Coding Schemes for Non-Volatile Memories

Michal Horovitz
Coding Schemes for Non-Volatile Memories

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Michal Horovitz

Submitted to the Senate of
the Technion — Israel Institute of Technology
Tishrei 5778 Haifa September 2017
The research thesis was done under the supervision of Assistant Prof. Eitan Yaakobi and Prof. Tuvi Etzion in the Computer Science Department.

Acknowledgements

I would like to thank HaShem who sent me the right people, the ideas, the strength and the ability to write this thesis.

I am indebted to my advisors Prof. Eitan Yaakobi and Prof. Tuvi Etzion who guided the research and accompanied me closely with professionalism and humanity. Prof. Tuvi Etzion, with whom I started working towards the Master degree, and Prof. Eitan Yaakobi, who has been the principal advisor since the beginning of my doctoral studies.

Tuvi, you ignited with in me the desire to investigate. Thank you for teaching me what to ask and how, and for showing me the secrets of academic writing. Your door was always open for me. With unparalleled patience, you guided me thoroughly on how to do research from the beginning until the writing of the results. You gave me advice on every matter, with only my best interest at heart.

Eitan, you strengthened my desire to research. Thank you for guiding me on how to arrange the ideas and define a problem. Thank you for leading me to independence in research. You threw me into the water and trusted me. You taught me to prepare a lecture and a presentation. You were available to help me anytime, anywhere. You generated a research group and created a wonderful atmosphere.

Thank you both, Tuvi and Eitan, for your encouragement and leading me from the first step in the academic world to independence in research. Thank you for helping me receive scholarships and for caring about my success, even more than I cared for myself. Thank you for letting me learn from you in all areas, and I promise to make efforts to apply this knowledge.

Many thanks to the research group that is professional as well as fun. Thank you, the participants and organizers of the coding seminar, for a professional and supportive community.

I would like to thank all the wonderful friends I met in the faculty. You always showed interest, supported me when I was stuck and going through difficult moments, and shared my happiness with me in the most beautiful periods. You advised, directed, and painted my way in beautiful and pleasant colors.
In particular, I would like to thank the staff of the course "Automata and Formal Languages", which was my second home at the Technion, and headed by Prof. Shmuel Zaks, Prof. Michael Kaminski, Prof. Yuval Filmus and Dr. Mor Weiss. I am thankful to you for supporting and directing my personal development. You are a source of inspiration for me, of combining research and teaching. Thank you to all the staff, I have learned from each one of you so much regarding teaching and in general.

Thank you to the amazing and efficient secretarial staff, and to the great IT staff. You have always supported and taken an interest in my progress. You went far and beyond the requirements of your role.

Thank you to my family and friends who always supported and trusted me and who were genuinely interested and helped.

Special thanks to my dear parents, who pushed me to learn and study from a young age. Thank you for removing the obstacles from my path and for the countless hours and an enormous effort you invested for my studies and research. Thank you for always believing in me, and encouraging me to succeed. Thank you for your endless love and dedication. I cannot express in a few lines, and in even much more words, the gratitude I owe you.

Thank you to my sisters and brother, especially my sister Racheli, and my niece Efrat, for your great help with my children and for your support.

I am grateful to my special husband, who stood by me throughout the entire period. Thank you for your support, encouragement, interest in my research, and patience. You gave me the ability to work during the stressful periods, and the strength to reach this moment. You were my pillar all along. Thank you to the family of my husband for the support and help.

Thank you to my wonderful children. You gave me the strength and motivation. You taught me how to see things in proportion and many life skills. You made me quicker and more efficient, doing tasks in parallel. You taught me many things about myself that, without you, I would never have known.

This thesis is dedicated to all the dear people I care about. I would not have made it this far without you. Thank you very much!

The generous financial support of the Technion, the Gutwirth and Jacobs foundations and the Benin Scholarship Fund is gratefully acknowledged.
Publications List - Michal Horovitz

Journal Papers


Submitted Papers


Peer-Reviewed Conference Papers


Contents

Abstract 1

Abbreviations and Notations 3

A Note for the Reader 5

1 Introduction 9
   1.1 Write-Once Memory (WOM) Codes . . . . . . . . . . . . 11
   1.2 Rank Modulation for Flash Memories. . . . . . . . . . . . 13
      1.2.1 Snake-in-the-Box Codes . . . . . . . . . . . . . . . . 14
      1.2.2 Local Rank Modulation . . . . . . . . . . . . . . . . 15
   1.3 Sequence Reconstruction . . . . . . . . . . . . . . . . 15

2 Research Methods 17

I Published Journal Papers 19

3 Constructions of snake-in-the-box codes for rank modulation 21
   3.1 Introduction . . . . . . . . . . . . . . . . . . . . 22
   3.2 Preliminaries . . . . . . . . . . . . . . . . . . . 25
   3.3 Framework for Constructions of $K$-Snakes . . . . . . . . . 27
   3.4 A Recursive Construction . . . . . . . . . . . . . . . 33
   3.5 A Direct Construction based on Necklaces . . . . . . . . . 35
   3.6 Conclusions and Future Research . . . . . . . . . . . . 44

4 On the Capacity of Write-Once Memories 47
   4.1 Introduction . . . . . . . . . . . . . . . . . . . . 48
List of Figures

3.1 A \((5,57,K)\)-snake obtained by \(T_5\) .................. 32
3.2 The nearly spanning tree \(T_7\) constructed from \(T_5\) .......... 33
3.3 The graph \(\hat{G}_7\) and its component spanning tree \(\hat{T}_7\) ...... 42
3.4 The graph \(\hat{G}_9\) .................................................. 44

4.1 A comparison between \(C_{2,2}\) – the capacity regions for binary two-write WOM in models 1, 2, and 3, and \(\tilde{C}_2\)– an achievable region for binary two-write WOM in model 4. The points are rates of specific constructions of zero-error WOM codes in model 4 ............................................................. 56
4.2 A comparison between the capacity regions \(C_{3,2}\) (models 1 and 2 – the outer line) and \(\tilde{C}_{3,2}\) (model 3 – the inner line). 76
4.3 A comparison between, \(R_{q,t}^{(c)}\), the maximum sum-rates in models 1 and 2, and \(R_{q,t}^{(c),e}\), the maximum sum-rates in model 3, for \(q \in \{3,4\}\) ..................................................... 79
## List of Tables

4.1 Maximum Sum-Rates in Model 4 ........................................ 56
4.2 An $[n, 2; M_1, M_2]^{(i,2)}$ WOM Code Constructed by an $(n, M, \tau)_Z$ Asymmetric-Error-Correcting Code .......................... 61
4.3 A Comparison between Values of $R_{t}^{\text{sum}}$ and $R_{2,t}^{(i,2)}$ .......... 62
4.4 The Encoder’s Maps of Example 7 .................................... 63

5.1 The encoding key of the $(1, 3, n)$-LRM scheme. ............... 93
5.2 Succeeding states for complete states in the $(1, 3, n)$-LRM scheme. ................................................................. 98
5.3 The number of possible pairs $(g_{n-2}, g_{n-1})$ which can complete a given prefix of a codeword $(g_0, \ldots, g_{n-3})$ to a legal codeword, correspond to a complete state $P_{n-1}$, given $\pi_1$, in the $(1, 3, n)$-LRM scheme. ........................................ 100
5.4 The encoding key of the $(1, 3, n)$-LRM scheme .................. 114
5.5 Succeeding states in the $(1, 3, n)$-LRM scheme ............... 117
5.6 The number of base-words represented by the complete states in the $(1, 3, n)$-LRM scheme ........................................ 117

6.1 Examples for the solution to Problem 2 ......................... 163
Abstract

Flash memory is one of the most non-volatile memories. However, the asymmetry between memory cell programming and removing charge from cell motivates the design of rewriting schemes as Write-Once Memory (WOM) codes and Rank Modulation (RM).

