0\ldots\ldots 0}_{c}\underbrace{1\ldots 1}_{s-c}) \\ \boldsymbol{u}_j &= (\underbrace{0\ldots 0}_{d}\underbrace{1\ldots\ldots 1}_{s-d}) \end{aligned}$$

thereupon $N(\boldsymbol{u}_i, \boldsymbol{u}_j) = 0$ and $N(\boldsymbol{u}_j, \boldsymbol{u}_i) = c - d$.
Since $i < j$ it follows that $k < l$ it follows that

$$\begin{aligned} s+1-(j-i) &= (k+1-l)(s+1)-(d-c) \\ &= \underbrace{(k+1-l)(s+1)}_{\leq 0} + c - d \quad \leq\ c-d. \end{aligned}$$

This shows that condition b.) holds, because

$$N(\boldsymbol{u}_j, \boldsymbol{u}_i) = c - d \geq s+1-(j-i).$$

□

In [7] a further construction of $R(p, q; m)$-matrices is given. It is also shown that this method is less efficient for the intention to construct skew codes with small redundancy. I omitted the second method, because for given $p \in \mathbb{N}$ the $B(p, s)$ has always one column less than this. This will translate into saving of one bit in the constructions of skew codes.
The constructions for systematic (t_1, t_2)-SD or ST codes will be given now. The procedure involves adding three tails to the information symbols. The first tail encodes the information bits into an $[n', k, 2t_1 + 2]$ error-correcting code; the second tail makes the code satisfy the conditions of Theorems 3.9 and 3.12; the third tail merely makes the code to be an antichain.

Construction C

a.) Assume one wants to construct a (t_1, t_2)-SD code $\mathcal{C}$ with k information bits. Let $t_1, t_2 \in \mathbb{N}, t_1 < t_2$.
Choose an $[n', k, 2t_1 + 2]$ error-correcting code $\mathcal{C}'$ in which the Hamming weights are at least two apart, i. e. all weight are even. Consider the matrix $B(\lceil \frac{n'-1}{2} \rceil, t_2 - t_1 - 1)$ as given in Definition 4.22.
Let $\boldsymbol{u}$ be an information vector. Then proceed as follows.

1. Encode $\boldsymbol{u}$ into a vector $\boldsymbol{v} \in \mathcal{C}'$.

2. Let $j = \omega(\boldsymbol{v})$. Then append to $\boldsymbol{v}$ row $\boldsymbol{u}_{\lfloor \frac{j}{2} \rfloor}$ of the matrix $B(\lceil \frac{n'-1}{2} \rceil, t_2 - t_1 - 1)$.

3. Append the complement of the binary representation of

$$\begin{cases} \left\lfloor \frac{j}{2t_1+2} \right\rfloor & \text{if} \quad t_1 + 1 \geq t_2 - t_1 \\ \left\lfloor \frac{j}{2(t_2-t_1)} \right\rfloor & \text{if} \quad t_1 + 1 < t_2 - t_1. \end{cases}$$

b.) Assume one wants to construct a (t_1, t_2)-ST code $\mathcal{C}$ with k information bits. Let $t_1, t_2 \in \mathbb{N},\ t_1 \leq t_2$.
Choose an $[n', k, 2t_1 + 2]$ error-correcting code $\mathcal{C}'$ in which the Hamming weights are at least two apart, i. e. all weight are even. Consider the matrix $B(\lceil \frac{n'-1}{2} \rceil, t_2 - 1)$ as given in Definition 4.22.
Let $\boldsymbol{u}$ be an information vector. Then proceed as follows.

1. Encode $\boldsymbol{u}$ into a vector $\boldsymbol{v} \in \mathcal{C}'$.

2. Let $j = \omega(\boldsymbol{v})$. Then append to $\boldsymbol{v}$ row $\boldsymbol{u}_{\lfloor \frac{j}{2} \rfloor}$ of the matrix $B(\lceil \frac{n'-1}{2} \rceil, t_2 - 1)$.

3. Append the complement of the binary representation of

$$\begin{cases} \left\lfloor \frac{j}{2t_1+2} \right\rfloor & \text{if} \quad t_1 = t_2 = t \\ \left\lfloor \frac{j}{2t_2} \right\rfloor & \text{if} \quad t_1 < t_2. \end{cases}$$

Remarks 4.24 a.) Given an $[n', k, 2t_1+2]$ code, Construction C a.) requires an additional $t_2 - t_1 - 1 + \lceil \log_2 \lceil \min\{\frac{n'+1}{2t_1+2}, \frac{n'+1}{2(t_2-t_1)}\}\rceil\rceil$ bits to obtain a (t_1, t_2)-SD code.
b.)Given an $[n', k, 2t_1+2]$ code, Construction C b.) requires an additional $t_2 - 1 + \lceil \log_2 \lceil \min\{\frac{n'+1}{2t_1+2}, \frac{n'+1}{2t_2}\}\rceil\rceil$ bits to obtain a (t_1, t_2)-ST code.

Theorem 4.25 Let $\boldsymbol{x} = (\boldsymbol{v}_1, \boldsymbol{r}_1, \boldsymbol{s}_1)$ and $\boldsymbol{y} = (\boldsymbol{v}_2, \boldsymbol{r}_2, \boldsymbol{s}_2)$ be code words obtained by Construction C a.) with parameters a and b, $a \leq b$. In detail, $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are even weight code words in a $[n', k, 2a+2]$ code; $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are tails corresponding to rows in the $B(\lceil \frac{n'-1}{2} \rceil, b-a-1)$-matrix; $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ are the tails described in the third step of the construction.
Then at least one of the following conditions holds

$$\begin{array}{lll} a.) & \min & \{N(\boldsymbol{x},\boldsymbol{y}), N(\boldsymbol{y},\boldsymbol{x})\} \geq a+1 \\ \text{or} & & \\ b.) & \min & \{N(\boldsymbol{x},\boldsymbol{y}), N(\boldsymbol{y},\boldsymbol{x})\} \geq 1 \\ & \text{and} & \\ & \max & \{N(\boldsymbol{x},\boldsymbol{y}), N(\boldsymbol{y},\boldsymbol{x})\} \geq b+1. \end{array}$$

Proof: First one shows that

$$\min\{N(\boldsymbol{x},\boldsymbol{y}), N(\boldsymbol{y},\boldsymbol{x})\} \geq a+1 \text{ or } \max\{N(\boldsymbol{x},\boldsymbol{y}), N(\boldsymbol{y},\boldsymbol{x})\} \geq b+1.$$

Consider $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, and assume without loss of generality that $\omega(\boldsymbol{v}_1) \leq \omega(\boldsymbol{v}_2)$, moreover, let $\omega(\boldsymbol{v}_2) = \omega(\boldsymbol{v}_1) + 2l$, $l \geq 0$, since the Hamming weights are at least two apart.
Consider the cases where $l \leq a$ and $l \geq a+1$.
Assume first that $0 \leq l \leq a$. Since $d_H(\boldsymbol{v}_1, \boldsymbol{v}_2) \geq 2a+2$ and $N(\boldsymbol{v}_2, \boldsymbol{v}_1) - N(\boldsymbol{v}_1, \boldsymbol{v}_2) = N(\boldsymbol{v}_2, 0) - N(\boldsymbol{v}_1, 0) = 2l$, one obtains

$$\begin{aligned} 2a+2 \leq d_H(\boldsymbol{v}_1, \boldsymbol{v}_2) &= N(\boldsymbol{v}_1, \boldsymbol{v}_2) + N(\boldsymbol{v}_2, \boldsymbol{v}_1) \\ &= N(\boldsymbol{v}_1, \boldsymbol{v}_2) + N(\boldsymbol{v}_1, \boldsymbol{v}_2) + 2l \\ &= 2N(\boldsymbol{v}_1, \boldsymbol{v}_2) + 2l, \end{aligned}$$

and therefore

$$N(\boldsymbol{v}_1, \boldsymbol{v}_2) \geq a+1-l \tag{4.11}$$

and respectively

$$N(\boldsymbol{v}_2, \boldsymbol{v}_1) \geq a+1+l. \tag{4.12}$$

According to Construction C a.), by the choice of $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$, which depend on the weights of $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, it is

$$N(\boldsymbol{r}_1, \boldsymbol{r}_2) \geq l \tag{4.13}$$

or

$$N(\boldsymbol{r}_2, \boldsymbol{r}_1) \geq b - a - l. \tag{4.14}$$

If inequality (4.13) holds, then since (4.11), one gets

$$N(\boldsymbol{x}, \boldsymbol{y}) \geq N(\boldsymbol{v}_1, \boldsymbol{v}_2) + N(\boldsymbol{r}_1, \boldsymbol{r}_2) \geq a + 1. \tag{4.15}$$

Inequality (4.12) implies

$$N(\boldsymbol{y}, \boldsymbol{x}) \geq N(\boldsymbol{v}_2, \boldsymbol{v}_1) \geq a + 1 + l \geq a + 1. \tag{4.16}$$

Therefore (4.15) and (4.16) ensure that condition a.) in Theorem 4.22 holds, provided $N(\boldsymbol{r}_1, \boldsymbol{r}_2) \geq l$.
If inequality (4.14) holds, then by inequality (4.12) one gets

$$N(\boldsymbol{y}, \boldsymbol{x}) \geq N(\boldsymbol{v}_2, \boldsymbol{v}_1) + N(\boldsymbol{r}_2, \boldsymbol{r}_1) \geq b + 1. \tag{4.17}$$

Therefore, the second part of the condition b.) is proven. In the last part of the proof it will be shown that the code is really an antichain.
But first assume the second case that $l \geq a + 1$.
If $2l \geq b + 1$, then condition b.) is satisfied, so assume that $2a + 2 \leq 2l \leq b$. Therefore,

$$N(\boldsymbol{y}, \boldsymbol{x}) \geq N(\boldsymbol{v}_2, \boldsymbol{v}_1) \geq 2l \geq 2a + 2 \geq a + 1. \tag{4.18}$$

As before either (4.13) or (4.14) hold.
If inequality (4.13) holds, then

$$N(\boldsymbol{x}, \boldsymbol{y}) \geq N(\boldsymbol{r}_1, \boldsymbol{r}_2) \geq l \geq a + 1. \tag{4.19}$$

Thus, inequalities (4.18) and (4.19) imply that condition a.) is satisfied.
If inequality (4.14) holds, then

$$\begin{aligned} N(\boldsymbol{y}, \boldsymbol{x}) &\geq N(\boldsymbol{v}_2, \boldsymbol{v}_1) + N(\boldsymbol{r}_2, \boldsymbol{r}_1) \\ &\geq 2l + (b - a - l) = l + b - a \geq b + 1 \end{aligned}$$

so, that the second part of condition b.) is satisfied.

Now one wants to show that the code is an antichain. It is clear that, if the third tail is the complement of the binary representation of $\lfloor \frac{j}{2a+2} \rfloor$, j the weight of the vector $\boldsymbol{v}$ in the error-correcting code, then the third tail makes $\mathcal{C}$ an antichain as shown in the proof of Lemma 4.18.
So, one needs to show that, when the third tail is the complement of the binary representation of $\left\lfloor \frac{j}{2(b-a)} \right\rfloor$, j as above, then this tail unorders the code, too.
To see this, divide the weights corresponding to the error-correcting code into sets of size $b-a$, i. e. the first set involves the weights $0, 2, \ldots, 2(b-a)-2$, the second involves the weights $2(b-a), 2(b-a)+2, \ldots, 4(b-a)-2$, etc. the ith set involves the weights $2(i-1)(b-a), \ldots, 2i(b-a)-2$. Now observe that the second tail unorders each of the sets, since given the ith set to weight $2(b-a)$ corresponds tail $\overbrace{11\ldots1}^{b-a-1}$, to weight $2(i-1)(b-a)+2$ corresponds tail $\overbrace{01\ldots1}^{b-a-1}$, etc.
Finally, notice that the third tail is identical within a set and unorders the different sets. $\square$
Based on the forgoing theorem it is easy to prove the following.

Corollary 4.26 The code obtained with Construction C a.) is (t_1, t_2)-SD, while the code obtained in Construction C b.) is (t_1, t_2)-ST.

Proof: For (t_1, t_2)-SD codes use $a = t_1$ and $b = t_2$ in Theorem 4.22 while for (t_1, t_2)-ST codes use $a = t_1$ and $b = t_1 + t_2$ in Theorem 4.22. $\square$

In [7] there are improvements shown for Construction C by unifying the second and the third tail, and exploiting the weight disribution of the particular code considered with ad hoc methods. By taking a coset of an $[n', k, 2t_1]$ code instead of the code itself one reduces the spead of the weight distribution. This idea to reduce the redundancy of the final code was already done in Section 4.1.1 for the t-EC/AUED code. Since the improvements are based on ad hoc methods I only mention them.

4.3 Examples

Assume one wants to construct a code of dimension $k = 4$ which is

(i) $(1, 5)$-skew-detecting or

(ii) $(1, 5)$-skew-tolerant.

Using all three constructions introduced above, codes which are (i) will be denoted by an index 1 and codes which are (ii) by an index 2.

Construction A: By using this construction one gets a code $\mathcal{C}_1 = \mathcal{C}_2$ that is as well (i) as (ii).
Since $t = \min\{1, 5\} = 1$, one has to construct an 1-EC/AUED. The shortest 1-error-correcting code with dimension 4 is the Hamming code of length $n' = 7$ with parity check matrix

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

Also one needs a descending tail matrix $T(7+1, q; 1+1) = T(8, q; 2)$. Therefore, take

$$T_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Since $s = 4$, the length of $\mathcal{C}_1$ is $n_1 = 7 + 4 = 11$ and the redundancy $r_1 = 7$.

Construction B: (i) To construct a code $\tilde{\mathcal{C}}_1$ the minimum distance d must at least be equal $1 + 5 + 1$. Let $d = 7$ and $\tilde{\mathcal{C}}'_1$ be an $[n', 4, 7]$ code.
By the Griesmer bound one knows that there exists a $[n', 4, 7]$ code, with $n' \geq 14$, and therefore $n = n' + \left\lceil \log_2 \left\lceil \frac{n'+1}{d} \right\rceil \right\rceil \geq 14 + 2 = 16$. So, the $(1, 5)$-SD code $\tilde{\mathcal{C}}_1$ has redundancy $\tilde{r}_1 \geq 12$.

(ii) To construct a code $\tilde{\mathcal{C}}_2$ the minimum distance d must at least be equal $1+5+1+1$. Let $d=8$ and $\tilde{\mathcal{C}'}_2$ be an $[n',4,8]$ code. It follows by the same argument that in this case $n' \geq 15$, and therefore $n \geq 15+1=16$. So, the $(1,5)$-ST code $\tilde{\mathcal{C}}_2$ has redundancy $\tilde{r}_2 \geq 12$.

<u>Construction C</u>: (i) First take the $[8,4,4]$ extended Hamming code $\overline{\mathcal{C}'}$ with parity check matrix

$$H^{ext} = \left(\begin{array}{ccc|c} 1 & \dots & 1 & 1 \\ \hline & & & 0 \\ & H & & \vdots \\ & & & 0 \end{array}\right),$$

where H is as defined in the first example. To the encoded vector $\boldsymbol{v}$ of length $n'=8$ one has to append a row of the matrix

$$B(\lceil (8+1)/2 \rceil, 5-1-1) = B(5,3) = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

Now, since $t_1+1=2<4=t_2-t_1$, the third tail is given by the complement of the binary representation of $\lfloor \frac{j}{2(5-1)} \rfloor$ which is either 0 or 1, because $j \in \{0,\dots 8\}$.
So, the total length of $\overline{\mathcal{C}}_1$ is $n=8+3+1=12$ and the redundancy $\overline{r}_1=8$.

(ii) Take also the extended Hamming code $\overline{\mathcal{C}'}$. To the encoded vector $\boldsymbol{v}$ of length $n'=8$ one has to append a row of the matrix

$$B(\lceil (8+1)/2 \rceil, 5-1) = B(5,4) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Since the maximal Hamming weight is $j=8$ and $\lfloor 8/10 \rfloor = 0$ a third tail is not necessary.
The final length of $\overline{\mathcal{C}}_2$ is $n=8+4$, and the redundancy $\overline{r}_2=8$.
(The value of redundancy deviate from the one calculated in [7] in one bit less.)