WOM is a storage device consisting of $q$-ary cells that can only increase their value. A WOM code is a coding scheme which allows to write multiple times to the memory without decreasing the levels of the cells. In the conventional model, it is assumed that the encoder can read the memory before encoding, while the decoder knows only the last memory state. However, there are three more models which depend on whether the encoder and decoder are informed or uninformed with the previous state. These models were first introduced by Wolf et al. They extensively studied the capacity in the binary case of these four models. Yet, in the non-binary case only the conventional model, encoder informed and decoder uninformed, was studied by Fu and Vinck. We first provide constructions of WOM codes for the encoder uninformed models. Then, we study the capacity regions and maximum sum-rates of non-binary WOM for all models.

Jiang et al. suggested the RM scheme to cope with the overshooting and the charge leakage problems. Snake-in-the-box code is a Gray code which can detect a single error. Two consecutive codewords in RM Gray code are obtained by using the minimal cost operation "push-to-the-top". We consider the Kendall’s $\tau$-metric due to its relevance for common errors. We proposed two constructions for snakes. The recursive construction for snake of length $M_{2n+1}$ of permutations from $S_{2n+1}$ attains $\lim_{n \to \infty} \frac{M_{2n+1}}{|S_{2n+1}|} \approx 0.4338$, improving on the previous ratio of $\frac{1}{\sqrt{\pi n}}$ given by Yehezkeally and Schwartz who also proved an upperbound of 0.5 for this ratio. The second is a direct construction for snakes of asymptotically optimal length $\frac{(2n+1)!}{2} - 2n + 1$. 

1
The proof of this construction was completed later by Zhang and Ge.

The local rank modulation (LRM) scheme was suggested to overcome the drawbacks of RM. In the \((s, t, n)\)-LRM scheme, \(0 < s \leq t \leq n\) with \(s\) divides \(n\), the \(n\) cells are locally viewed cyclically through a sliding window of size \(t\) which requires less comparisons and less distinct values. The gap between two windows is \(s\). We consider the encoding, decoding, and asymptotic enumeration of the \((1, t, n)\)-LRM scheme.

Lastly, motivated by DNA storage, we consider the sequence reconstruction problem, which was first studied by Levenshtein. We assume that the number of errors in different channels can vary, and we define two main problems. First, we study the minimum number of channels required for exact reconstruction for a given code. Then, we find the minimum distance of the code that guarantees exact reconstruction where the number of channels is given. In both problems, we define three models depend on whether the decoder knows the number of errors in each channel, the distribution for the number of errors, or only the average number of errors.
Abbreviations and Notations

\( S_n \) — The set of permutations over \( n \).

- \( \text{RM} \) — Rank modulation
- \( \text{LRM} \) — Local rank modulation
- \( \text{RMGC} \) — Rank modulation Gray code
- \( \text{WOM} \) — Write-once memory
- \( \text{EI/EU}, \text{DI/DU} \) — Encoder informed/uninformed, decoder informed/uninformed

The \( (s,t,n) \)-LRM scheme

- \( n \) cells,
- \( t \) is the size of the sliding window, and \( s \) is gap between two windows, where \( s \) divides \( n \)

WOM model 1,2,3,4

\([n,t; M_1, \ldots , M_{t/q}]^P_e\) WOM code

- \( M_i \) is the number of messages in the \( i \)-th write,
- \( q \) is the number of charge levels,
- \( p_e \) the decoding error probabilities, and \( k \) denotes the WOM model

- \( \mathcal{C}^{(k)}_{q,t} \), \( \mathcal{R}^{(k)}_{q,t} \) — The capacity region for \( q \)-ary \( t \)-write WOM in model \( k \), in the zero-error, the \( \epsilon \) error cases.

- \( \mathcal{R}^{(k)}_{q,t} \) — The maximum sum-rate for \( q \)-ary \( t \)-write WOM in model \( k \), in the zero-error, the \( \epsilon \) error cases.
A Note to the Reader

This dissertation is based on the following publications.

**Journal Papers**

[J1] M. Horovitz and E. Yaakobi,  
“On the Capacity of Write-Once Memories”,  

[J2] M. Horovitz and T. Etzion,  
“Constructions of Snake-in-the-Box Codes for Rank Modulation”,  

**Submitted Papers**

[J3] M. Horovitz and E. Yaakobi,  
“Reconstruction of Sequences over Non-Identical Channels”,  

[J4] M. Horovitz and T. Etzion,  
“Local Rank Modulation for Flash Memories”,  
submitted to *IEEE Transactions on Information Theory*, June 2017

**Peer-Reviewed Conference Papers**

[C1] M. Horovitz and E. Yaakobi,  
“Reconstruction of Sequences over Non-Identical Channels”,  
All these research works were done in collaboration with my supervisors, Prof. Eitan Yaakobi and Prof. Tuvi Etzion. They introduced me with the field, referred me to the relevant literature, and helped with asking the right questions. They suggested methods for solving the problems, and of course accompanied the research along all the way, including the writing of the above papers.

The above list consists of published journal papers, unpublished journal papers, and published conference papers. Paper [J1], which has already been published, is a long version of the two less detailed conference versions [C3] and [C2]. Thus, these two conferences papers are not included in this dissertation.

I have two papers, which were recently submitted to the IEEE transactions on information theory journal. Yet, to the date of writing this dissertation, the review reports of these papers have not been received yet. The journal paper [J4] is a long version of [C4] which was published in IEEE Information Theory Workshop (ITW) 2014, and paper [J3] generalizes the conference paper [C1] which was published in IEEE International Symposium on Information Theory (ISIT) 2017.

This dissertation contains two main parts. The first part contains the published journal papers [J2] and [J1], one chapter for each paper. The second part is dedicated to two conference papers with their unpublished journal version. In this part, each conference paper is followed by its submitted unpublished long version, where each topic is presented in a separate
chapter. On one hand, the conference versions are rather succinct and incomplete, while the journal versions are complete and detailed. But, on the other hand, the conference versions have already been reviewed and published, while the journal version have only been submitted recently. Thus, I provide the two versions. Note, that as a result, some parts of this research appear twice, however for readability I left those partial repetitions. This part contain the published conference papers [C4] and [C1] with their respective submitted journal versions [J4] and [J3].

Following the instructions of the Technion’s Graduate School, the structural of this dissertation is as follows. The first chapter, Introduction, presents a comprehensive and up-to-date overview of all areas of research. Afterwards, the Research Methods chapter describes the techniques developed and used during the work. Then, we present the conference and journal papers organized in the two parts as described above, where each paper contains its own list of references. The last chapter, Discussion, includes a brief summary of the results while taking into account the coherence and integration of the entire work. The separate bibliography which is given in p. 204 lists the references which are given in the Introduction, the Research methods, and the Discussion chapters.
Chapter 1

Introduction

One of the most useful applications of coding theory is the channel coding, which is known as error-correction coding. In this setup, the coding is used for communication as follows. The sender encodes the message data with redundancy using an error-correcting code. The encoded data is sent over the noisy channel, and the receiver gets the noisy encoded data. The redundancy allows the receiver to detect or to correct a limited number of errors.

A coding scheme for memories applies the same ideas. The memory which stores the data might be damaged during its lifetime, and therefore it behaves like a noisy channel. The data is encoded with redundancy, and the encoded data is stored in the memory. The reader reads the memory, and can decode the data by the redundancy. The main goal is to maintain the memory reliable, while decreasing the redundancy as much as possible.

The channel coding methods for communication have been studied extensively. However, the conventional techniques may no longer be appropriate in the context of new storage devices, since the types of the errors occurred are essentially different. For example, many traditional methods are designed to overcome symmetric errors, like the Hamming metric, yet modern memories, like flash memories, exhibit a very high levels of asymmetry. Using the symmetric error correcting codes in such applications is suboptimal and wasteful. The modern techniques for storing data which are essential to cope with the huge growing of the data, require the design of new coding schemes that are appropriate especially for these new memories.

The research of coding for memories involves both information theory
and coding theory. While information theory studies lower and upper bounds on the data size which can be stored reliably in the memory, coding theory is concerned with finding explicit methods, called codes, for efficiently store the data, when some of the goals are decreasing redundancy, increasing reliability, and improving the complexity of the encoding-decoding algorithms.

This work studies coding schemes for non-volatile memories, while focusing on two types of memories, flash memory and DNA storage. Most of the results in this dissertation concern with flash memory which is the main storage medium studied in this work. The exceptional part is Chapter 6 which studies the sequence reconstruction problem that is motivated by DNA storage.

Flash memory is the most popular non-volatile memory due to its reliability, high storage density, and relative low cost. This storage medium consists of memory elements, called cells, which are electrically programmable and erasable. However, the most conspicuous property of flash memory is its asymmetry between cell programming and cell erasing. While adding charge to a single cell is a fast and simple operation, removing charge is very difficult. In fact, most flash technologies do not allow erasing a single cell, but only a block of cells. The size of a block depends on the specific flash technology and can be as large as $10^5$ and as small as few hundreds. This constraint does not affect only the speed and performance, but also the lifetime, which is a function of the number of block erasures. Thus, flash memories motivate the study of several coding schemes which help in mitigating the constraints. All these methods are targeted to reduce the number of block erasures as much as possible. In this thesis, we refer to two of the known schemes, namely, write-once memory (WOM) and rank modulation (RM).

The new results regarding WOM are presented in Chapter 4 which consists of our published journal paper [20] based on the papers [17, 18]. This is a joint work with Prof. Yaakobi.

Chapters 3 and 5 refer to the rank modulation scheme. These chapters contain the following papers [13, 14, 16], which are a joint work with Prof. Etzion.

Finally, Chapter 6, studies the sequence reconstruction problem, which is connected to DNA storage. This is also a joint work with Prof. Yaakobi.
1.1 Write-Once Memory (WOM) Codes

A write-once memory (WOM) is a storage medium with memory elements, called cells. The cells can be either binary or can take on more values. The main property which distinguishes a WOM from other memories is that values written to the storage device are non-decreasing over a fixed number of writes.

A WOM code, first introduced in 1982 by Rivest and Shamir [33], is a coding scheme for storing messages in $n$ cells in such a way that each cell can change its value only from a lower level to a higher one. These codes make it possible to record data more than once in a write-once storage medium. The bit value '0' denotes the erased state of a cell. The goal is to maximize the number of bits which are possible to be written to the memory in $t$ writes, while guaranteeing that each cell can only increase its level.

Recently, there has been a renewed interest in these codes due to their relevance for the ubiquitous flash memories. In the conventional modulation scheme used in flash-memory cells, the charge levels of each cell is quantized to one of $q$ levels, resulting in a single demodulated symbol from an alphabet of size $q$. Since in flash memory it is only possible to increase the levels of the cells (unless it is erased), it can be modeled as a write-once memory, and WOM codes can be used to prolong its lifetime.

WOM codes have been extensively studied in the literature, especially for the binary case, see example [3, 22, 23, 27, 31, 34, 40]. Non-binary WOM codes were also studied in several other works, e.g. [11, 27, 42].

A similar phenomenon might appear in flash memories. Here, the cells are programmed by electrically charging them with electrons in order to represent multiple levels. If charge is trapped in a cell, then its level can only be increased, or it may happen that due to defects, the cell can only represent some lower levels stuck-at model, the cells are mopacan also be used for the cells become stuck-.

The rate of the $i$-th write is the ratio of the number of written bits to the total number of cells, and the sum-rate of the WOM code is the sum of all individual rates on all writes. The capacity region of a $t$-write WOM is the set of all achievable rate tuples. The main problems when studying WOM are finding the capacity regions, as well as finding efficient codes which achieve maximum sum-rate.
We also consider two types of WOM codes, the \( \epsilon \)-error and the zero-error cases. Informally, these families of codes refer to the probability of failure in decoding. In the zero-error case, each encoded message should be decoded correctly, and in the \( \epsilon \)-error case, the probability of failure in decoding tends to 0, as \( n \), the number of cells, increases.

Almost all existing research on WOM codes considers the model in which the encoder can read the current state of the cells before programming, while the decoder has access only to the last state of the cells after programming but not before that. However, there are three other models regarding the inputs of the encoder and decoder. These models were proposed first by Wolf et al. \cite{39}. The input of the encoding function either include the memory state or not, and it is denoted by EI (encoder informed) and EU (encoder uninformed), respectively. The input of the decoding function may include the memory state before the last encoding or not, and, as for the encoder, it will be denoted by DI (decoder informed) and DU (decoder uninformed), respectively. If the encoder reads the memory before programming the new data, then the memory state is part of the encoder’s input. Thus we have four models, denoted as follows, model 1 – EI:DI, model 2 – EI:DU, model 3 – EU:DI, and model 4 – EU:DU. For formal definition of the models for both the \( \epsilon \)-error and the zero-error cases, see Definition 7 in Chapter 4.

The conventional model, the EI:DU, was mostly studied in the literature due to its practical uses. In this model the encoder reads the memory before programming new data, and the decoder does not have any additional input, except for the last memory state. Additionally, models 3 and 4 can be used for the construction of RIO codes, which are designed for fast programming and reading in flash memories \cite{35, 41}.

We study all these four models. First, we present constructions for WOM codes in models 3 and 4, the EU models, especially for the binary case. We propose several simple constructions, and we also connect these models to two known channels; the binary erasure channel and the Z channel which are used for constructing binary WOM codes in models 3 and 4, respectively.

In addition, we study the capacity regions and the maximum sum-rates in all these models. From the information-theoretic point of view, model 1 is the easiest one, while the most difficult one is model 4. The binary case was rigorously studied by Wolf et al. \cite{39}. Regarding the non-binary case, the capacity region in model 2, and the maximum sum-rate for models 1
and 2 were already solved in [9, 10, 13] only for the zero-error case.

We extend the study of the capacity region and the maximum sum-rate in the non-binary case for all four models. For example, we prove that the capacity region, as well as the maximum sum-rate for the EI models in the $\epsilon$-error and the zero-error cases are all the same. For model 3 the main result is that the capacity region in the $\epsilon$-error non-binary case is, unlike the binary case, a proper subset of the capacity region in models 1 and 2, and the maximum sum-rate is lower. Model 4 is partially solved for the binary case in [39], and we have some results regarding the non-binary case. However, this model is still almost entirely unresolved.

Lastly, we mention that two-write WOM codes are closely connected to codes for stuck-at cells [12, 26, 28, 32]. In the binary case, the model of stuck-at cells assumes that a cell can get stuck at one of its two states, zero or one. For the non-binary case, assume that a cell can have one of the levels $0, 1, \ldots, q$. Then, it is said that a cell is partially stuck-at value $s$, where $0 < s \leq q - 1$, if the cell can store only values which are no less than $s$; see [36] and references therein. In the WOM model, and especially in flash memory, the cells cannot decrease their values. Thus, the programmed cells can be considered as partially stuck-at [36]. In the common stuck-at cells setup, the encoder knows the positions and the values of the stuck-at cells, but the decoder does not know this information. A code for binary stuck-at cells can be used in order to construct a binary two-write WOM code for the EI:DU model by treating the programmed cells on the first write of the WOM code as stuck-at cells. Similarly, codes for partially stuck-at cells can be used for constructing non-binary two-write WOM codes. More details about this strong connection are described in [36].

1.2 Rank Modulation for Flash Memories

Flash memory technology does not support charge removal from individual cells. For this reason, charge is injected into the cell over several iterations. Such programming is slow and can cause errors since cells may be injected with extra unwanted charge. Thus, a main challenge in flash memories is to exactly program each cell to its designated level. Furthermore, flash memories suffer from the cell leakage problem, by which a charge may leak from the cells and thus cause reading errors. In order to overcome these
difficulties, the novel framework of rank modulation was introduced in [24].

Under this setup, the information is represented by permutations which are derived by the relative charge levels of the cells, rather than by their absolute levels. This allows for more efficient programming of cells, and coding by the ranking of the cells’ levels is more robust to charge leakage than coding by their actual values. In this model codes are subsets of $S_n$, the set of all permutations on $n$ elements, and the codewords are permutations of $S_n$, where each permutation corresponds to a ranking of the $n$ cells’ levels. The permutation is defined by the order of the charge levels, from the highest one to the lowest one. The only allowed operation is the basic minimal cost push-to-the-top operation, which means raising the charge level of one of the cells to be the highest level.