4.4 Comparisons

In this section I will compare the different constructions of skew codes described in 4.1 and 4.2 by computing the redundancies.
To my knowledge there has not been a comparison of this type yet. Regarding the literature it is only shown that one can improve a given construction by reducing the redundancy by various methods.
In [19] there are tables of the parameters of the codes constructed by method A and A'; in [30] comparisons of Construction B and B' are presented. In [7] they compare Construction B and C.
All constructions are based upon an error-correcting code $\mathcal{C}'$. To calculate the total redundancy of the final SD or ST code one has to know the length n' of the EC code $\mathcal{C}'$. It has been described above that for each construction a special minimum distance d of the EC code $\mathcal{C}'$ is required. This minimum distance d depends on the values t_1 and t_2. Therefore, for fixed t_1 and t_2 and given dimension k I have used the well known Griesmer bound in order to compute the minimum length n' of an $[n', k, d]$ code $\mathcal{C}'$.
The problem of comparing all constructions is that the code length of a skew code obtained by Construction A, A' and A" cannot easily be calculated. First of all Construction A depends on finding a special tail matrix. Since there is no general rule for constructing these matrices a comparison becomes difficult. For Construction A' it is required that the EC code $\mathcal{C}$ contains the all-one vector. Using the Griesmer bound one can calculate n' which is the smallestst length of a code of dimension k and minimum distance d. However, one has to check if the all-one vector is contained in it.
It turns out to be difficult to decide which construction is the best for a given problem. Some tables are given to illustrate these difficulties.
In Table I and Table IV the parameters to calculate the total redandancy of skew codes are presented, where $t \triangleq \min\{t_1, t_2\}$ and $n' \triangleq \sum_{i=0}^{k-1} \lceil \frac{d}{2^i} \rceil$.
In Tables II, III, V, VI I have calculated the redundancies of some skew-detecting and skew-tolerant codes for given dimension $k = 4, 8$. Since Construction A' depends on the existence of a certain EC code I have marked the values with a star. The shortest redundancy of a code is emphasized.
In Table VII there are given some parameters of descending tail matrices, taken from [19]. I used Table VII to determine the number q of columns which I needed to calculation for the redundancies of A and A'.

Table I Parameters of a (t_1, t_2)-SD Code

	Redundancy	Parameters	Comments
A	$r = n' + q - k$	$d = 2t + 1$	distance of $\mathcal{C}'$
		q	number of columns of the tail matrix $T(p, q; t + 1)$, where p is differently chosen in A, A' and A"
B	$r = n' + l - k$	$d = t_1 + t_2 + 1$	distance of $\mathcal{C}'$
		$l = \lceil \log_2 \lceil \frac{n'+1}{d} \rceil \rceil$	length of the tail
C	$r = n' + b + w - k$	$t_1 < t_2$	requirement
		$d = 2t_1 + 2$	distance of code $\mathcal{C}'$
		$b = t_2 - t_1 - 1$	length of the 1st tail
		$w = \lceil \log_2 \lceil \frac{n'+1}{2t_1+2} \rceil \rceil$ if $t_1 + 1 \geq t_2 - t_1$ $w = \lceil \log_2 \lceil \frac{n'+1}{2(t_2-t_1)} \rceil \rceil$ if $t_1 + 1 < t_2 - t_1$	length of the 2nd tail

Table II Redundancy of (t_1, t_2)-SD Codes of Dimension $k = 4$

	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(2, 4)	(4, 7)
A	7	7	**7**	**7**	**7**	13	13	25
A'	5*	5*	5*	5*	5*	10*	10*	20*
B	**5**	**6**	9	10	12	**10**	12	**20**
C	-	7	8	**7**	8	**10**	**11**	**20**

Table III Redundancy of (t_1, t_2)-SD Codes of Dimension $k = 8$

	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(2, 4)	(4, 7)
A	8	8	8	**8**	**8**	14	14	26
A'	6*	6*	6*	6*	6*	11*	11*	21*
B	**5**	**6**	9	10	12	**10**	12	21
C	-	**6**	**7**	**8**	**8**	**10**	**11**	**20**

TABLE IV PARAMETERS OF A (t_1, t_2)-ST CODE

	REDUNDANCY	PARAMETERS	COMMENTS
A	$r = n' + q - k$	$d = 2t + 1$	distance of $\mathcal{C}'$
		q	number of columns of the tail matrix $T(p, q; t+1)$, where p is differently chosen in A, A' and A"
B	$r = n' + l - k$	$d = t_1 + t_2 + t + 1$	distance of $\mathcal{C}'$
		$l = \lceil \log_2 \lceil \frac{n'+1}{d} \rceil \rceil$	length of the tail
C	$r = n' + b + w - k$	$t_1 \leq t_2$	requirement
		$d = 2t_1 + 2$	distance of $\mathcal{C}'$
		$b = t_2 - 1$	length of the 1st tail
		$w = \lceil \log_2 \lceil \frac{n'+1}{2t_1+2} \rceil \rceil$ if $t_1 = t_2$ $w = \lceil \log_2 \lceil \frac{n'+1}{2t_2} \rceil \rceil$ if $t_1 < t_2$	length of the 2nd tail

TABLE V REDUNDANCY OF (t_1, t_2)-ST CODES OF DIMENSION $k = 4$

	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(2, 4)	(4, 7)
A	7	**7**	**7**	**7**	**7**	13	13	25
A'	5^*	5^*	5^*	5^*	5^*	10^*	10^*	21^*
B	**6**	9	10	12	12	**12**	17	27
C	7	8	**7**	8	8	**12**	**12**	**23**

TABLE VI REDUNDANCY OF (t_1, t_2)-ST CODES OF DIMENSION $k = 8$

	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(2, 4)	(4, 7)
A	8	8	**8**	**8**	**8**	14	14	26
A'	6^*	6^*	6^*	6^*	6^*	11^*	11^*	21^*
B	**6**	9	10	12	13	13	17	28
C	**6**	**7**	**8**	**8**	9	**12**	**13**	**23**

Table vii Parameters of Tail Matrices $T(p, q; t+1)$

t	p	q	t	p	q	t	p	q	t	p	q
1	4	2	2	6	3	3	8	4	4	10	5
1	6	3	2	6	4	3	10	5	4	12	6
1	8	4	2	10	5	3	12	6	4	14	7
1	12	5	2	12	6	3	14	7	4	16	8
1	16	6	2	16	7	3	16	8	4	18	9
1	24	7	2	20	8	3	20	9	4	20	10
1	48	8	2	24	9	3	24	10	4	24	11
1	72	9	2	32	10	3	28	11	4	28	12
1	144	10	2	48	11	3	32	12	4	32	13
1	248	11	2	72	12	3	40	13	4	36	14
1	432	12	2	120	13	3	48	14	4	40	15

Chapter 5

Bounds

In this chapter some bounds for systematic t-EC/AUED codes are stated.
The theoretical lower bounds on the word length n of any $[n, k]$-t-EC/AUED code for given k and t have been discussed in [13], [28] and [32]. In [32] the t-EC/AUED code is regarded as a t-antichain and by using LYM-type inequalities the tightest lower bounds known are derived.
The upper bounds of n depend on the best constructions that have been found. Although the problem has been studied extensively, significant gaps exist between the best known lower and upper bounds for the systematic t-EC/AUED codes. The best known bounds can be found, for instance, in [19], [28] and [32]. The bounds given in [13] refer only to SEC/AUED codes.
Since the t-EC/AUED code is able to correct a certain number of random errors and detects all unidirectional errors, the bounds found out in [25], [26] or [27] are useless. They refer to codes capable of correcting or detecting only one kind of error, i. e. either symmetric, asymmetric or unidirectional errors.
In Section 5.1 I present the lower bounds found out in [28] and [32]. Some of the bounds proved in [32] are improvements of the ones in [28].
The upper bounds stated in the second section of this chapter are also taken from [28]. They refer to the maximum number of code words with special weight conditions.

5.1 Bounds for Systematic t-EC/AUED Codes

5.1.1 Lower Bounds

First I present the results of [28]. Since they are improved by [32] they are given without proof. A t-EC/AUED code of length n, that has cardinality M, will often be described as an (n, M)-t-EC/AUED code. For the proof of Theorem 5.2 one needs the following:

Lemma 5.1 Let $\mathcal{C}$ be a t-EC/AUED code that is equidistant with distance $2t + 2$. Then $\mathcal{C}$ is a constant weight code.

Proof: It follows from the assumption that any two code words, say $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$, satisfy $d_H(\boldsymbol{x}, \boldsymbol{y}) = 2t + 2$, but also $N(\boldsymbol{x}, \boldsymbol{y}) \geq t + 1$ and $N(\boldsymbol{y}, \boldsymbol{x}) \geq t + 1$ and this implies that $N(\boldsymbol{x}, \boldsymbol{y}) = N(\boldsymbol{y}, \boldsymbol{x}) = t + 1$ and thus that $\boldsymbol{x}, \boldsymbol{y}$ have same weight. □

Theorem 5.2 Let $\mathcal{C}$ be a (n, M)-t-EC/AUED code. Let $m \triangleq \lceil M/2 \rceil$. Then

$$mn \geq 2(2m - 1)(t + 1). \tag{5.1}$$

If (5.1) holds with equality, then $\mathcal{C}$ is a constant weight code. If in this case $M \equiv 0 \pmod 4$, then $t + 1$ must be devisible by m.

Proof: One follows the standard derivation of the Plotkin bound. So let A be an $(M \times n)$-matrix with the code words of $\mathcal{C}$ as rows. Let c_i be the number of ones in the ith column of A. Then one has the following two inequalities for S, defined as the sum of the distances between all pairs of code words,

$$\begin{aligned} S &= \sum_{i=1}^{n} c_i(M - c_i) \leq m(M - m)n && (5.2) \\ &\text{and} \\ S &\geq (2t + 2)\binom{M}{2}. && (5.3) \end{aligned}$$

it follows from (5.2) and (5.3) that

$$M(M - 1)(t + 1) \geq m(M - m)n, \tag{5.4}$$

which can be rewritten as (5.1) by substituting $M = 2m$ or $M = 2m - 1$ in (5.4).

Assume that equality holds in (5.1). Then equality also holds in (5.4) and thus also in (5.2) and (5.3). So all pairs of code words have the same distance 2t+2. From Lemma 5.1 one concludes that $\mathcal{C}$ is a constant weight code.
Now assume that $M \equiv 0 \pmod 4$ (so m is even) and that equality in (5.1) holds. Therefore it follows that m devides $2(t+1)$. So, $2(t+1) = vm$ for some integer v. Suppose that v is odd, say $v = 2u+1$. Equality in (5.2) implies that the columns have weight m. Since $\mathcal{C}$ is a constant weight code all code words must have weight $\frac{n}{2}$. But

$$\frac{n}{2} = \frac{(t+1)(2m-1)}{m} = \frac{(2m-1)v}{2} = \frac{(2m-1)(2u+1)}{2},$$

which is not an integer. From this contradiction one concludes that v is even and thus m divides (t+1). $\square$

Next I will give a lower bound on the length n of a systematic t-EC/AUED code. If such a code $\mathcal{C}$ is systematic on k positions, one shall speak of an $[n,k]$-t-EC/AUED code. In the sequel one assumes that $k \geq 2$ and $\mathcal{C}$ is systematic on k positions.
The number $n[k,t]$ denotes the minimal length of all $[n,k]$-t-EC/AUED code. By shortening an $[n,k+1]$-t-EC/AUED code on one of its "systematic" positions one obtains the following inequality:

$$n[k+1,t] \geq n[k,t]+1. \tag{5.5}$$

When $n \leq 4t+4$ Theorem 5.2 provides a good bound on $n[k,t]$. In this case $M = 2^k, k \geq 2$, and $m = 2^{k-1}$. Rewriting (5.1) as

$$n \geq 4t+4 - \left\lfloor \frac{t+1}{2^{k-2}} \right\rfloor,$$

one obtains the following:

Theorem 5.3 The number $n[k,t]$ satisfies

$$n[k,t] \geq \begin{cases} 4t+4-\left\lfloor \frac{t+1}{2^{k-2}} \right\rfloor, & \text{if} \quad t+1 \not\equiv 2^{k-2} (\bmod\ 2^{k-1}) \\ 4t+5-\frac{t+1}{2^{k-2}}, & \text{if} \quad \text{else} \end{cases}.$$

In the following denote a asymmetric error-correcting code by AsEC.

Theorem 5.4 Let $Z(r,\Delta)$ denote the maximum number of code words in any AsEC code of length r and asymmetric distance at least Δ, $1 \leq \Delta \leq t+1$, and let $u = t+1-\Delta$. Then

$$Z(r,\Delta) \geq 2\sum_{i=0}^{u}\binom{k}{i} + \binom{k}{u+1} - \binom{2u+1}{u}. \tag{5.6}$$

Taking $u = t$ in (5.6) one obtains

$$2^r = Z(r,1) \geq 2\sum_{i=0}^{t}\binom{k}{i} + \binom{k}{t+1} - \binom{2t+1}{t}. \tag{5.7}$$

This result can be found in [14]. In [28] the following small improvement of inequality (5.7) was also shown.

Theorem 5.5 Let $\mathcal{C}$ be an $[n,k] - t$-EC/AUED code with redundancy $r = n-k$ and let $k, r \geq 2t+1$. Then

$$2^r \geq 2\sum_{i=0}^{t}\binom{k}{i} + \binom{k}{t+1} - \binom{2t+1}{t} + 2\sum_{i=1}^{t}\binom{r}{i}. \tag{5.8}$$

LYM-type inequalities for t-EC/AUED codes

In [32] Zhang improved the statements of Theorems 5.3 and 5.4. The improvements are derived by using LYM (Lubell, Yamamoto, Meshalkin)-type inequalities. The main idea is that a t-EC/AUED code can be regarded as a t-antichain. For this intention a weak and a strong version of the LYM inequality can be formulated, see [32] and [33]. The consequences for a code holding the condition of a t-antichain are pointed out in [32].
First of all the notions of an antichain and a t-antichain are rendered. Theorem 5.8 states the weak LYM inequality for t-antichains and Theorem 5.13 gives its strong version. The proof of the last is given in the Appendix B. From these two theorems one can derive new lower bounds on the redundancies of systematic t-EC/AUED codes.
In the previous chapters it was often shown that a code capable of detecting all unidirectional errors, obviously, has to be unordered, i. e. such a code is

an antichain in the following notion:
Let P be a partially ordered set with relation "$\leq$". An *antichain* is a subset F of P whose elements are not comparable; that is, if $\boldsymbol{x}, \boldsymbol{y} \in F$ then $\boldsymbol{x} \not\leq \boldsymbol{y}$ and $\boldsymbol{y} \not\leq \boldsymbol{x}$. A *chain* opposed to an antichain is a subset E of P, whose elements are all comparable; that is, if $\boldsymbol{x}, \boldsymbol{y} \in F$ then either $\boldsymbol{x} \leq \boldsymbol{y}$ or $\boldsymbol{y} \leq \boldsymbol{x}$. In what follows, $P = P_n = \{0,1\}^n$ and the relation $\leq$ is defined by $\boldsymbol{x} \leq \boldsymbol{y}$ when $x_i \leq y_i$ for all $i \in [n]$. Notice, that this relation is equivalent to a relation $'' \subseteq''$, if $\boldsymbol{x}$ denotes the support of an n-vector (see chapter 2).
A *maximal chain* in P is a sequence of n-vectors: $0 = \boldsymbol{a}_0 \leq \boldsymbol{a}_1 \leq \boldsymbol{a}_2 \leq \cdots \leq \boldsymbol{a}_n$, where the weight $\omega(\boldsymbol{a}_i)$ of $\boldsymbol{a}_i$ satisfies $\omega(\boldsymbol{a}_i) = i$ for each $0 \leq i \leq n$. Then there are exactly $n!$ maximal chains in P, and exactly $k!(n-k)!$ maximal chains passing a given $\boldsymbol{a}$ of weight k. If F is an antichain in P, then each maximal chain contains at most one member on F. Therefore,

$$\sum_{a \in F} \omega(\boldsymbol{a})!\,(n - \omega(\boldsymbol{a}))! \leq n!\,. \tag{5.9}$$

This gives the following well known LYM inequality for antichain F:

$$\sum_{a \in F} \frac{1}{\binom{n}{\omega(a)}} \leq 1. \tag{5.10}$$

If one denotes by f_k the number on n-vectors in F of weight k, then (5.10) becomes

$$\sum_{k=0}^{n} \frac{f_k}{\binom{n}{k}} \leq 1. \tag{5.11}$$

Because a t-EC/AUED code with $t \geq 1$ satisfies much stronger conditions than an antichain does, (5.10) is usually very weak.
The LYM inequality was sharpended by Ahlswede and Zhang ([34]) to the *Ahlswede-Zhang identity* as follows. For any subset $F \subseteq P$, the A-Z identity is

$$\sum_{a \in P} \frac{W_F(\boldsymbol{a})}{\omega(\boldsymbol{a})\binom{n}{\omega(a)}} \equiv 1, \tag{5.12}$$

$$\text{where } W_F(\boldsymbol{a}) = |\bigwedge_{b \in F.\, b \leq a} \boldsymbol{b}\,|.$$

In the definition of $W_F(\boldsymbol{a})$, $\boldsymbol{a} \wedge \boldsymbol{b} \triangleq \boldsymbol{c}$ is defined by $c_i = \min\{a_i, b_i\}$.