1.2.1 Snake-in-the-Box Codes

In [24], Gray codes were presented in order to exploit the full representational power of rank modulation and data rewriting schemes. The usage of Gray codes for rank modulation was also discussed in [6, 7, 46]. The goal is, as in the conventional technique, to maximize the number of times data can be written between two erase operations. A Gray code for rank modulation scheme is a sequence of permutations from $S_n$, such that each permutation in the sequence is a result of a push-to-the-top operation of the previous permutation. A snake-in-the-box code is a Gray code which is capable to detect a single error. Snake-in-the-box codes for rank modulation were presented first in [46]. The distance measure for a snake-in-the-box code is defined on permutations. There are some interesting metrics, and we focus on the Kendall’s $\tau$-metric [25]. The Kendall’s $\tau$-distance between two permutations is the minimum number of adjacent transpositions required to obtain one permutation from the second, where an adjacent transposition is an exchange of two distinct adjacent elements. The motivation for the Kendall’s $\tau$-metric is that a small charge constrained error is translated into a small distance in the metric.

One of the most important problems in the research on snake-in-the-box codes is to construct codes of size as large as possible. We improved the results in a recent paper by Yehezkeally and Schwartz [46] by constructing two snakes for odd $n$. The first construction is proved completely in our
paper. However the proof for second construction, called the direct con-
struction, is completed later by Zhang and Ge [47], who also proposed a
slightly improved construction using similar ideas. The direct construction
is asymptotically optimal. Our results regarding snake-in-the-box codes are
presented in [13], Chapter 3. The case of even $n$ was discussed later by
Wang and Fu [37], and then by Zhang and Ge [48].

1.2.2 Local Rank Modulation

The local rank modulation ($LRM$) scheme was defined recently [6, 7, 38] for
representing information in flash memories in order to overcome drawbacks
of rank modulation. In this scheme, the $n$ cells are locally viewed through a
sliding window, resulting in a sequence of small permutations which requires
less comparisons and less distinct values. To get the simplest hardware
implementation, in [6] a specific LRM-scheme was discussed in which the
size of the sliding window is 2. In [14, 16], Chapter 5, we present a new
family of local rank modulation schemes. We refer to the encoding, decoding
and enumeration in each proposed scheme.

1.3 Sequence Reconstruction

Recently, the research community, as well as the industry community, are
interested in DNA storage. Due to its very high density, it might be one of
the main storage technologies in the near future. In [19, 21], see Chapter 6,
we study the sequence reconstruction problem. This problem is suggested
and studied first by Levenshtein [29, 30]. In this model, a codeword is
transmitted over multiple channels and the receiver decodes the transmitted
word by the outputs of all the channels. This model is relevant to storage
systems where there exist multiple copies of the information or where a
single copy is read by several different read heads. Specifically, this model is
relevant to DNA storage [1, 2, 43, 44, 45]. In this setup, the information has a
large number of copies stored in DNA strands and the goal is to reconstruct
the data by reading these strands, where every estimation of the data is
erroneous. We generalize the sequence reconstruction problem proposed by
Levenshtein [29] to the case where not all the channels are identical. We
study two main problems. In the first one, the code is given, that is, the
set of all possible inputs to the channels, and the problem is to find the minimum number of channels which is required to enable the decoder to reconstruct the data. For the second problem, the number of channels is given, and we seek to find the minimum distance of the code which is required for reconstruction the data.
Chapter 2

Research Methods

This chapter provides an overview of several techniques which were used in the research described in this dissertation. It includes many mathematical methods which are derived from combinatorics, discrete mathematics, and information theory. We also used computer search algorithms for several purposes.

In the papers [18, 20] we prove the capacity region for some WOM models. Each such a proof includes two parts, which both of them contain well-known tools from the information theory study. The direct part relies on the random channel-coding theorem [4, p. 200], and the converse part is based on Fano’s inequality [4, p. 38]. In these proofs we apply some ideas from [9, 39].

The constructions proposed in [13] involve several techniques from discrete mathematics. The main tool which is used many times is recursion. It is applied in the first construction which is called the recursive construction, and it is used also for constructing the nearly spanning tree in Theorem 1. These constructions apply a similar technique to the method of joining cycles from the pure cycling register of order $2n - 1$, PCR$_{2n-1}$, into one cycle, which is known as a de Bruijn sequence [5, 8]. This join is the main idea which our constructions rely on.

We use additional basic tools from combinatorics and discrete mathematics. The induction technique is used in many other proofs for example, the proofs of Lemmas 13 and 25,27,28,29 from [20] and [21], respectively. The combinatorics for enumeration was used in [16] for finding out the sizes of the codeword in the proposed LRM schemes in Theorem 15, and also in
Lemma 19 to characterize the complete states. Additionally, in [21] it is applied for counting the number of words in the intersection of two balls centered at two different points of distance $d$ under the Hamming or the Johnson distance function; see Lemmas 31 and 35.

Lastly, we note that many results which are presented in this dissertation were supported by computer algorithms we composed and applied for our specific purposes. We used this tool first to find the idea, then to prove it or to provide examples. For example, in [13] we verified by computer search the correctness of the direct construction for small parameters, see Conjecture 2. Furthermore, all the examples in this paper were achieved by effective computer searches.
Part I

Published Journal Papers
Chapter 3

Constructions of snake-in-the-box codes for rank modulation

Michal Horovitz and Tuvi Etzion

Abstract

Snake-in-the-box code is a Gray code which is capable of detecting a single error. Gray codes are important in the context of the rank modulation scheme which was suggested recently for representing information in flash memories. For a Gray code in this scheme the codewords are permutations, two consecutive codewords are obtained by using the "push-to-the-top" operation, and the distance measure is defined on permutations. In this paper the Kendall’s $\tau$-metric is used as the distance measure. We present a general method for constructing such Gray codes. We apply the method recursively to obtain a snake of length $M_{2n+1} = ((2n + 1)(2n) - 1)M_{2n-1}$ for permutations of $S_{2n+1}$, from a snake of length $M_{2n-1}$ for permutations of $S_{2n-1}$. Thus, we have $\lim_{n \to \infty} \frac{M_{2n+1}}{|S_{2n+1}|} \approx 0.4338$, improving on the previous known ratio of $\frac{1}{\sqrt{\pi n}}$. By using the general method we also present a direct construction. This direct construction is based on necklaces and it might yield snakes of length $\frac{(2n+1)!}{2} - 2n + 1$ for permutations of $S_{2n+1}$. The direct construction was applied successfully for $S_7$ and $S_9$, and hence $\lim_{n \to \infty} \frac{M_{2n+1}}{|S_{2n+1}|} \approx 0.4743$. 

21
3.1 Introduction

Flash memory is a non-volatile technology that is both electrically programmable and electrically erasable. It incorporates a set of cells maintained at a set of levels of charge to encode information. While raising the charge level of a cell is an easy operation, reducing the charge level requires the erasure of the whole block to which the cell belongs. For this reason charge is injected into the cell over several iterations. Such programming is slow and can cause errors since cells may be injected with extra unwanted charge. Other common errors in flash memory cells are due to charge leakage and reading disturbance that may cause charge to move from one cell to its adjacent cells. In order to overcome these problems, the novel framework of rank modulation was introduced in [8]. In this setup the information is carried by the relative ranking of the cells’ charge levels and not by the absolute values of the charge levels. This allows for more efficient programming of cells, and coding by the ranking of the cells’ levels is more robust to charge leakage than coding by their actual values. In this model codes are subsets of \( S_n \), the set of all permutations on \( n \) elements, and the codewords are members of \( S_n \), where each permutation corresponds to a ranking of \( n \) cells’ levels from the highest one to the lowest. For example, the charge levels \((c_1, c_2, c_3, c_4) = (5, 1, 3, 4)\) are represented by the codeword \([1, 4, 3, 2]\) since the first cell has the highest level, the forth cell has the next highest level and so on.