Definition 5.6 A subset $C \subset P$ is called a *t-antichain* if for all distinct $\boldsymbol{a}, \boldsymbol{b} \in C$, $N(\boldsymbol{a}, \boldsymbol{b}) \geq t+1$.

Obviously, a 0-antichain and an antichain are equivalent. Moreover, if C is a non-trivial t-antichain, then for all $\boldsymbol{c} \in C$

$$\omega(\boldsymbol{c}) \geq t+1, n-\omega(\boldsymbol{c}) \geq t+1, \text{ and } n \geq 2t+2. \tag{5.13}$$

Since a t-EC/AUED code fulfills the condition of a t-antichain one can make use of the results of the weak and the strong LYM-type inequalities for t-antichains. These inequalities are two improvements of the standard LYM inequality (5.10).
Let C be a t-antichain in P. Define

$$\mathcal{E}_i(\boldsymbol{c}) \triangleq \{\boldsymbol{x} \in P : \omega(\boldsymbol{x}) = \omega(\boldsymbol{c}) + t - 2i \text{ and } d_H(\boldsymbol{x}, \boldsymbol{c}) \leq t\}, \tag{5.14}$$

for $0 \leq i \leq t$ and $\boldsymbol{c} \in C$.
Clearly, from the definition of $\mathcal{E}_i(\boldsymbol{c})$ and (5.13), one gets

Lemma 5.7 If C is a t-antichain, then for each $0 \leq i \leq t$ and each $\boldsymbol{c} \in C$

$$|\mathcal{E}_i(\boldsymbol{c})| = \sum_{j=0}^{i} \binom{\omega(\boldsymbol{c})}{j} \binom{n-\omega(\boldsymbol{c})}{t-2i+j},$$

$\mathcal{E}_i(\boldsymbol{c}) \cap \mathcal{E}_i(\boldsymbol{c}') = \emptyset$ when $\boldsymbol{c} \neq \boldsymbol{c}'$, and $\mathcal{E}_i(C) \triangleq \bigcup\limits_{c \in C} \mathcal{E}_i(\boldsymbol{c})$ is an antichain.

Define

$$\mathcal{M}_n(m,t,i) \triangleq \sum_{j=0}^{i} \binom{m+t-2i}{t-2i+j} \binom{n-m-t+2i}{j}, \tag{5.15}$$

for $0 \leq i \leq t$ and $m \in \mathbb{N}_0$. Then one has the following result

Theorem 5.8 (Weak LYM inequality for t-antichains)
If C is a t-antichain, then for each $0 \leq i \leq t$

$$\sum_{c \in C} \frac{\mathcal{M}_n(\omega(\boldsymbol{c}), t, i)}{\binom{n}{\omega(c)}} \leq 1.$$

Proof: By Lemma 5.7 , $\mathcal{E}_i(C)$ is an antichain. From (5.14) and (5.10)

$$\sum_{c \in C} \frac{|\mathcal{E}_i(\boldsymbol{c})|}{\binom{n}{\omega(c)+t-2i}} \leq 1.$$

By Lemma 5.7

$$\sum_{c \in C} \frac{\sum_{j=0}^{i} \binom{\omega(c)}{j}\binom{n-\omega(c)}{t-2i+j}}{\binom{n}{\omega(c)+t-2i}} = \sum_{c \in C} \frac{\sum_{j=0}^{i} \binom{\omega(c)+t-2i}{t-2i+j}\binom{n-\omega(c)-t+2i}{j}}{\binom{n}{\omega(c)}} \leq 1. \tag{5.16}$$

Combining (5.15) and (5.16) proves the theorem. □

Since the t-antichain with the n-vectors being the binary complements of the ones in a t-antichain is also a t-antichain, Theorem 5.8 implies the following

Corollary 5.9 If C is a t-antichain, then for $0 \leq i \leq t$

$$\sum_{c \in C} \frac{\mathcal{M}_n(n-\omega(\boldsymbol{c}), t, i)}{\binom{n}{n-\omega(c)}} \leq 1.$$

Now one can present a property of the coefficients in the weak LYM inequality. From (5.15) and Theorem 5.8, one can see that $\mathcal{M}_n(\omega(\boldsymbol{c}), t, i)$ depends on $\omega(\boldsymbol{c}), t$ and i. But when one adds them together for i from 0 to t, the summation turns out to be independent of the n-vector weight $\omega(\boldsymbol{c})$. This can be stated as the following

Theorem 5.10 The following summation

$$\sum_{i=0}^{t} \mathcal{M}_n(m,t,i) = \sum_{i=0}^{t}\sum_{j=0}^{i} \binom{m+t-2i}{t-2i+j}\binom{n-m-t+2i}{j} \quad \triangleq \quad S(n,t),$$

is independent of m when $n-t \geq m \geq t$.

Remark 5.11 From (5.13), $n-t \geq m \geq t$ is satisfied when m is the weight of an n-vector in a non-trivial t-antichain and $S(n,t) > t+1$.

The proof of Theorem 5.10 can be found in the Appendix B.

Corresponding to the basic LYM inequality for antichains one has a stronger result for t-antichains as follows.

Corollary 5.12 If C is a non-trivial antichain, then

$$\sum_{c \in C} \frac{1}{\binom{n}{\omega(c)}} \leq \frac{t+1}{S(n,t)} < 1,$$

where $S(n,t)$ is defined as in Theorem 5.10.

Now it will be shown that the inequality in Theorem 5.8 can be improved by counting the maximal chains passng through $\mathcal{E}_i(C)$ more carefully. The weak LYM inequality is used in the proof of the Lemma B.2 formulated in the Appendix B.3, which plays a key role in the proof of the following strong LYM inequality.
For each $1 \leq i \leq t$, define

$$\begin{aligned} \overline{\mathcal{M}_n}(m,t,0) &= \mathcal{M}_n(m,t,0) \\ \overline{\mathcal{M}_n}(m,t,i) &= \max\{\mathcal{M}_n(m,t,i-1), \mathcal{M}_n(m,t,i)\}, \end{aligned}$$

where $m \in \mathbb{N}_0$ is arbitrary.

Theorem 5.13 (Strong LYM inequality for t-antichains)
If C is a t-antichain, then for each $0 \leq i \leq t$,

$$\sum_{c \in C} \frac{\overline{\mathcal{M}_n}(\omega(\boldsymbol{c}),t,i)}{\binom{n}{\omega(c)}} \leq 1.$$

Similar to Corollary 5.9, one has the following

Corollary 5.14 If C is a t-antichain, then for each $0 \leq i \leq t$,

$$\sum_{c \in C} \frac{\overline{\mathcal{M}_n}(n-\omega(\boldsymbol{c}),t,i)}{\binom{n}{n-\omega(c)}} \leq 1.$$

Now, one can consider t-EC/AUED codes. Let $\mathcal{C}$ be such a code and $r = n-k$ be its redundancy. For any vector $\boldsymbol{c}_1 \in \{0,1\}^k$ there is exactly one $\boldsymbol{c}_2 \in \{0,1\}^r$ such that the concatenation $(\boldsymbol{c}_1, \boldsymbol{c}_2) \in \{0,1\}^n$ is a code word in $\mathcal{C}$ ($\boldsymbol{c}_2$ is the tail of $\boldsymbol{c}_1$). For fixed n, k and r, not all words in $\{0,1\}^r$ can be assigned to words in $\{0,1\}^k$ such that their concatenations are code words in $\mathcal{C}$. In [32] there is an estimation given for the minimum number on this kind of unusable words in $\{0,1\}^r$. Since we are mainly interestet in the

bounds which are consequences from the LYM inequalities I will present this estimation without the proof.
Let

$$C[n,r,t] \triangleq \{\text{all } [n,k]-t-\text{EC/AUED codes with } k=n-r\}. \tag{5.17}$$

A word $\boldsymbol{y} \in \{0,1\}^r$ is called *unused* for an $[n,k]-t-$EC/AUED code $\mathcal{C}$ if there is no $\boldsymbol{x} \in \{0,1\}^k$ such that $\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{C}$. Let

$$U(\mathcal{C}) \triangleq \{\boldsymbol{y} : \boldsymbol{y} \in \{0,1\}^r \text{ is unused for } \mathcal{C}\}. \tag{5.18}$$

Define

$$\delta(n,r,t) \triangleq \min\{|U(\mathcal{C})| : \mathcal{C} \in C[n,r,t\}. \tag{5.19}$$

From this definition, the number of unused words in $\{0,1\}^r$ for any fixed $[n,k]-t-$EC/AUED code with $k=n-r$ is greater than or equal to $\delta(n,r,t)$. In other words, the number of words in $\{0,1\}^r$ that are tails of code words in a fixed $\mathcal{C} \in C[n,r,t]$ is less than or equal to $2^r - \delta(n,r,t)$. Clearly for nonempty $C[n,r,t]$, $r \geq 2t+1$ is required. It is natural to assume $k \geq t$ in this sequel.
The main idea to estimate $\delta(n,r,t)$ is to find a subset of $U(\mathcal{C})$ for any fixed $[n,k]-t-$EC/AUED code $\mathcal{C}$ and to use the cardinality of this subset as a lower bound for $\delta(n,r,t)$.
To estimate the lower bound of the cardinality of $U(\mathcal{C})$ one uses the following

Theorem 5.15

$$|U(\mathcal{C})| \geq \min\{2G(t), G(\lceil r/2 \rceil - t)\}$$

where

$$G(s) = 2^s \left[\binom{r-s}{0} + \cdots \binom{r-s}{t} \right] - 1.$$

The proof can be found in [32]. From Theorem 5.15 the following corollary is straightforward.

Corollary 5.16

$$\delta(n,r,t) \geq \min\{2G(t), G(\lceil r/2 \rceil - t)\}$$

where

$$G(s) = 2^s \left[\binom{r-s}{0} + \cdots \binom{r-s}{t} \right] - 1.$$

Now bounds for systematic t-EC/AUED code deduced from the strong and the weak LYM inequalities will be given. For a code $\mathcal{C} \in C[n, r, t]$. one defines a code $\mathcal{C}_{\boldsymbol{y}}$ for each $\boldsymbol{y} \in \{0,1\}^r$ by letting

$$\mathcal{C}_{\boldsymbol{y}} \triangleq \left\{\boldsymbol{x} \in \{0,1\}^k : (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{C}\right\}. \tag{5.20}$$

If $\boldsymbol{y} \notin U(\mathcal{C})$ (see (5.18)), then $\mathcal{C}_{\boldsymbol{y}}$ is not empty. In this case $\mathcal{C}_{\boldsymbol{y}}$ is also a t-EC/AUED code in $\{0,1\}^k$. For $j = o, 1, \cdots k$, let

$$\Omega_j(k) \triangleq \left\{\boldsymbol{x} \in \{0,1\}^k : \omega(\boldsymbol{x}) = j\right\} \tag{5.21}$$

be the set of all words with length k and weight j, and

$$T_{\mathcal{C}}(\Omega_j(k)) \triangleq \{T_{\mathcal{C}}(\boldsymbol{c}_j) : \boldsymbol{c}_j \in \Omega_j(k)\} \tag{5.22}$$

where $T_{\mathcal{C}}(\boldsymbol{x})$ is the tail of $\boldsymbol{x}$ with respect to $\mathcal{C}$ such that $(\boldsymbol{x}, T_{\mathcal{C}}(\boldsymbol{x})) \in \mathcal{C}$.

One knows that

$$|\{\boldsymbol{y} : \mathcal{C}_{\boldsymbol{y}} = \emptyset\}| \geq \delta(n, r, t),$$

that is, if one defines $Y_r \triangleq \{\boldsymbol{y} : \mathcal{C}_{\boldsymbol{y}} \neq \emptyset\}$, then

$$|Y_r| \leq 2^r - \delta(n, r, t). \tag{5.23}$$

For $\mathcal{C}_{\boldsymbol{y}}$ one has the following

Lemma 5.17 For $\boldsymbol{y} \notin U(\mathcal{C})$, $|\mathcal{C}_{\boldsymbol{y}}| = 1$ if there exists a word $\boldsymbol{a} \in \mathcal{C}_y$ with weight $\omega(\boldsymbol{a})$ satisfying either $0 \leq \omega(\boldsymbol{a}) \leq t$ or $0 \leq k - \omega(\boldsymbol{a}) \leq t$.

From this lemma, one has

$$\bigcup_{0 \leq i \leq t} (\Omega_i(k) \cup \Omega_{k-i}(k)) \subset \bigcup_{|\mathcal{C}_y| = 1} \mathcal{C}_{\boldsymbol{y}}. \tag{5.24}$$

Let

$$Y_r' \triangleq \left\{\boldsymbol{y} \in Y_r : \mathcal{C}_{\boldsymbol{y}} \subset \bigcup_{0 \leq i \leq t} (\Omega_i(k) \cup \Omega_{k-i}(k))\right\}, \tag{5.25}$$

then

$$|Y_r'| = 2 \sum_{i=0}^{t} \binom{k}{i}.$$

Therefore, by (5.24),

$$|Y_r - Y_r'| \le 2^r - \delta(n,r,t) - 2\sum_{i=0}^{t}\binom{k}{i}. \tag{5.26}$$

Applying the strong LYM inequality in Theorem 5.13 to $\mathcal{C}_{\boldsymbol{y}}$ for each $\boldsymbol{y} \in Y_r - Y_r'$, one obtains

$$\sum_{c\in\mathcal{C}_y} \frac{\overline{\mathcal{M}_k}(\omega(\boldsymbol{c}),t,i)}{\binom{k}{\omega(c)}} \le 1,$$

for $i = 0, 1, \cdots t$. This leads to

$$\sum_{y\in Y_r - Y_r'}\sum_{c\in C} \frac{\overline{\mathcal{M}_k}(\omega(\boldsymbol{c}),t,i)}{\binom{k}{\omega(c)}} \le |Y_r - Y_r'|.$$

One knows

$$\bigcup_{y\in Y_r - Y_r'} \mathcal{C}_{\boldsymbol{y}} = \bigcup_{i=t+1}^{k-t-1} \Omega_i(k).$$

Therefore,

$$\sum_{j=t+1}^{k-t-1}\binom{k}{j}\frac{\overline{\mathcal{M}_k}(j,t,i)}{\binom{k}{j}} + 2\sum_{j=0}^{t}\binom{k}{j} \le 2^r - \delta(n,r,t). \tag{5.27}$$

This proves the following result.

Theorem 5.18

$$2^r - \delta(n,r,t) \ge \max_{0\le i\le t}\sum_{j=t+1}^{k-t-1}\overline{\mathcal{M}_k}(j,t,i) + 2\sum_{j=0}^{t}\binom{k}{j},$$

where $\delta(n,r,t)$ satisfies the inequality in Corollary 5.16.

This is an improvement of inequality (5.7), which leads to new lower bounds on $n[k,t]$ for $t = 1$, (see TABLE I, in [32]). For $t \ge 2$ a better lower bound can be given by applying the weak LYM inequality.

For $\mathcal{C} \in C[n,r,t]$, $\mathcal{C}_{\boldsymbol{y}}$ for $\boldsymbol{y} \in \{0,1\}^r$ was defined as the set of words in $\{0,1\}^k$ with tail $\boldsymbol{y}$. Since a nonempty $\mathcal{C}_{\boldsymbol{y}}$ is either a nontrivial or a trivial t-EC/AUED code in $\{0,1\}^k$, one can define $\mathcal{E}_i(\mathcal{C}_{\boldsymbol{y}})$ in $\{0,1\}^k$ as in (5.14) and Lemma 5.7 for $0 \le i \le t$. In the notation $\mathcal{E}_i(\mathcal{C}_{\boldsymbol{y}})$, the parameter t was omitted, but it is essential to indicate $\mathcal{C}_{\boldsymbol{y}}$ being a t-EC/AUED code in the forgoing. Therefore, now one uses $\mathcal{E}_i^t(\mathcal{C}_{\boldsymbol{y}})$ in place of $\mathcal{E}_i(\mathcal{C}_{\boldsymbol{y}})$. The motivation to introduce $\mathcal{E}_i^t(\mathcal{C}_{\boldsymbol{y}})$ before was to use its property of being an antichain. In the following, one will enlarge this antichain by considering the tail words in the neighborhood of $\boldsymbol{y}$ in the redundancy part. To do so, one redefines $\mathcal{E}_i^t(\mathcal{C}_{\boldsymbol{y}})$ in another way. Let

$$\begin{aligned}\mathcal{C}_{\boldsymbol{y}}(\alpha,\beta) \triangleq \{\boldsymbol{x} \in \{0,1\}^k \quad : \quad & \text{it exists } \boldsymbol{c} \in \mathcal{C}_{\boldsymbol{y}} \text{ such that} \\ & N(\boldsymbol{x},\boldsymbol{c}) = \alpha \text{ and } N(\boldsymbol{c},\boldsymbol{x}) = \beta\}\end{aligned}$$

and

$$T_{\boldsymbol{y}}(\alpha,\beta) \triangleq \{\boldsymbol{y}' \in \{0,1\}^r : N(\boldsymbol{y},\boldsymbol{y}') = \alpha \text{ and } N(\boldsymbol{y}',\boldsymbol{y}) = \beta.$$

Then, for $0 \le i \le t$

$$\mathcal{E}_i^t(\mathcal{C}_{\boldsymbol{y}}) = \bigcup_{j=0}^{\min\{i,t-i\}} \mathcal{C}_y(t-i-j, i-j). \tag{5.28}$$

For $0 \le i \le t$, define

$$E_i(\boldsymbol{y}) \triangleq \bigcup_{\substack{0 \le \alpha \le t-i \\ 0 \le \beta \le i}} \bigcup_{y \in T_y(\alpha,\beta)} \left[\bigcup_{j=0}^{\min\{t-i-\alpha, t-\beta} \mathcal{C}_{\boldsymbol{y}}(t-i-\alpha-j, i-\beta-j) \right].$$

For $E_i(\boldsymbol{y})$, one has the following three lemmas which are proven in [32].

Lemma 5.19 For $0 \le i \le t$ and $\boldsymbol{y} \in \{0,1\}^r$, $E_i(\boldsymbol{y})$ is an antichain in $\{0,1\}^k$.

Lemma 5.20 For $\boldsymbol{c}_j \in \Omega_j(k)$, $0 \le j \le t$, let $\boldsymbol{y}_j \triangleq T_{\mathcal{C}}(\boldsymbol{c}_j)$be a tail of $\boldsymbol{c}_j$. Then, $E_i(\boldsymbol{y}_i)$ is empty when $t \ge i \ge \lfloor \frac{t+j}{2} \rfloor + 1$.
If $\boldsymbol{y} \in T_{T_{\mathcal{C}}(0)}(\alpha', \beta')$ and $\alpha' \le t$, then $\mathcal{C}_{\boldsymbol{y}} = \emptyset$.

Lemma 5.21 If $\boldsymbol{y} \in T_{T_{\mathcal{C}}(0)}(\alpha', \beta')$, then $E_i(\boldsymbol{y})$ is empty when $t \ge i \ge \max\{0, \lfloor \frac{t-\alpha'-\beta'}{2} \rfloor + 1\} + \alpha'$.

Applying the basic LYM inequality of the antichains to nonempty $E_i(\boldsymbol{y})$ for all $0 \leq i \leq t$ and all $\boldsymbol{y} \in \{0,1\}^r$, and Lemma 5.20 and Lemma 5.21 one obtains the following theorem which was proved in detail in [32].

Theorem 5.22 If $\mathcal{C}$ is an $[n,k]-t$−EC/AUED code, then

$$\sum_{s=0}^{t} \sum_{t=0} k \binom{r}{s} F_k(l, t-s) \leq (t+1)\, 2^r - D(k,r,t), \tag{5.29}$$

where $r = n-k,\ F_k(m,u) \triangleq \sum_{i=0}^{u} \mathcal{M}(m,u,i)$, and

$$\begin{aligned} D(k,r,t) \ &\triangleq \\ \min_{t \leq d \leq r-t-1} \ &\left\{ \sum_{j=0}^{t} \left\{ 2\left(t - \left\lfloor \frac{t+j}{2} \right\rfloor \right) \binom{k}{j} \right. \right. \\ + \ &\left. \left. \left[\sum_{i=0}^{t-j} \left(t - \left\lfloor \frac{t-i-j}{2} \right\rfloor \right) + \sum_{i=t+1-j}^{t} (t+1-j) \right] \binom{r-d}{j} \binom{d}{i} \right\} \right\}. \end{aligned}$$

5.1.2 Upper Bounds

Definition 5.23 Let $\mathcal{C}$ be an $[n,k]-t$−EC/AUED code that is systematic on the first k positions. Let u be an integer satisfying $0 \leq u \leq t$ and $2u+1 \leq k$. Then $\mathcal{C}^{(u)}$ is defined as the set of code words $\boldsymbol{c} = (\boldsymbol{a}, \boldsymbol{b})$ such that one of the following conditions holds:

1. $k-u \leq \omega(\boldsymbol{a}) \leq u$
2. $\boldsymbol{a}$ starts with i zeros and has exactly u zeros in the remaining $k-i$ positions of $\boldsymbol{a}$, for some i, $1 \leq i \leq k-(2u+1)$

Lemma 5.24 Let $\mathcal{C}$ be an $[n,k]-t$−EC/AUED code with redundancy $r = n-k$. Let $\mathcal{C}^{(u)}$ be defined as in Definition 5.23 and Δ as in (2.9). Let the set of elements of $\{\boldsymbol{b} : (\boldsymbol{a},\boldsymbol{b}) \in \mathcal{C}^{(u)}\}$ be denoted by $\mathcal{C}^{(u)}(r,\Delta,t)$. Then one has the following:

1. It is $t \leq \omega(\boldsymbol{b}) \leq r-t$ for all $\boldsymbol{b} \in \mathcal{C}^{(u)}(r,\Delta,t)$.
2. $|\{\boldsymbol{b} \in \mathcal{C}^{(u)}(r,\Delta,t) : \omega(\boldsymbol{b}) = t\}| \leq 1$ and $|\{\boldsymbol{b} \in \mathcal{C}^{(u)}(r,\Delta,t) : \omega(\boldsymbol{b}) = r-t\}| \leq 1$ (corresponding with $\boldsymbol{a} = \mathbf{1}$, resp. $\boldsymbol{a} = \mathbf{0}$).

Proof: One knows that word $\boldsymbol{c}_0 = (\mathbf{0}, \boldsymbol{b}_0)$ and $\boldsymbol{c}_1 = (\mathbf{1}, \boldsymbol{b}_1)$ exist in $\mathcal{C}^{(u)}$, for some value $\boldsymbol{b}_0$ and $\boldsymbol{b}_1$. Therefore each vector $\boldsymbol{b} \in \mathcal{C}^{(u)}(r, \Delta, t)$, from a vector $\boldsymbol{c} = (\boldsymbol{a}, \boldsymbol{b})$, $\boldsymbol{c} \neq \boldsymbol{c}_0$ or $\boldsymbol{c} \neq \boldsymbol{c}_1$, must have at least $t+1$ zeros and at least $t+1$ ones, since $N(\boldsymbol{c}_0, \boldsymbol{c}) \geq t+1$ and $N(\boldsymbol{c}, \boldsymbol{c}_1) \geq t+1$ must hold. Furthermore $\boldsymbol{b}_0$ must have at least t zeros, since $(\boldsymbol{e}_1, \boldsymbol{b})$ with $\boldsymbol{e}_1 = (1, 0, \cdots, 0)$ is in $\mathcal{C}(u)$ and similary $\boldsymbol{b}_1$ must have at least t ones. □

To obtain powerful bounds from Lemma 5.24 , one can derive two theorems, the first of which can be found in [29]. Let $A(n, d, w)$ denote the maximum number of words of length n, weight w and mutual distance d.

Theorem 5.25 Let $A_i^{(u)}(r, \Delta, t)$ be the number of code words of weight i in $\mathcal{C}^{(u)}(r, \Delta, t)$, $\Delta > 1$, as defined in Lemma 5.24 Then for any w and any $s, 0 \leq s \leq w$,

$$\sum_{i=w-s}^{w} A_i^{(u)}(r, \Delta, t) \leq A(r+s, 2\Delta, w).$$

Proof: To every code word $\boldsymbol{b}$ of $\mathcal{C}^{(u)}(r, \Delta, t)$ of weight in between $w-s$ and w one juxtaposes a vector $\boldsymbol{b}'$ of length s in such a way that the length $(r+s)$-vector $\tilde{\boldsymbol{b}} = (\boldsymbol{b}, \boldsymbol{b}')$ has total weight w.
In this way one obtains $\sum_{i=w-s}^{w} A_i^{(u)}(r, \Delta, t)$ vectors of weight w and length $r+s$. Since $\max\{N(\boldsymbol{b}_1, \boldsymbol{b}_2), N(\boldsymbol{b}_2, \boldsymbol{b}_1)\} \geq \Delta$ and the words , previously obtained all have the same weight, one concludes that $N(\tilde{\boldsymbol{b}}_1, \tilde{\boldsymbol{b}}_2) = N(\tilde{\boldsymbol{b}}_2, \tilde{\boldsymbol{b}}_1) \geq \Delta$ for any pair $\tilde{\boldsymbol{b}}_1$ and$\tilde{\boldsymbol{b}}_2$ of them. So their mutal distance is at least 2Δ and their total number is bounded above by $A(r+s, 2\Delta, w)$. □

Theorem 5.26 The numbers B_w defined below give an upper bound for the number of code words of weight less than or equal to w in $\mathcal{C}^{(u)}(r, \Delta, t)$.

$$B_w = \begin{cases} 0 & \text{if} \quad 0 \leq w \leq t-1 \\ 1 & \text{if} \quad w = t \\ F(w, t, r) & \text{if} \quad t+1 \leq w \leq r-t-1 \\ \min\{B_{r-t-1}+1, F(r-t, t, r)\} & \text{if} \quad w = r-t \\ B_{r-t} & \text{if} \quad r-t+1 \leq w \leq r \end{cases}$$

$$\text{where } F(w, t, r) = \min_{0 \leq s \leq w-t} (B_{w-s-1} + A(r+s, 2\Delta, w))$$

Proof: For $w < t$ and $w > r - t$ the previous assertions follow from 1. in Lemma 5.24, while these assertions for $w = t$ and $w = r - t$ follow from 2. in Lemma 5.24
For $t+1 \leq w \leq r-t-1$ one proceeds by induction. So assume that B_{w-s-1} is an upper bound for the number of code words of weight at most $w - s - 1$ in $\mathcal{C}^{(u)}(r, \Delta, t)$, where $0 \leq s \leq w - t$. Then $B_{w-s-1} + \sum_{i=w=s}^{w} A_i^{(u)}(r, \Delta, t)$ is an upper bound for the number of code words of weight at most w in $\mathcal{C}^{(u)}(r, \Delta, t)$. By Theorem 5.25 $B_{w-s-1} + A(r + s, 2\Delta, w)$ is also an upper bound for this number. $\square$

Chapter 6

Optimality Issues

In this chapter, it will be shown that Construction B and Construction C are in a certain sense optimal. This means the following:
The constructions are based on error-correcting codes to which one resp. two tails are added in such a way that the final code filfills the required conditions. Now, one can prove that these tails are the shortest ones that can be added to the information bits. Of course, this does not mean that the code constructed by B and C are globally optimal.
First one will see that the construction for the error-correcting unordered code is optimal in the described sense if it is based on either an Hamming code or on certain BCH codes. The results are taken from [6] resp. [30]
In the second section the proof is given that the second and the third tail of Construction C are the shortest possible. This was proved in [7].

6.1 For ECU Codes

In this section, the optimality of Construction B for extended Hamming codes and for certain BCH codes will be proved in the following sense: the tail added to the error-correcting code has minimal length, i. e. it is impossible to find a shorter tail making the code unordered.
The idea in proving the optimality of the construction is to exhibit a long enough chain of code words in the error-correcting code.

Lemma 6.1 Let $\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_m\}$ be a chain of vectors, each being a code word in a given code $\mathcal{C}$. Then the length of the tail that one has to add to $\mathcal{C}$ to make it unordered is at least $\lfloor \log_2 m \rfloor$ bits.

Proof: Since all code words in the chain are ordered, one needs to have a different tail for each one of them to make them unordered. Hence, one needs at least m different tails. $\Box$

Now one proves the optimality of some of the Constructions B by exhibiting chains of length $\lceil n/d \rceil + 1$ in an $[n, k, d]$ code. First one proves the optimality of the construction for the extended Hamming code by exhibiting a chain of $2^{m-2} + 1$ code words in a code of length 2^m.

Proposition 6.2 The $[2^m, 2^m - m - 1, 4]$ extended Hamming code contains a chain of $2^{m-2} + 1$ code words.

Proof: The columns of the parity check matrix of a $[2^m, 2^m - m - 1, 4]$ extended Hamming code are:

$$\{(v_1, v_2, \ldots, v_m, 1)^T \,:\, (v_1, v_2, \ldots, v_m) \in \{0,1\}^m\}.$$

Note, that one can arrange the columns in the parity check matrix in pairs such that the first m bits are complementary. Namely, column $(v_1, v_2, \ldots, v_m, 1)^T$ is paired with $(\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_m, 1)$. Hence, the sum of a pair of columns in this arrangement gives the vector $(1, 1, 1, \ldots, 1, 0)^T$ and the sum of two pairs (four columns) is the all-0 vector. Denote this matrix H_m. For example

$$H_3 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Consider the extended Hamming code that corresponds to the matrix H_m. It follows from the construction of H_m that the following set of $2^{m-2}+1$ code words is a chain:

$$\{(1111)^i 0^{2m-4i} \,:\, 0 \le i \le 2^{m-2}\}$$

where $\boldsymbol{s}^i$, $\boldsymbol{s}$ a binary vector, is a vector obtained by concatenating $\boldsymbol{s}$ i times. $\Box$

The second result is related to BCH codes. One can prove that in many cases one can exhibit chains of the code words that show the optimality of Construction B. The key to exhibiting long chains is the following

Lemma 6.3 Consider a binary t-error-correcting BCH code defined in a standard way, i. e. as a cyclic code of length $2^m - 1$. Let a and b be two integers such that

$$a \cdot b = 2^m - 1$$

and

$$a \geq 2t + 1.$$

Then the following b polynomials correspond to code words:

$$z_1(X) = 1 + X^b + X^{2b} + \cdots + X^{(a-1)b}$$

and for $2 \leq i \leq b$,

$$z_i(X) = X^{i-1} z_1(X).$$

The proof can be found in [31].

Proposition 6.4 Given a t-error-correcting BCH code of length $2^m - 1 = a \cdot b$ where a and b are integers, and $a \geq 2t + 1$, one can exhibit a chain of length $b + 1$.

Proof: The proof follows from Lemma 6.3. The chain consists of the all-0 vector and the set of b vectors that correspond to partial sums of the polynomials from Lemma 6.3 as follows:

$$\left\{ \sum_{i=1}^{j} z_i \ : \ 1 \leq j \leq b \right\}.$$

$\square$

6.2 For Systematic Skew Codes

In this section one wants to show that the Construction C is optimal in the sense that the second and the third tail, i. e. steps 2. and 3. are the shortest possible. If the construction is globally optimal is an open problem.
First one shows that the smallest number of columns that a $T(p, q; s)$-matrix may have is $s - 1$.

Theorem 6.5 Let $p, m \in \mathbb{N}$, $p > m$ and let T be a $T(p, s; m)$-matrix, i. e. the matrix has s columns and p rows. Then $s \geq m - 1$.