To detect and/or correct errors caused by injection of extra charge or due to charge leakage we will use an appropriate distance measure. Several metrics on permutations are used for this purpose. In this paper we will consider only the Kendall’s \( \tau \)-metric [9, 10]. The Kendall’s \( \tau \)-distance between two permutation \( \pi_1 \) and \( \pi_2 \) in \( S_n \) is the minimum adjacent transpositions required to obtained \( \pi_2 \) from \( \pi_1 \), where adjacent transposition is an exchange of two distinct adjacent elements. For example, the Kendall’s \( \tau \)-distance between \( \pi_1 = [2, 1, 4, 3] \) and \( \pi_2 = [2, 4, 3, 1] \) is 2 as \([2, 1, 4, 3] \rightarrow [2, 4, 1, 3] \rightarrow [2, 4, 3, 1] \). Two permutations in this metric are at distance one if they differ in exactly one pair of adjacent elements. Distance one between these two permutations represent an exchange of two cells, which are adjacent in the permutation, due to a small change in their charge level which change their order.
Gray codes are very important in the context of rank modulation as was explained in [8]. They are used in many other applications, e.g. [3, 12]. An excellent survey on Gray codes is given in [11]. The usage of Gray codes for rank modulation was also discussed in [2, 6, 8, 13]. The permutations of $S_n$ in the rank modulation scheme represent "new" logical levels of the flash memory. The codewords in the Gray code provide the order of these levels which should be implemented in various algorithms with the rank modulation scheme. Usually, a Gray code is just a simple cycle in a graph, in which the edges are defined between vertices with distance one in a given metric. Two adjacent vertices in the graph represent on one hand two elements whose distance is one by the given metric; and on the other hand a move from a vertex to a vertex implied by an operation defined by the metric. A snake-in-the-box code is a Gray code in which two elements in the code are not adjacent in the graph, unless they are consecutive in the code. Such a Gray code can detect a single error in a codeword. Snake-in-the-box codes were mainly discussed in the context of the Hamming scheme, e.g. [1].

In the rank modulation scheme the Gray code is defined slightly different since the operation is not defined by a metric. The permutation is defined by the order of the charge levels, from the highest one to the lowest one. From a given ranking of the charge levels, which defines a permutation, the next ranking is obtained by raising the charge level of one of the cells to be the highest level. This operation, called "push-to-the-top", is used in the rank modulation scheme. For example, the charge levels $(c_1, c_2, c_3, c_4) = (5, 1, 3, 4)$ are represented by the codeword $[1, 4, 3, 2]$, and by applying push-to-the-top operation on the second cell which has the lowest charge level, we have, for example, the charge levels $(c_1, c_2, c_3, c_4) = (5, 6, 3, 4)$ which are represented by the codeword $[2, 1, 4, 3]$. Hence, the permutation $\pi_2$ can follow the permutation $\pi_1$ if $\pi_2$ is obtained from $\pi_1$ by applying a push-to-the-top operation on $\pi_1$. Therefore, the related graph is directed with an outgoing edge from the vertex which represents $\pi_1$ into the vertex which represents $\pi_2$. On the other hand, one possible metric for the scheme is the Kendall’s $\tau$-metric. A Gray code (and a snake-in-the-box code as a special case) related to the rank modulation scheme is a directed simple cycle in the graph. In a snake-in-the-box code, related to this scheme, there is another requirement that the Kendall’s $\tau$-distance between any two codewords is at least two, including consecutive codewords. For example,
is a snake-in-the-box code in $S_4$ obtained by applying a push-to-the-top operation on the lowest cell at each time. The Kendall’s $\tau$-distance between any two permutations in $C$ is at least 2.

One of the most important problems in the research on snake-in-the-box codes is to construct the largest possible code for the given graph. In a snake-in-the-box code for the rank modulation scheme we would like to find such a code with the largest number of permutations. In a recent paper by Yehezkeally and Schwartz [13], the authors constructed a snake-in-the-box code of length $M_{2n+1} = (2n + 1)(2n - 1)M_{2n-1}$ for permutations of $S_{2n+1}$, from a snake of length $M_{2n-1}$ for permutations of $S_{2n-1}$. We will improve on this result by constructing a snake of length $M_{2n+1} = ((2n + 1)2n - 1)M_{2n-1}$ for permutations of $S_{2n+1}$, from a snake of length $M_{2n-1}$ for permutations of $S_{2n-1}$. Thus, we have $\lim_{n \to \infty} \frac{M_{2n+1}}{|S_{2n+1}|} \approx 0.4338$, improving on the previous known ratio of $\frac{1}{\sqrt{2\pi n}}$ [13]. For these constructions of snake-in-the-box codes we need an initial snake-in-the-box code and the largest one known to start both constructions is a snake of length 57 for permutations of $S_5$. We also propose a direct construction to form a snake of length $\frac{(2n+1)!}{2} - 2n + 1$ for permutations of $S_{2n+1}$. The direct construction was applied successfully for $S_7$ and $S_9$. This implies better initial condition for the recursive constructions, and the ratio $\lim_{n \to \infty} \frac{M_{2n+1}}{|S_{2n+1}|} \approx 0.4743$.

The rest of this paper is organized as follows. In Section 3.2 we will define the basic concepts of Gray codes in the rank modulation scheme, the push-to-the-top operation, and the Kendall’s $\tau$-metric required in this paper. In Section 3.3 we present the main ideas and a framework for constructions of snake-in-the-box codes. In Section 3.4 we present a recursive construction based on the given framework. This construction is used to obtain snake-in-the-box codes longer than the ones known before. In Section 3.5, based on the framework, we present an idea for a direct construction based on necklaces. The construction is used to obtain snake-in-the-box codes of length $\frac{(2n+1)!}{2} - 2n + 1$ in $S_{2n+1}$, which we believe are optimal. The construction was applied successfully on $S_7$ and on $S_9$, and we conjecture that it can be applied on $S_n$ for any odd $n > 6$. Conclusions and problems for future research are presented in Section 3.6.
3.2 Preliminaries

In this section we will repeat some notations defined and mentioned in [13], and we also present some other definitions.

Let \([n] \triangleq \{1, 2, \ldots, n\}\) and let \(\pi = [a_1, a_2, \ldots, a_n]\) be a permutation over \([n]\), i.e., a permutation in \(S_n\), such that for each \(i \in [n]\) we have that \(\pi(i) = a_i\).

Given a set \(S\) and a subset of transformations \(T \subseteq \{f | f : S \rightarrow S\}\), a Gray code over \(S\) of size \(M\), using transitions from \(T\), is a sequence \(C = (c_0, c_1, \ldots, c_{M-1})\) of \(M\) distinct elements from \(S\), called codewords, such that for each \(j \in [M-1]\) there exists a \(t \in T\) for which \(c_j = t(c_{j-1})\). The Gray code is called complete if \(M = |S|\), and cyclic if there exists \(t \in T\) such that \(c_0 = t(c_{M-1})\). Throughout this paper we will consider only cyclic Gray codes.

In the context of rank modulation for flash memories, \(S = S_n\) and the set of transformations \(T\) comprises of push-to-the-top operations. We denote by \(t_i\) the push-to-the-top operation on index \(i\), \(2 \leq i \leq n\), defined by

\[
t_i([a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n]) = [a_i, a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n].
\]

and a \(p\)-transition will be an abbreviated notation for a push-to-the-top operation.

A sequence of \(p\)-transitions will be called a transitions sequence. A permutation \(\pi_0\) and a transitions sequence \(t_1, t_2, \ldots, t_\ell\) define a sequence of permutations \(\pi_0, \pi_1, \pi_2, \ldots, \pi_{\ell-1}, \pi_\ell\), where \(\pi_i = t_i(\pi_{i-1})\), for each \(i, 1 \leq i \leq \ell\). This sequence is a cyclic Gray code, if \(\pi_\ell = \pi_0\) and for each \(0 \leq i < j < \ell\), \(\pi_i \neq \pi_j\). In the sequel the word cyclic will be omitted.