Proof: Let $\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{p-1}$ be the rows of T. Assume that $s \leq m-2$ and reach a contradiction.
Notice that by Definition 4.17, for $0 \leq i \leq p-2$ it holds

$$\text{either } N(\boldsymbol{u}_i, \boldsymbol{u}_{i+1}) \geq 1 \text{ or } N(\boldsymbol{u}_{i+1}, \boldsymbol{u}_i) \geq m-1.$$

Since $N(\boldsymbol{u}_{i+1}, \boldsymbol{u}_i) \geq m-1 > m-2 \geq s$ is a contradiction, it must be that

$$N(\boldsymbol{u}_i, \boldsymbol{u}_{i+1}) \geq 1 \text{ for } 0 \leq i \leq p-2. \tag{6.1}$$

Claim:

$$N(\boldsymbol{u}_0, \boldsymbol{u}_i) \geq i \text{ for } 0 \leq i \leq m-2. \tag{6.2}$$

Since $m-2 < p-2$, this is certainly true for $i = 1$, by (6.1).
Assume that for $2 \leq j \leq m-2$, (6.2) does not hold. Moreover, let j be the first integer with this property. Therefore,

$$N(\boldsymbol{u}_0, \boldsymbol{u}_{j-1}) \geq j-1 \tag{6.3}$$

and by Definition 4.17

$$N(\boldsymbol{u}_j, \boldsymbol{u}_0) \geq m-j. \tag{6.4}$$

From (6.3), one concludes that

$$\omega(\boldsymbol{u}_0) \geq j-1. \tag{6.5}$$

From (6.4), one concludes that the number of zeros in $\boldsymbol{u}_0$, i. e. $s - \omega(\boldsymbol{u}_0)$, is at least $m-j$. Therefore, $\omega(\boldsymbol{u}_0) \leq s-m+j$, and since $s \leq m-2$, one gets

$$\omega(\boldsymbol{u}_0) \leq j-2. \tag{6.6}$$

But (6.6) and (6.7) contradict each other, so (6.2) is true.
In particular, $N(\boldsymbol{u}_0, \boldsymbol{u}_{m-2}) \geq m-2$, and since each row $\boldsymbol{u}_i$ has length at least $m-2$, $\boldsymbol{u}_{m-2}$ is the all-zero vector.
On the other hand, by (6.1), $N(\boldsymbol{u}_{m-2}, \boldsymbol{u}_{m-1}) \geq 1$, so one reaches a contradiction.□

The following lemma describes a method for optimal unordering of error-correcting codes, i. e. to make a code an antichain. Remember therefore that a set of binary vectors $\mathcal{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m\}$ is called a *chain* of length m if $\boldsymbol{v}_1 \subset \boldsymbol{v}_2 \subset \ldots \subset \boldsymbol{v}_m$, i. e. $N(\boldsymbol{v}_i, \boldsymbol{v}_{i+1}) = 0$ for $1 \leq i \leq m$.

Lemma 6.6 Let $\mathcal{C}$ be a binary code of length n, and the maximal length of the chain in $\mathcal{C}$. Then

1. The tail one needs to append to the code words of $\mathcal{C}$ to make $\mathcal{C}$ an antichain is of length at least $\lceil \log_2 m \rceil$ bits.

2. Code $\mathcal{C}$ can be made an antichain by appending to each code word a tail of length $\lceil \log_2 m \rceil$.

Proof: The lower bound for the length of the tail is obvious. Since m is the maximal length of a chain in $\mathcal{C}$, one needs at least m different tails to append to each of the code words of the chain. The other code words are already unordered so they may have any tail used for one of the code words of the chain. Therefore, the number of bits one has to add is at least $r \triangleq \lceil \log_2 m \rceil$.

The second part of the proof is to show that the code $\mathcal{C}$ can be made an antichain by appending a tail of length r to each code word.
Let $\mathcal{W} = \{\omega_1, \omega_2, \ldots, \omega_l\}$ be the set of possible weights of code words of $\mathcal{C}$. Define subsets of $\mathcal{C}$ by $\mathcal{C}_{\omega_i} \triangleq \{\boldsymbol{x} \in \mathcal{C} : \omega(\boldsymbol{x}) = \omega_i\}$, $1 \leq i \leq l$, and denote the binary representation of j of length r by σ_j.
First, one gives an assignment of the tails σ_i to each code word of $\mathcal{C}$ so that $\mathcal{C}' \triangleq \{(\boldsymbol{x}, \sigma_i) : \boldsymbol{x} \in \mathcal{C}\}$ becomes an antichain; and second, one demonstrates that at most m different tails, i. e. $\sigma_0, \ldots, \sigma_{m-1}$ suffice to realize this goal.
For the assignment of tail proceed as follows:

1. Assign tail $\sigma_0 = \overbrace{(00\ldots0)}^{r}$ to each code word in $\mathcal{C}_{\omega_l}$ (the code word with the highest number of ones).

2. Assume that sets $\mathcal{C}_{\omega_l}, \mathcal{C}_{\omega_{l-1}}, \ldots, \mathcal{C}_{\omega_j}$, $j \geq 1$ have been assigned tails, and the last assigned tail is σ_i, $i \leq m-1$.
   If either $j = 1$ (all sets have been assigned tails)
   or $i = m-1$ (exactly m tails have been assigned), then stop.
   Otherwise, assign tail σ_i to the code words in $\mathcal{C}_{\omega_{j-1}}$ that are not contained in any code word that has been assigned tail σ_i, and tail σ_{i+1} to the rest of the code words in $\mathcal{C}_{\omega_{j-1}}$.

With the above described method $\mathcal{C}'$ gets an antichain. In effect, let $(\boldsymbol{x}, \sigma_i)$ and $(\boldsymbol{y}, \sigma_j)$ be any two different of this vectors, and assume that $\omega(\boldsymbol{x}) \geq \omega(\boldsymbol{y})$. Therefore, by construction, $i \leq j$.

If $i < j$, then $N(\sigma_j, \sigma_i) > 0$, and since $N(\boldsymbol{x}, \boldsymbol{y}) > 0$, $(\boldsymbol{x}, \sigma_i)$ and $(\boldsymbol{y}, \sigma_j)$ are unordered.
If $i = j$, then by construction, $\boldsymbol{x}$ and $\boldsymbol{y}$ are unordered.
One completes the proof by showing that the inductive process described above assigns a tail to each code word in $\mathcal{C}$. The number of assign tail is at most m. Assume that some code words have not been assigned tails, and that $\mathcal{C}_{\omega_s}$ is the first set having code words that have not been assigned a tail. In particular, some code words have been assigned tail σ_{m-1} (last tail). One can reach a contradiction by constructing a chain of length $m+1$ in $\mathcal{C}$. Let $\boldsymbol{x}_0 \in \mathcal{C}_{\omega_s}$ be a code word having no assigned tail. Then there exists a code word $\boldsymbol{x}_1 \in \mathcal{C}$ which has assigned tail σ_{m-1} and it holds that $\boldsymbol{x}_0 \subset \boldsymbol{x}_1$. Since the assignment of tails changes only for code words which are contained in one of the code words have been assigned the last given tail, one can construct a chain $\boldsymbol{x}_0 \subset \boldsymbol{x}_1 \ldots \subset \boldsymbol{x}_m \in \mathcal{C}$, contradicting that m is the maximal length of a chain in $\mathcal{C}$. Therefore, each code word in $\mathcal{C}$ has been assigned a tail. $\square$

Appendix A

Decoding Algorithms

A.1 Decoding Algorithm for Skew-Detecting Codes

Let $\hat{Z} = z_1, z_2, z_3, \cdots, z_j, \cdots$ be the received sequence. Then

SET the initial conditions as $\boldsymbol{x} \leftarrow \emptyset$ and $j \leftarrow 0$.
START: SET $j \leftarrow j + 1$.

IF $z_j \in \boldsymbol{x}$, THEN detect an error and stop.
ELSE, SET $\boldsymbol{x} \leftarrow \boldsymbol{x} \cup \{z_j\}$.

IF $\boldsymbol{x} \in \mathcal{C}$, THEN output $\boldsymbol{x}$, set $\boldsymbol{x} = \emptyset$

GOTO START

A.2 Decoding Algorithm for Skew-Tolerant Codes

The following decoding algorithm is an improvement of the one given in [6]. It contains the following two functions. The first is a function of the sequence $\hat{Z}$. It will be observed after including in each step on further item of the sequence if an elimination of a subset produces a code word. By the way one checks if the number of repeated arrivals does not exceed t_2. The second is a function of the decoded word. It must be deleted or labeled in

the sequence. to not disturbing the further decoding.
Notice that the sets A, B and X can contain several equal elements.

Let $\hat{Z} = z_1, z_2, z_3, \cdots, z_j, \cdots$ be the received sequence. Then

SET the initial conditions $j \leftarrow 0$, $X \leftarrow \emptyset$, $B \leftarrow \emptyset$
and $W_l \leftarrow \emptyset$ for $1 \leq l \leq t_1 + t_2$.

START $j \leftarrow \min\{i \in \mathbb{N} : i \geq j+1 \text{ and } z_i \in \hat{Z}\}$.

IF $z_j \in X$ and $|B| = t_2$, THEN declare an uncorrectable error and stop.

ELSE

IF $z_j \in X$, SET $B \leftarrow B \cup \{z_j\}$
ELSE $B \leftarrow B$

SET $X \leftarrow X \cup \{z_j\}$,
$W_l \leftarrow W_{l+1}$ for $1 \leq l \leq t_1 + t_2 - 1$ and $W_{t_1+t_2} \leftarrow \{z_j\}$

FOR $0 \leq i \leq t_2$ and
for $A \subseteq \bigcup_{s=1}^{t_1-1+i} W_{t_2-i+s}$ of size $|A| = i$, DO

IF $X - A \in \mathcal{C}$ for some A, THEN output $X - A$
SET $X - A = \{z_a, z_b, \cdots, z_k\}$, $1 \leq a < b < \cdots < k$
$a, b, \cdots k \in \mathbb{N}$
$m \leftarrow m(X - A; \hat{Z})$ and $i \leftarrow 0$.

REPEAT WHILE $i \leq k$,

IF $i \leq m$, THEN $\hat{Z} \leftarrow (\hat{Z} - \{z_i\})$
ELSE, $0 \leftarrow z_i$

SET $i \leftarrow \min\{s \in \mathbb{N} : s \geq i+1 \text{ and } z_s \in X - A\}$

SET $X \leftarrow \emptyset$, $B \leftarrow (B - (X - A))$
$W_l \leftarrow \emptyset$ for $1 \leq l \leq t_1 + t_2$.

ELSE, GOTO START

Appendix B

Proofs

B.1 Proof of Theorem 5.13

Throughout this chapter, $|\boldsymbol{a}| = \omega(\boldsymbol{a})$ for a vector $\boldsymbol{a}$, where $\omega(\boldsymbol{a})$ is the Hamming weight and as usual $|F|$ denotes the cardinality of a set F.
For $i = 0$, the statement of Theorem 5.13 is just the one of Theorem 5.10. So we only need to prove Theorem 5.13 for $i = 1, 2, \ldots, t$. To do so, for each i with $1 \leq i \leq t$ and each $\boldsymbol{c} \in C$, define

$$\begin{aligned} A_i(\boldsymbol{c}) = \{X : X &\text{ is a maximal chain in } P_n \text{ and} \\ &\text{it exists } \boldsymbol{x} \in \mathcal{E}_i(\boldsymbol{c}) \text{ such that } X \text{ passes } \boldsymbol{x}\}. \end{aligned} \tag{B.1}$$

Since $\mathcal{E}_i(\boldsymbol{c})$ is an antichain for each $i \in \{1, 2, \ldots, t\}$, Lemma 5.7 and the definition of $\mathcal{E}_i(C)$

$$|A_i(\boldsymbol{c})| = |E_i(\boldsymbol{c})|\,(|\boldsymbol{c}| + t - 2i)!\,(n - |\boldsymbol{c}| - t + 2i)! \text{ for } i = 1, 2, \cdots, t. \tag{B.2}$$

Moreover,

$$\left|\bigcup_{c \in C} A_i(\boldsymbol{c})\right| = \sum_{c \in C} |A_i(\boldsymbol{c})|. \tag{B.3}$$

Because $\mathcal{E}_i(C)$ is also an antichain for each i by Lemma 5.7, (B.3) follows from the fact that $A_i(\boldsymbol{c})$ are disjoint for different $\boldsymbol{c} \in C$. In the following we count all maximal chains in

$$\left(\bigcup_{c \in C} A_{i-1}(\boldsymbol{c})\right) \cup \left(\bigcup_{c \in C} A_i(\boldsymbol{c})\right) \text{ for } i = 1, 2, \ldots, t.$$

Since $\mathcal{E}_{i-1}(C) \cup \mathcal{E}_i(C)$ is no longer an antichain, generally

$$\left|\left(\bigcup_{c\in C} A_{i-1}(\boldsymbol{c})\right) \cup \left(\bigcup_{c\in C} A_i(\boldsymbol{c})\right)\right| \neq \left|\bigcup_{c\in C} A_{i-1}(\boldsymbol{c})\right| + \left|\bigcup_{c\in C} A_i(\boldsymbol{c})\right|.$$

For $i = 1, 2, \ldots, t$, by defining

$$B_i(\boldsymbol{c}) \triangleq A_{i-1}(\boldsymbol{c}) \cup \left\{A_i(\boldsymbol{c}) - \bigcup_{c'\in C} A_{i-1}(\boldsymbol{c}')\right\}, \tag{B.4}$$

we obtain

$$\left|\bigcup_{c\in C} B_i(\boldsymbol{c})\right| = \sum_{c\in C} |B_i(\boldsymbol{c})|, \tag{B.5}$$

which follows from the fact that $B_i(\boldsymbol{c})$ are disjoint for different $\boldsymbol{c} \in C$ and fixed i. Therefore,

$$\sum_{c\in C} |B_i(\boldsymbol{c})| \leq n!. \tag{B.6}$$

On the other hand, for each $i \in \{1, 2, \ldots, t\}$,

$$\begin{aligned}
\sum_{c\in C} |B_i(\boldsymbol{c})| &= \sum_{c\in C} \left\{|A_{i-1}(\boldsymbol{c})| + \left|A_i(\boldsymbol{c}) - \bigcup_{c'\in C} A_{i-1}(\boldsymbol{c}')\right|\right\} \\
&= \sum_{c\in C} \left\{|A_{i-1}(\boldsymbol{c})| + |A_i(\boldsymbol{c})| - \left|\bigcup_{c'\in C} (A_i(\boldsymbol{c}) \cap A_{i-1}(\boldsymbol{c}'))\right|\right\} \\
&\overset{(B.3)}{=} \sum_{c\in C} \{|A_{i-1}(\boldsymbol{c})| + |A_i(\boldsymbol{c})| - |A_i(\boldsymbol{c}) \cap A_{i-1}(\boldsymbol{c})| \\
&\qquad - \sum_{c'\in C, c'\neq c} |A_i(\boldsymbol{c}) \cap A_{i-1}(\boldsymbol{c}')|\}.
\end{aligned} \tag{B.7}$$

To estimate

$$|A_i(\boldsymbol{c}) \cap A_{i-1}(\boldsymbol{c})| \text{ and } \sum_{c'\in C, c'\neq c} |A_i(\boldsymbol{c}) \cap A_{i-1}(\boldsymbol{c}')|,$$

we need the following two lemmas to be proved in section B.3.