Given a permutation \(\pi = [a_1, a_2, \ldots, a_n] \in S_n\), an adjacent transposition is an exchange of two distinct adjacent elements \(a_i, a_{i+1}\), in \(\pi\), for some \(1 \leq i \leq n-1\). The result of such an adjacent transposition is the permutation \([a_1, \ldots, a_{i-1}, a_{i+1}, a_i, a_{i+2}, \ldots, a_n]\). The Kendall’s \(\tau\)-distance [10] between two permutations \(\pi_1, \pi_2 \in S_n\) denoted by \(d_K(\pi_1, \pi_2)\) is the minimum number of adjacent transpositions required to obtain the permutation \(\pi_2\) from the permutation \(\pi_1\). A snake-in-the-box code is a Gray code in which for each two permutations \(\pi_1, \pi_2 \in S_n\) we have \(d_K(\pi_1, \pi_2) \geq 2\). Hence, a snake-in-the-box code is a Gray code capable of detecting one Kendall’s \(\tau\)-error. We will call such a snake-in-the-box code a \(K\)-snake. We further denote by \((n, M, K)\)-snake a \(K\)-snake of size \(M\) with permutations from \(S_n\).
A $\mathcal{K}$-snake can be represented in two different equivalent ways:

- the sequence of codewords (permutations),
- the transitions sequence along with the first permutation.

Let $T$ be a transitions sequence and let $\pi$ be a permutation in $S_n$. If a $\mathcal{K}$-snake is obtained by applying $T$ on $\pi$ then a $\mathcal{K}$-snake will be obtained by using any other permutation from $S_n$ instead of $\pi$. This is a simple observation from the fact that $t(\pi_2(\pi_1)) = \pi_2(t(\pi_1))$, where $t$ is a $p$-transition and $\pi_2(\pi_1)$ refers to applying the permutation $\pi_2 \in S_n$ on the permutation $\pi_1 \in S_n$. In other words applying $T$ on a different permutation just permute the symbols, by a fixed given permutation, in all the resulting permutations when $T$ is applied on $\pi$. Therefore, such a transitions sequence $T$ will be called an $S$-skeleton.

For a transitions sequence $\sigma = t_{k_1}, t_{k_2}, \ldots, t_{k_\ell}$ and a permutation $\pi \in S_n$, we denote by $\sigma(\pi)$, the permutation obtained by applying the sequence of $p$-transitions in $\sigma$ on $\pi$, i.e., $t_{k_1}$ is applied on $\pi$, $t_{k_2}$ is applied on $t_{k_1}(\pi)$, and so on. In other words, $\sigma(\pi) = (t_{k_1} \circ t_{k_2} \circ \ldots \circ t_{k_\ell})(\pi) = t_{k_\ell}(t_{k_{\ell-1}}(\ldots t_{k_2}(t_{k_1}(\pi))))$. Let $\sigma_1, \sigma_2$ be two transitions sequences. We say that $\sigma_1$ and $\sigma_2$ are matching sequences, and denote it by $\sigma_1 \leftrightarrow \sigma_2$, if for each $\pi \in S_n$ we have $\sigma_1(\pi) = \sigma_2(\pi)$.

In [13] it was proved that a Gray code with permutations from $S_n$ using only $p$-transitions on odd indices is a $\mathcal{K}$-snake. By starting with an even permutation and using only $p$-transitions on odd indices we get a sequence of even permutations, i.e., a subset of the $A_n$, the alternating group of order $n$. This observation saves us the need to check whether a Gray code is in fact a $\mathcal{K}$-snake, at the cost of restricting the permutations in the $\mathcal{K}$-snake to the set of even permutations. However, the following assertions were also proved in [13]:

- If $C$ is an $(n, M, \mathcal{K})$-snake then $M \leq \frac{|S_n|}{2}$.
- If $C$ is an $(n, M, \mathcal{K})$-snake which contains a $p$-transition on an even index then $M \leq \frac{|S_n|}{2} - \frac{1}{n-1}(\lfloor n/2 \rfloor - 1)$.

This motivates not to use $p$-transitions on even indices. Since we will use only $p$-transitions on odd indices, we will describe our constructions only for even permutations with odd length.
3.3 Framework for Constructions of $\mathcal{K}$-Snakes

In this section we present a framework for constructing $\mathcal{K}$-snakes in $S_{2n+1}$. Our snakes will contain only even permutations. We start by partitioning the set of even permutations of $S_{2n+1}$ into classes. Next, we describe how to merge $\mathcal{K}$-snakes of different classes into one $\mathcal{K}$-snake. We conclude this section by describing how to combine most of these classes by using a hypergraph whose vertices represent the classes and whose edges represent the classes that can be merge together in one step.

We present two constructions for a $(2n+1, M_{2n+1}, \mathcal{K})$-snake, $C_{2n+1}$, one recursive and one direct. In this section we present the framework for these constructions. First, the permutations of $A_{2n+1}$, the set of even permutations from $S_{2n+1}$, are partitioned into classes, where each class induces one $\mathcal{K}$-snake which contains permutations only from the class. All these snakes have the same S-skeleton. Let $L_{2n+1}$ be the set of all the classes.

The construction of $C_{2n+1}$ from the $\mathcal{K}$-snakes of $L_{2n+1}$ proceeds by a sequence of joins, where at each step we have a main $\mathcal{K}$-snake, and two $\mathcal{K}$-snakes from the remaining $\mathcal{K}$-snakes of $L_{2n+1}$ are joined to the current main $\mathcal{K}$-snake. A join is performed by replacing one transition in the main $\mathcal{K}$-snake with a matching sequence.

In order to join the $\mathcal{K}$-snakes we need the following lemmas, for which the first can be easily verified. In the sequel, let $\sigma^k \triangleq \underbrace{\sigma \circ \sigma \circ \ldots \circ \sigma}_{k \text{ times}}$, i.e., performing the transitions sequence $\sigma$, $k$ times.

**Lemma 1.** If $\alpha, \beta \in S_n$ then $\beta = t_i(\alpha)$ if and only if $\alpha = t_i^{-1}(\beta)$.

**Lemma 2.** If $i \in [n-2]$ then $t_i \leftrightarrow t_{i+2} \circ (t_i^{-1} \circ t_{i+2})^2$.

**Proof.** Let $\alpha = [a_1, a_2, \ldots, a_i, a_{i+1}, a_{i+2}, \ldots, a_n]$ be a permutation over $[n]$.

\[
\begin{align*}
t_{i+2}(\alpha) &= [a_{i+2}, a_1, \ldots, a_i, a_{i+1}, a_{i+3}, \ldots, a_n], \\
t_i^{-1}(t_{i+2}(\alpha)) &= [a_1, a_2, \ldots, a_{i-1}, a_{i+2}, a_i, a_{i+1}, a_{i+3}, \ldots, a_n], \\
t_{i+2}(t_i^{-1}(t_{i+2}(\alpha))) &= [a_{i+1}, a_1, a_2, \ldots, a_{i-1}, a_{i+2}, a_i, a_{i+3}, \ldots, a_n], \\
t_i^{-1}(t_{i+2}(t_i^{-1}(t_{i+2}(\alpha)))) &= [a_1, a_2, \ldots, a_{i-1}, a_{i+1}, a_{i+2}, a_i, a_{i+3}, \ldots, a_n],
\end{align*}
\]
and hence we have,

$$t_{i+2}(t_i^{-1}(t_{i+2}(t_i^{-1}(t_{i+2}(\alpha))))) = [a_i, a_1, \ldots, a_{i-1}, a_{i+1}, a_{i+2}, \ldots, a_n] = t_i(\alpha).$$

**Corollary 1.** If $\pi \in S_{2n+1}$ then

$$t_{2n-1}(\pi) = t_{2n+1} \left(t_{2n-1}^{2n-2} \left(t_{2n+1}^{2n-2} \left(t_{2n+1}(\pi)\right)\right)\right).$$

Lemma 2 can be generalized as follows (the following lemma is given for completeness, but it will not be used in the sequel, and hence its proof is omitted).