Lemma B.1 For $i = 1, 2, \ldots, t$ and $\boldsymbol{c} \in C$,

$$|A_i(\boldsymbol{c}) \cap A_{i-1}(\boldsymbol{c})| = |A_{i-1}(\boldsymbol{c})| - n!\, \frac{\binom{|c|+t-2i}{t-i+1}\binom{n-|c|-t+2i-2}{i-1}}{\binom{n}{|c|}}.$$

Lemma B.2 For $i = 1, 2, \ldots, t$ and $\boldsymbol{c} \in C$,

$$\sum_{c' \in C, c' \neq c} |A_i(\boldsymbol{c}) \cap A_{i-1}(\boldsymbol{c}')| \le n!\, \frac{\binom{|c|+t-2i}{t-i+1}\binom{n-|c|-t+2i-2}{i-1}}{\binom{n}{|c|}}.$$

By (B.2) and Lemma 5.7,

$$\begin{aligned}
|A_i(\boldsymbol{c})| &= |\mathcal{E}_i(\boldsymbol{c})|\,(|\boldsymbol{c}| + t - 2i)!\,(n - |\boldsymbol{c}| - t + 2i)! \\
&= \sum_{j=0}^{i} \binom{|\boldsymbol{c}|}{j}\binom{n - |\boldsymbol{c}|}{t - 2i + j}(|\boldsymbol{c}| + t - 2i)!\,(n - |\boldsymbol{c}| - t + 2i)! \\
&= n!\, \frac{\sum_{j=0}^{i} \binom{m+t-2i}{t-2i+j}\binom{n-m-t+2i}{j}}{\binom{n}{|c|}} \\
&= n!\, \frac{\mathcal{M}_n(|c|, t, i)}{\binom{n}{|c|}}. \qquad \text{(B.8)}
\end{aligned}$$

From (B.8), we can see that to prove the theorem we need to prove only

$$|B_i(\boldsymbol{c})| \ge \max\{|A_i(\boldsymbol{c})|, |A_{i-1}(\boldsymbol{c})|\}. \qquad \text{(B.9)}$$

While the inequality $|B_i(\boldsymbol{c})| \ge |A_{i-1}(\boldsymbol{c})|$ is obvious from the definition of $B_i(\boldsymbol{c})$, the inequality $|B_i(\boldsymbol{c})| \ge |A_i(\boldsymbol{c})|$ is derived as follows:

$$\begin{aligned}
&|B_i(\boldsymbol{c})| \\
&= |A_{i-1}(\boldsymbol{c})| + |A_i(\boldsymbol{c})| - |A_i(\boldsymbol{c}) \cap A_{i-1}(\boldsymbol{c})| - \sum_{c' \in C, c' \neq c} |A_i(\boldsymbol{c}) \cap A_{i-1}(\boldsymbol{c}')| \\
&\overset{\text{Lem. B.1}}{=} |A_{i-1}(\boldsymbol{c})| + |A_i(\boldsymbol{c})| - |A_{i-1}(\boldsymbol{c})| + n!\frac{\binom{|c|+t-2i}{t-i+1}\binom{n-|c|-t+2i-2}{i-1}}{\binom{n}{|c|}} \\
&\quad - \sum_{c' \in C, c' \neq c} |A_i(\boldsymbol{c}) \cap A_{i-1}(\boldsymbol{c})| \\
&\overset{\text{Lem. B.2}}{\ge} |A_i(\boldsymbol{c})|.
\end{aligned}$$

Therefore by (B.8) and (B.9) and the definition of $\overline{\mathcal{M}}_n(m,t,i)$,

$$\sum_{c\in C}\frac{\overline{\mathcal{M}}_n(|\boldsymbol{c}|,t,i)}{\binom{n}{|c|}}\leq 1.$$

This proves Theorem 5.13. $\square$

B.2 Proof of Theorem 5.10

In the proof of the theorem, we need the following

Lemma B.3 For any fixed non–negative integers a and t, the following summation

$$\sum_{i=0}^{t}\binom{n-m+ai}{i}\binom{m-ai}{t-i}$$

is independent of m for $n\geq m\geq at$.

Proof: When $a=0$, Lemma B.3 is equal to $\binom{n}{t}$ for $n\geq m$. So we only need to prove the lemma for $a>0$. Let

$$f_a(n,m,t)\triangleq\sum_{i=0}^{t}\binom{n-m+ai}{i}\binom{m-ai}{t-i}.$$

Fixing a, we use induction on t.
When $t=0$, $f_a(n,m,t)=1$ for $n\geq m$. So $f_a(n,m,t)$ is independent of m when $n\geq m\geq at$.
Assume that $f_a(n,m,t')$ is independent of m when $n\geq m\geq at'$ for $0\leq t'\leq t-1$; that there exists a function $g_a(n,t-j)$ of n and $t-j$ such that

$$f_a(n,m,t-j)=g_a(n,t-j),\tag{B.10}$$

where $n\geq m\geq a(t-j)$ and $1\leq j\leq t$.
We now prove that $f_a(n,m,t)$ is also independent of for $n\geq m\geq at$. To do so, we use induction on n. First, when $n=at$, this is clear because there is only one possible m which is at. For $n=at+1$, it is enough to prove that

$$f_a(at+1,at,t)=f_a(at+1,at+1,t).\tag{B.11}$$

In fact,

$$\begin{aligned}
f_a(at+1, at, t) &= \sum_{i=0}^{t} \binom{1+ai}{i}\binom{at-ai}{t-i} = \sum_{i=1}^{t} \binom{1+a(t-i)}{t-i}\binom{ai}{i} \\
&= f_a(at+1, at+11, t).
\end{aligned}$$

Assume that $f_a(n', m, t)$ is independent of m when $n-1 \geq n' \geq m \geq at$; that is, there exists a function $h_a(n', t)$ of n' and t such that

$$f_a(n', m, t) = h_a(n', t), \tag{B.12}$$

where $n-1 \geq n' \geq m \geq at$.
We prove that $f_a(n, m, t)$ is also independent of m when $n \geq m \geq at$. In the case of $n = m = at$ and $n = at+1$, this has been proved. So we assume $n > at+1$.

(i) When $n-2 \geq m \geq at$,

$$\begin{aligned}
f_a(n, m, t) &= \sum_{i=0}^{t} \binom{n-m+ai}{i}\binom{m-ai}{t-i} \\
&= \sum_{i=0}^{t} \binom{n-1-m+ai}{i}\binom{m-ai}{t-i} \\
&\quad + \sum_{i=1}^{t} \binom{n-m+a-1+a(i-1)}{i-1}\binom{m-a-a(i-1)}{t-1-(i-1)} \\
&= f_a(n-1, m, t) + f_a(n-1, m-a, t-1).
\end{aligned} \tag{B.13}$$

By (B.10) and (B.12),

$$f_a(n, m, t) = h_a(n-1, t) + g_a(n-1, t-1). \tag{B.14}$$

(ii) When $n \geq m \geq at+2$,

$$\begin{aligned}
&\sum_{i=0}^{t} \binom{n-m+ai}{i}\binom{m-ai}{t-i} \\
&= \sum_{i=0}^{t} \binom{n-m+ai}{i}\binom{m-ai-1}{t-i} + \sum_{i=0}^{t} \binom{n-m+ai}{i}\binom{m-ai-1}{t-i-1} \\
&= f_a(n-1, m-1, t) + f_a(n-1, m-1, t-1).
\end{aligned} \tag{B.15}$$

By (B.10) and (B.12),

$$f_a(n,m,t) = h_a(n-1,t) + g_a(n-1,t-1). \tag{B.16}$$

Combining (B.14) and (B.16), $f_a(n,m,t) = h_a(n-1,t)+g_a(n-1,t-1)$ for $n \geq m \geq at$. This proves that $f_a(n,m,t)$ is independent of m when $n \geq m \geq at$. By induction, Lemma B.3 is proved.

Moreover, from (B.13) and (B.15),

$$f_a(n,m,t) = f_a(n,t) = f_a(n-1,t) + f_a(n-1,t-1). \tag{B.17}$$

This is a recursive formula for the summation in Lemma B.3.

Proof of Theorem 5.10

Let

$$J(n,m,t) \triangleq \sum_{i=0}^{t}\sum_{j=1}^{i}\binom{m+t-2i}{t-2i+j}\binom{n-m-t+2i}{j}.$$

We use induction on t.
When $t=0$, $J(n,m,t) = 1$ for all $n \geq m \geq 0$. So $J(m,n,t)$ is independent of m when $n \geq m \geq 0$. When $t=1$, for $n-1 \geq m \geq 1$

$$J(n,m,t) = \binom{m+1}{1} + \binom{n-m+1}{1} = n+2.$$

So $J(n,m,t)$ is independent of m when $n-1 \geq m \geq 1$.
Assume that $J(n,m,t)$ is independent of m when $n-t \geq m \geq t$. We prove that $J(n,m,t+2)$ is also independent of m when $n-(t+2) \geq m \geq t+2$.
Let $i' = t-2i$. Then,

$$\begin{aligned}
& J(n,m,t+2) \\
= & \sum_{i' \in \{t+2,t,t-2,\ldots,-t,-t-2\}} \sum_{j=0}^{\frac{t-i'}{2}+1} \binom{m+i'}{i'+j}\binom{n-m-i'}{j} \\
= & \sum_{i' \in \{t+2,-t-2\}} \sum_{j=0}^{\frac{t-i'}{2}} \binom{m+i'}{i'+j}\binom{n-m-i'}{j}
\end{aligned}$$

$$+\sum_{i'\in\{t,t-2,\ldots,-t+2,-t\}}\sum_{j=0}^{\frac{t-i'}{2}}\binom{m+i'}{i'+j}\binom{n-m-i'}{j}$$
$$+\sum_{i'\in\{t+2,t,t-2,\ldots,-t,-t-2\}}\binom{m+i'}{i'+\frac{t-i'}{2}+1}\binom{n-m-i'}{\frac{t-i'}{2}+1}$$
$$= \quad 0+J(n,m,t)+\sum_{i=0}^{t+2}\binom{m+t-2i+2}{t+2-i}\binom{n-m-t+2i-2}{i}.$$

When $n-(t+2)\geq m\geq t+2$, we have $n-t\geq m\geq t$ and $n\geq m+t+2\geq 2(t+2)$. Therefore, by the hypothesis of the induction and Lemma B.3 with $a\triangleq 2$, $m\triangleq m+t+2$, $J(n,m,t)$ and

$$\sum_{i=0}^{t+2}\binom{m+t-2i+2}{t+2-i}\binom{n-m-t+2i-2}{i}$$

are independent of m when $n-(t+2)\geq m\geq t+2$. So $J(n,m,t+2)$ is independent of m when $n-(t+2)\geq m\geq t+2$. By induction, Theorem 5.10 is proved. $\square$

B.3 Proofs of Lemma B.1 and Lemma B.2

In the proofs of Lemma B.1 and Lemma B.2, we make the following assumptions. A maximal chain X always means $X=\{\boldsymbol{a}_0,\boldsymbol{a}_1,\ldots,\boldsymbol{a}_n\}$ with $\boldsymbol{a}_0\leq\boldsymbol{a}_1\leq\ldots\leq\boldsymbol{a}_n$ such that $\boldsymbol{a}_i\in P_n$ and $|\boldsymbol{a}_i|=i$ for $i=0,1,\ldots,n$.
Without loss of generality, to prove Lemma B.1 and Lemma B.2 for an n–vector $\boldsymbol{c}$ in C we only need to prove them for $\boldsymbol{c}=\underbrace{11\ldots1}_{|c|}\underbrace{00\ldots0}_{n-|c|}$ with $c_1=c_2=\ldots=c_{|c|}=1$ and $c_{|c|+1}=c_{|c|+2}=\ldots=c_n=0$.

Proof of Lemma B.1

$$\begin{aligned}
&|A_i(\boldsymbol{c})\cap A_{i-1}(\boldsymbol{c})|\\
=\ &\sum_{x\in\mathcal{E}_{i-1}(\boldsymbol{c})}|\{X: X\in A_i(\boldsymbol{c})\text{ and } C\text{ passes }\boldsymbol{x}\}|\\
\stackrel{1}{=}\ &|A_{i-1}(\boldsymbol{c})|-\sum_{x\in\mathcal{E}_{i-1}(\boldsymbol{c})}|\{X: X\text{ is a maximal chain in }P_n, X\notin A_i(\boldsymbol{c})\\
&\qquad\text{and } X\text{ passes }\boldsymbol{x}\}|, \qquad\qquad (B.18)
\end{aligned}$$

where step 1 is because $\mathcal{E}_{i-1}(\boldsymbol{c})$ is an antichain by Lemma 5.7. Let us check what is

$$D(\boldsymbol{x}) \triangleq |\{X : X \text{ is a maximal chain in } P_n, \text{ and } X \text{ passes } \boldsymbol{x}\}|$$

for $\boldsymbol{x} \in \mathcal{E}_{i-1}(\boldsymbol{c})$.
If a maximal chain X passes $\boldsymbol{x} \in \mathcal{E}_{i-1}(\boldsymbol{c})$, then by the definition of $\mathcal{E}_i(\boldsymbol{c})$

$$\boldsymbol{a}_{|c|+t-2i+2} = \boldsymbol{x}. \tag{B.19}$$

Claim 1 If $N(x, c) < t - i + 1$, then $D(\boldsymbol{x}) = 0$.

To prove Claim 1, let X be an arbitrary maximal chain in P_n such that X passes $\boldsymbol{x}$. Thus (B.19) is true. We now prove $X \in A_i(\boldsymbol{c})$. It is enough to show that $\boldsymbol{y} = \boldsymbol{a}_{|c|+t-2i} \in \mathcal{E}_i(\boldsymbol{c})$. To prove this, by the definition of $\mathcal{E}_i(\boldsymbol{c})$ and $|\boldsymbol{y}| = |\boldsymbol{c}| + t - 2i$ we only need to check if $d_H(\boldsymbol{y}, \boldsymbol{c}) \le t$. By (B.19),

$$\begin{aligned} N(\boldsymbol{c}, \boldsymbol{x}) &= N(\boldsymbol{x}, \boldsymbol{c}) + |\boldsymbol{c}| - |\boldsymbol{x}| \\ &= N(\boldsymbol{x}, \boldsymbol{c}) + |\boldsymbol{c}| - |\boldsymbol{c}| + 2i - 2 - t \\ &= N(\boldsymbol{x}, \boldsymbol{c}) + 2i - 2 - t. \end{aligned} \tag{B.20}$$

By $N(\boldsymbol{x}, \boldsymbol{c}) \le t - i$,

$$d_H(\boldsymbol{x}, \boldsymbol{c}) \le 2N(\boldsymbol{x}, \boldsymbol{c}) + 2i - 2 - t \le t - 2. \tag{B.21}$$

Since $N(\boldsymbol{x}, \boldsymbol{y}) = 2$ and $N(\boldsymbol{y}, \boldsymbol{x}) = 0$, we have $N(\boldsymbol{c}, \boldsymbol{y}) = N(\boldsymbol{c}, \boldsymbol{x}) = N(\boldsymbol{c}, \boldsymbol{x}) + 2$, and $N(\boldsymbol{y}, \boldsymbol{c}) \le N(\boldsymbol{y}, \boldsymbol{x}) = N(\boldsymbol{x}, \boldsymbol{c})$. By (B.21), we have $d_H(\boldsymbol{y}, \boldsymbol{c}) = N(\boldsymbol{y}, \boldsymbol{c}) + N(\boldsymbol{c}, \boldsymbol{y}) \le d_H(\boldsymbol{x}, \boldsymbol{c}) + 2 \le t$. Therefore, any maximal chain X passing $\boldsymbol{x} \in \mathcal{E}_{i-1}(\boldsymbol{c})$ passes a vector $\boldsymbol{y} \in \mathcal{E}_i(\boldsymbol{c})$. This proves Claim 1.

Claim 2 If $N(\boldsymbol{x}, \boldsymbol{c}) = t - i + 1$, then

$$D(\boldsymbol{x}) = \binom{|\boldsymbol{c}| - i + 1}{2} 2!\, (|\boldsymbol{c}| + t - 2i)!\, (n - |\boldsymbol{c}| - t + 2i - 2)!\ .$$

For simplicity, we assume that

$$\boldsymbol{x} = \underbrace{00\ldots0}_{i-1}\ \overbrace{11\ldots1}^{|c|-i+1}\ \overbrace{11\ldots1}^{t-i+1}\ \ \underbrace{00\ldots0}_{n-|c|-t+i-1}\ ;$$

that is, $x_1 = x_2 = \ldots = x_{i-1} = 0$, $x_i = x_{i+1} = \ldots = x_{|c|} = x_{|c|+1} = \ldots = x_{|c|+t-i+1} = 1$, and $x_{|c|+t-i+2} = x_{|c|+t-i+3} = \ldots = x_n = 0$. For other $\boldsymbol{x}$, the proof is the same.

Let X be an arbitrary maximal chain in P_n passing $\boldsymbol{x}$ such that $\boldsymbol{a}_l = \boldsymbol{x}$, where $l = |\boldsymbol{x}| + t - 2i + 2$ and $X = \{\boldsymbol{a}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$ with $\boldsymbol{a}_j = (a_{j,1}, a_{j,2}, \ldots, a_{j,n})$ for $j = 1, 2, \ldots, n$.