**Lemma 3.** If $i, j \in [n]$ and $|i - j| = k$, then $t_i \leftrightarrow t_j \circ (t_i^{i-1} \circ t_j)^k$.

The partition of $A_{2n+1}$ into the set of classes $L_{2n+1}$ should satisfy the following properties:

(P1) The last two ordered elements of two permutations in same class are equal.

(P2) Any two permutations which differ only by a cyclic shift of the first $2n - 1$ elements, belong to the same class.

**Corollary 2.** Let $\pi$ be a permutation in $A_{2n+1}$.

- $\pi$ and $t_{2n+1}(\pi)$ belong to different classes in $L_{2n+1}$.
- $\pi$ and $t_{2n-1}(\pi)$ belong to the same class in $L_{2n+1}$.

We continue now with the description of the method to join the $K$-snakes of $L_{2n+1}$ into $C_{2n+1}$. In the rest of the paper, $A_{2n+1}$ is partitioned into classes according to the last two ordered elements in the permutations. Let $[x, y]$ denote the class of $A_{2n+1}$ in which the last ordered pair in the permutations is $(x, y)$. Let $T$ be the S-skeleton of the $K$-snakes in $L_{2n+1}$. Let $C_T^\pi$ be a $K$-snake for which $T$ is its transitions sequence, and $\pi$ is its first permutation. If $\pi$ belongs to the class $[x, y]$, we say that $C_T^\pi$ represents the class $[x, y]$. Note that all the permutations in $C_T^\pi$ belong to the same class.

The transitions sequence $T$ should satisfy the following properties (these properties are needed in order to make the required joins of cycles):

28
(P3) $t_{2n-1}$ is the last transition in $T$.

(P4) Given a permutation $\pi = [a_1, \ldots, a_{2n}, a_{2n+1}]$, for each $x \in [2n + 1] \setminus \{a_{2n}, a_{2n+1}\}$ there exists a permutation $\pi' \in C_T$ whose last ordered three elements are $(x, a_{2n}, a_{2n+1})$.

**Corollary 3.** For each class $[x, y]$, a permutation $\pi \in [x, y]$, and $z \in [2n + 1] \setminus \{x, y\}$, there exists a permutation $\pi' \in C_T$ whose last ordered three elements are $(z, x, y)$, followed by the permutation $t_{2n-1}(\pi')$.

**Lemma 4.** Let $C$ be a $K$-snake which doesn’t contain any permutation from the classes $[y, z]$ or $[z, x]$, let $\pi = [a_1, a_2, \ldots, a_{2n-2}, z, x, y]$ be a permutation in $C$ followed by $t_{2n-1}$, and let $\sigma$ be a transitions sequence such that $T = \sigma \circ t_{2n-1}$. Then replacing this $t_{2n-1}$ transition in $C$, with

$$t_{2n+1} \circ \sigma \circ t_{2n+1} \circ \sigma \circ t_{2n+1},$$

joins two $K$-snakes representing the classes $[y, z]$ and $[z, x]$ into $C$ (after $\pi$).

**Proof.** Observe that by Lemma 1 we have $\sigma \leftrightarrow t_{2n-1}$. Thus, we have

$$\pi = [a_1, a_2, \ldots, a_{2n-2}, z, x, y]$$

$$\downarrow t_{2n+1}$$

$$[y, a_1, a_2, \ldots, a_{2n-2}, z, x]$$

$$\downarrow \sigma \leftrightarrow t_{2n-1}$$

$$[a_1, a_2, \ldots, a_{2n-2}, y, z, x]$$

$$\downarrow t_{2n+1}$$

$$[x, a_1, a_2, \ldots, a_{2n-2}, y, z]$$

$$\downarrow \sigma \leftrightarrow t_{2n-1}$$

$$[a_1, a_2, \ldots, a_{2n-2}, x, y, z]$$

$$\downarrow t_{2n+1}$$

return to the $K$-snake $C$

$$t_{2n-1}(\pi) = [z, a_1, a_2, \ldots, a_{2n-2}, x, y]$$

The next step is to present an order for merging all the $K$-snakes of $L_{2n+1}$, except one, into $C_{2n+1}$. This step will be performed by translating the merging problem into a 3-graph problem. We start with a sequence of definitions taken from [7].

29
Definition 1. A 3-graph (also called a 3-uniform hypergraph) $H = (V, E)$ is a hypergraph where $V$ is a set of vertices and $E \subseteq \binom{V}{3}$. A hyperedge of $H$ will be called triple.

A path in $H$ is an alternating sequence of $\ell+1$ distinct vertices and $\ell$ distinct triples: $v_0, e_1, v_1, \ldots, v_{\ell-1}, e_\ell, v_\ell$, with the property that $\forall i \in [\ell]: v_{i-1}, v_i \in e_i$.

A cycle is a closed path, i.e. $v_0 = v_\ell$.

A sub-3-graph contains a subset $E' \subseteq E$ and the subset $V' \subseteq V$ which contains all the vertices in $E'$.

A tree $T$ in $H$ is a connected sub-3-graph of $H$ with no cycles.

Let $H_{2n+1} = (V_{2n+1}, E_{2n+1})$ be a 3-graph defined as follows:

$$V_{2n+1} = \{[x, y] : x, y \in [2n + 1], x \neq y\},$$

$$E_{2n+1} = \{\{[x, y], [y, z], [z, x]\} : x, y, z \in [2n + 1], x \neq y, x \neq z, y \neq z\}.$$

We denote a hyperedge $\{[x, y], [y, z], [z, x]\}$, where $x < y$ and $x < z$, by the triple $\langle x, y, z \rangle$.

The vertices in $H_{2n+1}$ correspond to the classes in the set $L_{2n+1}$. Each $e \in E_{2n+1}$ contains three vertices, which correspond to three classes. These three classes can be represented by three $\cal K$-snakes, generated from the $S$-skeleton, which can be merged together by Corollary 3 and Lemma 4. Note that for any two edges $e_1, e_2$ in $H_{2n+1}$ either $e_1 \cap e_2 = \emptyset$ or $|e_1 \cap e_2| = 1$.

Let $T_{2n+1} = (V_{T_{2n+1}}, E_{T_{2n+1}})$ be a tree in $H_{2n+1}$. We join $|V_{T_{2n+1}}|$ $\cal K$-snakes which represent $|V_{T_{2n+1}}|$ classes of $L_{2n+1}$ to form the $\cal K$-snake $C_{2n+1}$, by Corollary 3 and Lemma 4. The hyperedges which represent the joins which are performed are determined by $T_{2n+1}$, but these joins are not unique, and hence they can yield different final $\cal K$-snakes. The order in which the hyperedges are selected for these joins is also not unique, but this order doesn’t affect the final $\cal K$-snakes. The size of the $\cal K$-snake $C_{2n+1}$ depends on the number of vertices in the tree $T_{2n+1}$. A tree in a 3-graph contains an odd number of vertices [7]. Since in $H_{2n+1}$ there are $(2n + 1)(2n)$ vertices it follows that there is no tree in $H_{2n+1}$ which contains all the vertices of $V_{2n+1}$. This motivates the following definition.

Definition 2. A nearly spanning tree in a 3-graph $H = (V, E)$ is a tree in $H$ which contains all the vertices of $V$ except one.
Now, let $T_{2n+1}$ be a nearly spanning tree in $H_{2n+1}$.

**Example 1.** One choice for $T_5$ is given below.

The edges in the tree $T_5$ are:

- $(1, 2, 5)$, $(1, 2, 4)$, $(1, 2, 3)$,
- $(1, 4, 5), (2, 5, 4)$, $(1, 3, 4), (2, 4, 3)$, $(1, 5, 3), (2, 3, 5)$.

The order of merging $K$-snakes from these classes obtained by this choice of $T_5$ can be chosen as follows.