If $a_{l-1,m} = 0$ or $a_{l-2,m} = 0$ for some $m \in \{|\boldsymbol{c}|+1, |\boldsymbol{c}|+2, \ldots, |\boldsymbol{c}|+t-i+1\}$, then $\boldsymbol{a}_{l-2} \in \mathcal{E}_i(\boldsymbol{c})$ because $d_H(\boldsymbol{a}_{l-2}, \boldsymbol{c}) \leq d_H(\boldsymbol{a}_l, \boldsymbol{c}) - 1 + 1 \leq t$.

If $a_{l-1,m_1} = 0$ for an $m_1 \in \{i, i+1, \ldots, |\boldsymbol{c}|\}$ and $a_{l-2,m_2} = 0$ for an $m_2 \in \{i, i+1, \ldots, |\boldsymbol{c}|\}$ different from m_1, then $\boldsymbol{a}_j \notin \mathcal{E}_i(\boldsymbol{c})$ for any $j \in \{0, 1, 2, \ldots, n\}$. To prove this, we only need to check if $\boldsymbol{a}_{l-2} \in \mathcal{E}_i(\boldsymbol{c})$ because the weight of any vector in $\mathcal{E}_i(\boldsymbol{c})$ is $l-2$. Since $a_{l-1,m_1} = a_{l-2,m_2} = 0$ for two different $m_1, m_2 \in \{i, i+1, \ldots, |\boldsymbol{c}|\}$,

$$N(\boldsymbol{c}, \boldsymbol{a}_{l-2}) = i - 1 + 2 = i + 1 \text{ and } N(\boldsymbol{a}_{l-2}, \boldsymbol{c}) = N(\boldsymbol{a}_l, \boldsymbol{c}) = t - i + 1.$$

Therefore, $d_H(\boldsymbol{c}, \boldsymbol{a}_{l-2}) = t + 2 > 1$. This proves that $\boldsymbol{a}_{l-2} \notin \mathcal{E}_i(\boldsymbol{c})$.

Combining the above two cases for $\boldsymbol{a}_{l-1}$ and $\boldsymbol{a}_{l-2}$, we see $\boldsymbol{a}_{l-2} \notin \mathcal{E}_i(\boldsymbol{c})$ if and only if $a_{l-1,m_1} = a_{l-2,m_2} = 0$ for two different m_1 and m_2 in $\{i, i+1, \ldots, |\boldsymbol{c}|\}$.

We have

$$\binom{|\boldsymbol{c}| - i + 1}{2}$$

different pairs $\{m_1, m_2\} \subset \{i, i+1, \ldots, |\boldsymbol{c}|\}$. Moreover, for different order of m_1 and m_2, $\boldsymbol{a}_{l-1}$ is different. Therefore, there are

$$\binom{|\boldsymbol{c}| - i + 1}{2} 2!$$

different arrangements for $\boldsymbol{a}_{l-1}$ and $\boldsymbol{a}_{l-2}$.

Other $\boldsymbol{a}_j$'s with $j \notin \{l-2, l-1, l\}$ can be selected arbitrarily to make $\{\boldsymbol{a}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$ a maximal chain in P_n. There are $(n - |\boldsymbol{c}| - t + 2i - 2)!$ different arangements for $\{\boldsymbol{a}_{l+1}, \boldsymbol{a}_{l+2}, \ldots, \boldsymbol{a}_n\}$, and $(|\boldsymbol{c}| + t - 2i)!$ different arrangements for $\{\boldsymbol{a}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{l-3}\}$. Therefore, the total number of maximal chains passing $\boldsymbol{x} \in \mathcal{E}_{i-1}(\boldsymbol{c})$ without passing any $\boldsymbol{y} \in \mathcal{E}_i(\boldsymbol{c})$ is

$$\binom{|\boldsymbol{c}| - i + 1}{2} 2!\, (|\boldsymbol{c}| + t - 2i)!\, (n - |\boldsymbol{c}| - t + 2i - 2)!.$$

This proves Claim 2.

Noticing that for $\boldsymbol{x} \in \mathcal{E}_{i-1}(\boldsymbol{c})$ there are only the above two cases in Claim 1 and Claim 2 for $N(\boldsymbol{x}, \boldsymbol{c})$, we have

$$\begin{aligned}
& \sum_{x \in \mathcal{E}_{i-1}(c)} D(\boldsymbol{x}) \\
= & \sum_{x \in \mathcal{E}_{i-1}(c), N(x,c)=t-i+1} \binom{|\boldsymbol{c}|-i+1}{2} 2!\,(|\boldsymbol{c}|+t-2i)!\,(n-|\boldsymbol{c}|-t+2i-2)! \\
\overset{1}{=} & \binom{|c|}{i-1}\binom{n-|c|}{t-i+1}\binom{|\boldsymbol{c}|-i+1}{2} 2!\,(|\boldsymbol{c}|+t-2i)!\,(n-|\boldsymbol{c}|-t+2i-2)! \\
= & n!\,\frac{\binom{n-|c|-t+2i-2u}{i-1}\binom{|c|+t-2i}{t-i+1}}{\binom{n}{|c|}},
\end{aligned} \tag{B.22}$$

where step 1 is because there are

$$\binom{|\boldsymbol{c}|}{i-1}\binom{n-|\boldsymbol{c}|}{t-i+1}$$

different $\boldsymbol{x}$ such that $\boldsymbol{x} \in \mathcal{E}_{i-1}(\boldsymbol{c})$ and $N(\boldsymbol{x}, \boldsymbol{c}) = t - i + 1$. Lemma B.1 is implied by (B.18) and (B.20).

Proof of Lemma B.2 Let

$$E(\boldsymbol{c}) \triangleq \bigcup_{c' \in C, c' \neq c} A_i(\boldsymbol{c}) \cap A_{i-1}(\boldsymbol{c}'),$$

and

$$C(\boldsymbol{c}) \triangleq \{\boldsymbol{c}' : \boldsymbol{c}' \in C, \boldsymbol{c}' \neq \boldsymbol{c} \text{ and } A_i(\boldsymbol{c}) \cap A_{i-1}(\boldsymbol{c}') \neq \varnothing\}.$$

To prove Lemma B.2, we first investigate some properties of $E(\boldsymbol{c})$ and $C(\boldsymbol{c})$.

Claim 3. If $\boldsymbol{c}' \in C(\boldsymbol{c})$, then $N(\boldsymbol{c}, \boldsymbol{c}') = t + 1$.
Proof: Let $X \in A_i(\boldsymbol{c}) \cap A_{i-1}(\boldsymbol{c}')$ pass an $\boldsymbol{x} \in \mathcal{E}_i(\boldsymbol{c})$ and a $\boldsymbol{y} \in \mathcal{E}_{i-1}(\boldsymbol{c}')$. Then

$$t + 1 \leq N(\boldsymbol{c}, \boldsymbol{c}') \leq N(\boldsymbol{c}, \boldsymbol{x}) + N(\boldsymbol{x}, \boldsymbol{y}) + N(\boldsymbol{y}, \boldsymbol{c}') \leq i + t - i + 1 + N(\boldsymbol{x}, \boldsymbol{y}).$$

To prove Claim 3, we only need to prove $N(\boldsymbol{x}, \boldsymbol{y}) = 0$. Since X passes $\boldsymbol{x}$ and $\boldsymbol{y}$, we have either $\boldsymbol{x} \leq \boldsymbol{y}$ or $\boldsymbol{y} \leq \boldsymbol{x}$. To prove $N(\boldsymbol{x}, \boldsymbol{y}) = 0$, it is enough to exclude the case $\boldsymbol{y} \leq \boldsymbol{x}$. If $\boldsymbol{y} \leq \boldsymbol{x}$, then $N(\boldsymbol{y}, \boldsymbol{c}) \leq N(\boldsymbol{y}, \boldsymbol{x}) + N(\boldsymbol{x}, \boldsymbol{c}) = t - i$.

And $\boldsymbol{y} \in \mathcal{E}_{i-1}(\boldsymbol{c})$ implies $N(\boldsymbol{c}', \boldsymbol{y}) \leq i-1$. Thus $N(\boldsymbol{c}, \boldsymbol{c}') = t-1$. This contradicts the fact that $N(\boldsymbol{c}, \boldsymbol{c}') \geq t+1$. Claim 3 is proved.
For convenience, we introduce two notations. For any $\boldsymbol{x} \in \{0,1\}^n$ with $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, define $\tilde{\boldsymbol{x}} = (x_1, x_2, \ldots, x_{|c|})$ and $\hat{\boldsymbol{x}} = (x_{|\boldsymbol{c}|+1}, \ldots, x_n)$. Then $\boldsymbol{x} = (\tilde{\boldsymbol{x}}, \hat{\boldsymbol{x}})$. From the assumption at the beginning of this section, $\tilde{\boldsymbol{c}} = (1,1,\ldots,1)$ and $\hat{\boldsymbol{c}} = (0,0,\ldots,0)$. With this notation, Claim 3 can be rephrased as

$$\text{For all } \boldsymbol{c}' \in C(\boldsymbol{c}), |\tilde{\boldsymbol{c}}'| = |\boldsymbol{c}| - t - 1. \tag{B.23}$$

Claim 4
If $\boldsymbol{x} \in E_i(\boldsymbol{c})$ and $\boldsymbol{y} \in E_{i-1}(\boldsymbol{c}')$ are passed by some $X \in A_i(\boldsymbol{c}) \cap A_{i-1}(\boldsymbol{c}')$, then $\boldsymbol{x} \leq \boldsymbol{y}$, $N(\boldsymbol{c}, \boldsymbol{x}) = N(\tilde{\boldsymbol{c}}, \tilde{\boldsymbol{x}}) = i$, $N(\boldsymbol{y}, \boldsymbol{c}') = N(\boldsymbol{y}, \boldsymbol{c}') = N(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{c}}') = t-i+1$, $|\tilde{\boldsymbol{x}}| = |\boldsymbol{c}| - i$, $\tilde{\boldsymbol{c}}' \leq \tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{y}}$.

Proof: $\boldsymbol{x} \leq \boldsymbol{y}$ has already been proved in the proof of Claim 3. We now prove the rest of Claim 4.
Since $N(\boldsymbol{c}, \boldsymbol{x}) \leq i$, $N(\boldsymbol{y}, \boldsymbol{c}') \leq t-i+1$ and $N(\boldsymbol{c}, \boldsymbol{c}') = t+1$, we have

$$N(\boldsymbol{c}, \boldsymbol{x}) = N(\tilde{\boldsymbol{c}}, \tilde{\boldsymbol{x}}) = i \text{ and } N(\boldsymbol{y}, \boldsymbol{c}') = t-i+1.$$

By $N(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{c}}') \leq t-i+1$, $\tilde{\boldsymbol{x}} \leq \tilde{\boldsymbol{y}}$ and $N(\tilde{\boldsymbol{c}}, \tilde{\boldsymbol{c}}') = t+1$, we have $N(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{c}}') = t-i+1$. It is clear that $|\tilde{\boldsymbol{x}}| = |\boldsymbol{c}| - i$. Next we prove $\tilde{\boldsymbol{c}}' \leq \tilde{\boldsymbol{x}}$. If it is not true, that is, $N(\tilde{\boldsymbol{c}}', \tilde{\boldsymbol{x}}) > 0$, then

$$N(\tilde{\boldsymbol{c}}, \tilde{\boldsymbol{c}}') = N(\tilde{\boldsymbol{c}}, \tilde{\boldsymbol{x}}) + N(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{c}}') - N(\tilde{\boldsymbol{c}}', \tilde{\boldsymbol{x}}) < t+1.$$

This contradicts Claim 3. To prove $\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{y}}$, we see that if $\tilde{\boldsymbol{x}} \neq \tilde{\boldsymbol{y}}$, then $N(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{x}}) > 0$. Let $J \triangleq \{j : \tilde{\boldsymbol{y}}_j = 1 \wedge \tilde{\boldsymbol{x}}_j = 0\}$. Then, by $\tilde{\boldsymbol{c}}' \leq \tilde{\boldsymbol{x}}$, $\forall j \in J$, $\tilde{\boldsymbol{c}}' = 0$. Therefore, $N(\tilde{\boldsymbol{c}}, \tilde{\boldsymbol{c}}') = N(\tilde{\boldsymbol{c}}, \tilde{\boldsymbol{x}}) + N(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{c}}') - N(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{x}}) < t+1$. This also contradicts Claim 3. Claim 4 is proved.

From Claim 4 we have the following consequences. Let $J_{\tilde{\boldsymbol{x}}} \triangleq \{j : \tilde{\boldsymbol{x}}_j = 1\}$. Then, from Claim 4, under the same assumptions, $N(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{c}}') = t-i+1$; that is,

$$|\{j \in J_{\tilde{x}} : \tilde{\boldsymbol{c}}'_j = 0\}| = t-i+1. \tag{B.24}$$

For any $\boldsymbol{u} \in \{0,1\}^n$ and $j \in \{0,1,2,\ldots,t\}$, define

$$E_j(\boldsymbol{u}) \triangleq \{\boldsymbol{v} \in \mathcal{E}_j(\boldsymbol{u}) : d_H(\boldsymbol{u}, \boldsymbol{v}) = t\},$$

and

$$A'_j(\boldsymbol{u}) \triangleq \{X \in A_i(\boldsymbol{u}) : X \text{ passes some } \boldsymbol{v} \in E_j(\boldsymbol{u})\},$$

(compare this with (B.1)). Then Claim 4 says that for any $\boldsymbol{c}' \in C(\boldsymbol{c})$,

$$A_i(\boldsymbol{c}) \cap A_{i-1}(\boldsymbol{c}') = A'_i(\boldsymbol{c}) \cap A'_{i-1}(\boldsymbol{c}'). \tag{B.25}$$

For $\boldsymbol{x} \leq \boldsymbol{y}$, define

$$A(\boldsymbol{x}, \boldsymbol{y}) \triangleq \{\text{all maximal chains passing } \boldsymbol{x} \text{ and } \boldsymbol{y}\}.$$

Then for any two different pairs $\{\boldsymbol{x}_1, \boldsymbol{y}_1\}$ with $\boldsymbol{x}_m \in E_i(\boldsymbol{c})$ and $\boldsymbol{y}_m \in E_{i-1}(\boldsymbol{c}')$ for $m = 1, 2$, we have

$$A(\boldsymbol{x}_1, \boldsymbol{y}_1) \cap A(\boldsymbol{x}_2, \boldsymbol{y}_2) = \varnothing. \tag{B.26}$$

This is because for each fixed j, $E_j(\boldsymbol{c})$ is an antichain. Moreover,

$$|A(\boldsymbol{x}, \boldsymbol{y})| = |\boldsymbol{x}|!\,(n - |\boldsymbol{y}|)!\,(|\boldsymbol{y}| - |\boldsymbol{x}|)!. \tag{B.27}$$

To prove Lemma B.2, we devide $E(\boldsymbol{c})$ into a union of some of its subsets. To do so, we need more notations. For any $\tilde{\boldsymbol{c}}'$ with weight $|\boldsymbol{c}| - t - 1$, let

$$\hat{C}(\tilde{\boldsymbol{c}}') \triangleq \{\hat{\boldsymbol{c}}' : \boldsymbol{c}' = (\tilde{\boldsymbol{c}}', \hat{\boldsymbol{c}}') \in C(\boldsymbol{c})\}.$$

Define

$$\tilde{C}(\boldsymbol{c}, \tilde{\boldsymbol{x}}) \triangleq \{\tilde{\boldsymbol{c}}' : |\tilde{\boldsymbol{c}}'| = |\boldsymbol{c}| - t - 1, \text{ and } \tilde{\boldsymbol{c}}' \leq \tilde{\boldsymbol{x}}\},$$

$$\hat{C}(\tilde{\boldsymbol{x}}) \triangleq \bigcup_{\tilde{c}' \in \tilde{C}(c, \tilde{x})} \hat{C}(\tilde{\boldsymbol{c}}'),$$

and for $j \in J_{\tilde{x}}$,

$$\hat{C}_j(\tilde{\boldsymbol{x}}) \triangleq \bigcup_{\tilde{c}' \in \tilde{C}(c, \tilde{x}), \tilde{c}'_j = 0} \hat{C}(\tilde{\boldsymbol{c}}').$$

The following are two more facts needed in the proof of Lemma 3.

Claim 5 If $\hat{\boldsymbol{c}}' \in \hat{C}(\tilde{\boldsymbol{x}})$, then $|\{j : \hat{\boldsymbol{c}}' \in \hat{C}_j(\tilde{\boldsymbol{x}})\}| = t - i + 1$.
The proof of this claim is straightforward and therefore omitted.

For integers $0 \leq j \leq k$, let

$$\Omega_j(k) \triangleq \{\boldsymbol{x} \in \{0, 1\}^k : |\boldsymbol{x}| = j\}. \tag{B.28}$$

Claim 6 For each $\tilde{\boldsymbol{x}} \in \Omega_{|\boldsymbol{c}|-i}(|\boldsymbol{c}|)$ and each $j \in J_{\tilde{\boldsymbol{x}}}$, $\hat{C}_j(\tilde{\boldsymbol{x}})$ is an i–antichain.

Proof For any two different $\hat{\boldsymbol{c}}'_1$ and $\hat{\boldsymbol{c}}'_2$ in $\hat{C}_j(\tilde{\boldsymbol{x}})$, we need to prove that $N(\hat{\boldsymbol{c}}'_1, \hat{\boldsymbol{c}}'_2) \geq i+1$. Suppose $\hat{\boldsymbol{c}}'_m \in \hat{C}_j(\tilde{\boldsymbol{c}}'_m)$ with $\tilde{\boldsymbol{c}}'_m \in \tilde{C}(\tilde{\boldsymbol{c}}, \tilde{\boldsymbol{x}})$ for $m = 1, 2$. Then, $\tilde{\boldsymbol{c}}'_{1,j} = \tilde{\boldsymbol{c}}'_{2,j} = 0$. Also, by $N(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{c}}'_m) = t - i + 1$ and $\tilde{\boldsymbol{c}}'_m \leq \tilde{\boldsymbol{x}}$ for $m = 1, 2$, we have $N(\tilde{\boldsymbol{c}}'_1, \tilde{\boldsymbol{c}}'_2) \leq t - i$. But, $N(\boldsymbol{c}'_1, \boldsymbol{c}'_2) \geq t + 1$. Therefore, $N(\hat{\boldsymbol{c}}'_1, \hat{\boldsymbol{c}}'_2) = N(\boldsymbol{c}'_1, \boldsymbol{c}'_2) - N(\tilde{\boldsymbol{c}}'_1, \tilde{\boldsymbol{c}}'_2) \geq i + 1$. This proves Claim 6.
Now we are ready to prove Lemma B.2. By (B.23),

$$\begin{aligned}
& E(\boldsymbol{c}) \\
= & \bigcup_{c' \in C, c' \neq c} (A'_i(\boldsymbol{c}) \cap A'_{i-1}(\boldsymbol{c})) \\
\stackrel{1}{=} & \bigcup_{x \in E_i(c)} \bigcup_{c' \in C(c)} \bigcup_{y \in E_{i-1}(c'), x \leq y} A(\boldsymbol{x}, \boldsymbol{y}) \\
\stackrel{2}{=} & \bigcup_{\tilde{x} \in \Omega_{|c|-1}(|c|)} \bigcup_{\hat{x} \in \Omega_{n-|c|}} \bigcup_{\tilde{c}' \in C(c, \tilde{x})} \bigcup_{\hat{c}' \in \hat{C}(\tilde{c}')} \bigcup_{\hat{x} \leq \hat{y} \leq \hat{c}', N(\hat{c}', \hat{y}) = i-1} A(\boldsymbol{x}, \boldsymbol{y}) \\
= & \bigcup_{\tilde{x} \in \Omega_{|c|-1}(|c|)} \bigcup_{\hat{c}' \in \hat{C}(\tilde{x})} \bigcup_{\hat{y} \leq \hat{c}', N(\hat{c}', \hat{y}) = i-1} \bigcup_{\hat{x} \leq \hat{y}, |\hat{x}| = t-i} A(\boldsymbol{x}, \boldsymbol{y}).
\end{aligned}$$

Therefore, by Claim 5 and eq. (B.24)

$$\begin{aligned}
& |E(\boldsymbol{c})| \\
\leq & \sum_{\tilde{x} \in \Omega_{|c|-1}(|c|)} \frac{1}{t-i+1} \sum_{j \in J_{\tilde{c}}} \sum_{\hat{c}' \in \hat{C}_j(\tilde{x})} \sum_{\hat{y} \leq \hat{c}', N(\hat{c}', \hat{y}) = i-1} \sum_{\hat{x} \leq \hat{y}, |\hat{y}| = t-i} |A(\boldsymbol{x}, \boldsymbol{y})| \\
\stackrel{3}{=} & \frac{1}{t-i+1} \sum_{\tilde{x} \in \Omega_{|c|-1}(|c|)} \sum_{j \in J_{\tilde{c}}} \sum_{\hat{c}' \in \hat{C}_j(x)} \binom{|\hat{\boldsymbol{c}}'|}{i-1} \binom{|\hat{\boldsymbol{c}}'| - i + 1}{t - i} \\
& \cdot (|\boldsymbol{c}| + t - 2i)!\,(n - |\boldsymbol{c}| - |\hat{\boldsymbol{c}}'| + 2i - 1)!\,(|\hat{\boldsymbol{c}}'| - t + 1)! \\
\stackrel{4}{\leq} & \frac{(|\boldsymbol{c}| + t - 2i)!}{(t-i+1)!} \sum_{\tilde{\boldsymbol{c}} \in \Omega_{|\boldsymbol{c}|-1}(|\boldsymbol{c}|)} \sum_{j \in J_{\tilde{\boldsymbol{x}}}} \sum_{\hat{\boldsymbol{c}}' \in \hat{C}_j(\tilde{\boldsymbol{x}})} |\hat{\boldsymbol{c}}'|!(n - |\boldsymbol{c}| - |\hat{\boldsymbol{c}}'| + i)! \\
& \cdot \frac{(n - |c| - t + 2i - 2)(n - |c| - t + 2i - 3) \ldots (n - |c| - t + i)}{(i-1)!} \\
= & \frac{(|\boldsymbol{c}| + t - 2i)!}{(t-i+1)!} \binom{n - |\boldsymbol{c}| - t + 2i - 2}{i - 1}
\end{aligned}$$

$$\cdot \sum_{\tilde{x}\in\Omega_{|c|-1}(|c|)} \sum_{j\in J_{\tilde{x}}} \left[\sum_{\hat{c}'\in\hat{C}_j(\tilde{x})} \frac{\binom{n-|c|-|\hat{c}'|+i}{i}}{\binom{n-|c|}{|\hat{c}'|}} \right] \cdot i!\,(n-|\boldsymbol{c}|)!$$

$$\overset{5}{\leq} \frac{(|\boldsymbol{c}|+t-2i)!\,i!\,(n-|\boldsymbol{c}|)!}{(t-i+1)!} \binom{n-|\boldsymbol{c}|-t+2i-2}{i-1} \cdot \sum_{\tilde{x}\in\Omega_{|c|-i}(|c|)} \sum_{j\in J_{\tilde{x}}} 1$$

$$\overset{6}{=} \frac{(|c|+t-2i)!\,i!\,(n-|\boldsymbol{c}|)!}{(t-i+1)!} \binom{n-|\boldsymbol{c}|-t+2i-2}{i-1} \binom{|\boldsymbol{c}|}{i} (|\boldsymbol{c}|-i)$$

$$= n! \binom{|\boldsymbol{c}|+t-2i}{t-i+1} \binom{n-|\boldsymbol{c}|-t+2i-2}{i-1} \frac{1}{\binom{n}{|c|}}.$$

Step 1 is from Claim 4. Step 2 is because from $E_i(\boldsymbol{c}) = \{(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{x}}) : \tilde{\boldsymbol{x}} \in \Omega_{|c|-i}(|\boldsymbol{c}|)$ and $N(\hat{\boldsymbol{x}}, \hat{\boldsymbol{c}}) = t-i\}$, and $\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{y}}$ by Claim 4, we have $\{\boldsymbol{y} \in E_{i-1}(\boldsymbol{c}') : \boldsymbol{x} \leq \boldsymbol{y}\} = \{(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{y}}) : N(\hat{\boldsymbol{c}}', \hat{\boldsymbol{y}}) = i-1 \text{ and } \hat{\boldsymbol{x}} \leq \hat{\boldsymbol{y}}\}$, where $\Omega_{|c|-i}(|\boldsymbol{c}|)$ is defined by (B.26). Step 3 is because

$$|\{\hat{\boldsymbol{x}} : \hat{\boldsymbol{y}} \leq \hat{\boldsymbol{c}}' \text{ and } N(\hat{\boldsymbol{c}}', \hat{\boldsymbol{y}}) = i-1\}| = \binom{|\hat{\boldsymbol{c}}'|}{i-1},$$

and

$$|\{\hat{\boldsymbol{x}} : \hat{\boldsymbol{x}} \leq \hat{\boldsymbol{y}} \text{ and } |\hat{\boldsymbol{x}}| = t-i\}| = \binom{|\hat{\boldsymbol{y}}|}{t-i} = \binom{|\hat{\boldsymbol{c}}'|-i+1}{t-i}.$$

Step 4 is because $|\hat{\boldsymbol{c}}'| \geq t+1$ by $N(\boldsymbol{c}', \boldsymbol{c}) \geq t+1$. From Claim 6, $\hat{C}_j(\tilde{\boldsymbol{x}})$ is an i–antichain, which implies

$$\sum_{\hat{c}'\in\hat{C}_j(\tilde{x})} \frac{\binom{n-|c|-|\hat{c}'|+i}{i}}{\binom{n-|c|}{|\hat{c}'|}} \leq 1$$

by Corollary 1 with i and t in (5.) replaced by 0 and i respectively.
This justifies step 5. Step 6 is because $|\Omega_{|c|-i}(|\boldsymbol{c}|)| = \binom{|c|}{i}$ and $|J_{\tilde{x}}| = |\boldsymbol{c}| - i$. The latter one is from the fact $N(\tilde{\boldsymbol{c}}, \tilde{\boldsymbol{x}}) = i$. This concludes the proof of Lemma B.2. □

Bibliography

[1] T. Verhoeff, "Delay-insensitive codes - an overview," *Distributed Computing*, vol. 3, pp. 1-8, 1988.

[2] J. M. Berger, "A note on error detecting codes for asymmetric channels," *Information and Control*, vol. 4, pp. 68-73, 1961.

[3] H. v. Tilborg, "Error-correcting codes - a first course," *Studentlitteratur*, Lund, 1993.

[4] W. W. Peterson, E. J. Weldon, Jr., "Error-correcting codes," *MIT*, 2nd edition, 1972.

[5] F. J. MacWilliams, N. J. Sloane, "The theory of error correcting codes," *North-Holland*, 8th printing, 1993.

[6] M. Blaum, J. Bruck, "Coding for skew-tolerant parallel asynchronous communications," *IEEE Transactions on Information Theory*, vol. 39, No. 2, pp. 379-388, March 1993.

[7] M. Blaum, J. Bruck, L. H. Khachatrian, "Constructions of skew-tolerant and skew-detecting codes," *IEEE Trans. Inform. Theory*, vol. 39, No. 5, pp. 1751-1757, Sept. 1993.

[8] D. A. Anderson, "Design of self-checking digital networks using coding techniques," *Univ. of Illinois, Urbana, CSL Rep.*, R-527, Oct. 1971.

[9] T. R. N. Rao, A. S. Chawla, "Asymmetric error codes for some LSI semiconductor memories," *Proc. Annu. Southeastern Symp. on Syst. Theory*, pp. 170-171, March 1975.

[10] S. D. Constantin, T. R. N. Rao, "Group theoretic codes for binary asymmetric channels," *Inform. Control*, vol. 40, pp. 20-26, Jan. 1979.

[11] D. K. Pradhan, "A new class of error-correcting/detecting codes for fault-tolerant computer application," *IEEE Trans. Computers*, vol. C-29, pp. 471-481, June 1980.

[12] B. Bose, "On systematic SEC/MUED code," *Proceedings Intern. Sympos. Fault Tolerant Computing*, vol. 11, pp. 265-267, June 1981.

[13] B. Bose, T. R. N. Rao, "Theory of unidirectional error-correcting/detecting codes," *IEEE Trans. Computers*, vol. C-31, pp. 521-530, June 1982.

[14] B. Bose, D. K. Pradhan, "Optimal error detecting/correcting codes," *IEEE Trans. Computers*, vol. C-31, pp. 564-568, June 1982.

[15] N. Nikolos, N. Gaitanis. G. Philokyprou, "Systematic t-error-correcting/all unidirectional error detecting codes," *IEEE Trans. Computers*, vol. C-35, No. 5, pp. 394-402, May 1986.

[16] D. L. Tao, C. R. P. Hartmann, P. K. Lala, "An efficient class of unidirectional error detecting/correcting codes," *IEEE Trans. Computers*, vol. 37, No. 7, pp. 877-882, July 1988.

[17] M. Blaum, H. van Tilborg, "On t-error-correcting/all unidirectional error detecting codes," *IEEE Trans. Computers*, vol. 38, pp. 1493-1501, Nov. 1989.

[18] S.-J. Wang, N. K. Jha, "Systematic t-error correcting/all unidirectional error detecting codes with easy encoding/decoding," *Computers and Mathematics with Applications*, vol. 20, No. 1, pp. 5-13, 1990.

[19] J. Bruck, M. Blaum, "New techniques for constructing EC/AUED codes," *IEEE Trans. Computers*, pp. 1318-1324, October 1992.

[20] Z. Zhang, C. Tu, "On the construction of systematic t-EC/AUED codes,"*IEEE Trans. Inform. Theory*, vol. 39, pp. 1662-1669, Sept. 1993.

[21] S. Al-Bassam, B. Bose, "Asymmetric/unidirectional error correcting and detecting codes," *IEEE Trans. Computers*, vol. 43, No. 5, pp. 590-597, May 1994.

[22] S. Katti, M. Blaum, "An improvement on constructions of t- EC/AUED codes," *IEEE Trans. Computers*, vol. 45, No. 5, pp. 607-608, May 1996.

[23] C.-N. Yang, C.-S. Laih, "On the design of t-EC/AUED codes," *Proceedings IEEE Intern. Sympos. Inform. Theory*, Ulm, Germany, 1997.

[24] S. Al-Bassam, B. Bose, "Design of efficient balanced codes," *IEEE Trans. Comp.*, vol. 43, No. 3, March 1994.

[25] T. Etzion, "New lower bound for asymmetric and unidirectional codes," *IEEE Trans. Inform. Theory*, vol. 37, pp. 1696-1704, Nov. 1991.

[26] J. M. Borden, "Optimal asymmetric error detecting codes," *Inform. Control*, vol. 53, pp. 66-73, April 1982.

[27] J. H. Weber, C. de Vroedt, D. E. Boekee, "Bounds and constructions for codes correcting unidirectional errors," *IEEE Trans. Inform. Theory*, vol. 35, pp. 797-810, July 1989.

[28] F. J. H. Boinck, H. van Tilborg, "Constructions and bounds for systematic t-EC/AUED codes," *IEEE Trans. Inform. Theory*, vol. 36, pp. 1381-1390, Nov. 1990.

[29] P. Delsarte, P. Piret, "Bounds and constructions for binary asymmetric error correcting codes," *IEEE Trans. Inform. Theory*, vol. 27, pp. 125-128, Jan. 1981.

[30] M. Blaum, J. Bruck, "Unordered error-correcting codes and their applications," *Proceedings Intern. Sympos. Fault Tolerant Computing*, vol. 22, pp. 486-493, 1992.

[31] R. H. Deng, M. A. Hero, "DC-free coset codes," *IEEE Trans. Inform. Theory*, vol. IT-34, pp. 786-792, July 1988.

[32] Z. Zhang, X. -G. Xia, "LYM-type inequalities for tEC/AUED codes," *IEEE Trans. Inform. Theory*, vol. 39, pp. 232-238 , 1993.

[33] Z. Zhang, X. -G. Xia, "LYM inequalities for t-antichains," *Science in China (Series A)*, vol. 39, No. 10, pp. 1009-1024, October 1996.

[34] R. Ahlswede, Z. Zhang, "An identity in combinatorial extremal theory," *Advances in Mathematics*, vol. 80, pp. 137-151, 1990.