1. vertex $[1, 2]$;
2. vertices $[3, 1], [2, 3], (through the edge (1, 2, 3));$
3. vertices $[4, 1], [2, 4], (through the edge (1, 2, 4));$
4. vertices $[5, 1], [2, 5], (through the edge (1, 2, 5));$
5. vertices $[5, 3], [1, 5], (through the edge (1, 5, 3));$
6. vertices $[5, 2], [3, 5], (through the edge (2, 3, 5));$
7. vertices $[3, 4], [1, 3], (through the edge (1, 3, 4));$
8. vertices $[3, 2], [4, 3], (through the edge (2, 4, 3));$
9. vertices $[4, 5], [1, 4], (through the edge (1, 4, 5));$
10. vertices $[4, 2], [5, 4], (through the edge (2, 5, 4)).$

Using the S-skeleton $T = t_3, t_3, t_3$ of the $(3, 3, K)$-snake, the snake-in-the-box code which is obtained by $T_5$ is a $(5, 57, K)$-snake presented in Figure 3.1. There is no $(5, M, K)$-snake for which $M > 57$ [13]. The S-skeleton of this code is $\sigma^3$, where

$$\sigma = t_5, t_5, t_3, t_5, t_3, t_5, t_3, t_5, t_5, t_3,$$
$$t_5, t_5, t_3, t_5, t_3, t_3, t_5, t_3, t_5, t_3, t_5, t_5$$

**Theorem 1.** If $n \geq 2$, then there exists a nearly spanning tree $T_{2n+1}$ in $H_{2n+1}$ which doesn’t include the vertex $[2, 1]$. 

31
We present a recursive construction for such a nearly spanning tree. We start with the nearly spanning tree given in Example 1. Note that doesn’t include the vertex $[2, 1]$. Assume that there exists a nearly spanning tree, $T_{2n-1}$, in $H_{2n-1}$, which doesn’t include the vertex $[2, 1]$. Note that $H_{2n-1}$ is a sub-graph of $H_{2n+1}$ and therefore $T_{2n-1}$ is a tree in $H_{2n+1}$. The vertices of $H_{2n+1}$ which are not spanned by $T_{2n-1}$ are

- $[x, 2n], [2n, x], [x, 2n+1], [2n+1, x]$ for each $x \in [2n - 1]$,
- $[2n, 2n+1], [2n+1, 2n],$
- $[2, 1]$.

The nearly spanning tree $T_{2n+1}$ is constructed from $T_{2n-1}$ as follows. For each $x, 2 \leq x \leq 2n - 2$, the edges $(x, x + 1, 2n)$ and $(x, x + 1, 2n + 1)$ are joined to $T_{2n+1}$; also the edges $(1, 2, 2n), (1, 2n, 2n - 1), (1, 2n + 1, 2n - 1), (1, 2n, 2n + 1)$, and $(2, 2n + 1, 2n)$ are joined to $T_{2n+1}$. It is easy to verify that all the vertices of $H_{2n+1}$ which are not spanned by $T_{2n-1}$ (except for $[2, 1]$) are contained in the list of the edges which are joined to $T_{2n-1}$. When an edge is joined to the tree it has one vertex which is already in the tree and two vertices which are not on the tree. Hence, connectivity is preserved and no cycle is formed. Hence, it is easy to verify that by joining these edges to $T_{2n-1}$ we form a nearly spanning tree in $H_{2n+1}$.

**Example 2.** By using Theorem 1 and the nearly spanning tree $T_5$ of Example 1 we obtain the spanning tree $T_7$ depicted in Figure 3.2. The dashed boxes edges and the double lines nodes are added to $T_5$ in order to form $T_7$. 

![Figure 3.1: A (5, 57, K)-snake obtained by $T_5$](image-url)
3.4 A Recursive Construction

In this section we present the recursive construction for a \((2n+1, M_{2n+1}, \mathcal{K})\)-snake from a \((2n-1, M_{2n-1}, \mathcal{K})\)-snake. The construction is based on the nearly spanning tree \(T_{2n+1}\) presented in the previous section. Each of its vertices represent a class in which a \(\mathcal{K}\)-snake based on the \((2n-1, M_{2n-1}, \mathcal{K})\)-snake is generated. Those \(\mathcal{K}\)-snakes are merged together into one \((2n+1, M_{2n+1}, \mathcal{K})\)-snake using the framework presented in the previous section. We conclude this section with an analysing the length of the generated \(\mathcal{K}\)-snake compared the total number of permutations in \(S_{2n+1}\).

We generate a \((2n+1, M_{2n+1}, \mathcal{K})\)-snake, \(C_{2n+1}\), whose transitions sequence is \(t_{k_1}, t_{k_2}, \ldots, t_{k_{M_{2n+1}}}\). \(C_{2n+1}\) has the following properties:

(Q1) \(k_j\) is odd for all \(j \in [M_{2n+1}]\).

(Q2) \(k_{M_{2n+1}} = 2n + 1\).

(Q3) For each \(z \in [2n + 1]\) there exists a permutation \(\pi \in C_{2n+1}\) such that 
\(\pi(2n + 1) = z\).

The starting point of the recursive construction is \(2n+1 = 3\). The transitions
sequence for \(2n + 1 = 3\) is \(t_3, t_3, t_3\), and the complete \((3, 3, K)\)-snake is 
\[C_3 \triangleq \{[1, 2, 3], [3, 1, 2], [2, 3, 1]\}.\] Clearly, \(Q1\), \(Q2\), and \(Q3\) hold for this transitions sequence and \(C_3\).

Now, assume that there exists a \((2n - 1, M_{2n-1}, K)\)-snake, \(C_{2n-1}\), which satisfies properties \((Q1)\), \(Q2\), \(Q3\), and let \(T_{2n-1} = t_{k_1}, t_{k_2}, \ldots, t_{k_{f2n-1}}\) be its S-skeleton, i.e., \(T_{2n-1}\) is the transitions sequence of \(C_{2n-1}\). Note that \((Q1)\), \(Q2\), \(Q3\) depend on the transitions sequence \(T_{2n-1}\) and are independent of the first permutation of \(C_{2n-1}\). We construct a \((2n + 1, M_{2n+1}, K)\)-snake, \(C_{2n+1}\), where \(M_{2n+1} = ((2n + 1)(2n) - 1)M_{2n-1}\), which also satisfies \((Q1)\), \(Q2\), and \(Q3\).

First, all the permutations of \(A_{2n+1}\) are partitioned into \((2n + 1)(2n)\) classes according to the last ordered two elements in the permutations. This implies that \((P1)\) and \(P2\) are satisfied. In addition, \((P3)\) and \((P4)\) for \(T_{2n-1}\) are immediately implied by \(Q2\) and \(Q3\) for \(C_{2n-1}\), respectively. Hence \(T_{2n-1}\) can be used as the S-skeleton for the \(K\)-snakes in \(L_{2n+1}\). Now, we merge the \(K\)-snakes of the classes in \(L_{2n+1}\) (except \([2, 1]\)), by using Lemma 4 and the nearly spanning tree \(T_{2n+1}\) of Theorem 1. We have to show that \((Q1)\), \((Q2)\), and \((Q3)\) are satisfied for \(C_{2n+1}\). \((Q1)\) is readily verified. Clearly, \(t_{2n+1}\) was used to obtain \(C_{2n+1}\) (see Lemma 4), and therefore we can always define \(T_{2n+1}\) in such a way that its last transition is \(t_{2n+1}\), and hence \(Q2\) is satisfied. For each \(z \in [2n + 1]\) there exists a class \([x, z]\) whose \(K\)-snake is joined into \(C_{2n+1}\), and therefore \((Q3)\) is satisfied. Thus, we have

**Theorem 2.** Given a \((2n - 1, M_{2n-1}, K)\)-snake which satisfies \((Q1)\), \((Q2)\), and \((Q3)\), we can obtain a \((2n + 1, M_{2n+1}, K)\)-snake, where \(M_{2n+1} = ((2n + 1)(2n) - 1)M_{2n-1}\), which also satisfies \((Q1)\), \((Q2)\), and \((Q3)\).

Following [13], we define \(D_{2n+1} = \frac{M_{2n+1}}{(2n+1)!}\) as the ratio between the number of permutations in the given \((2n + 1, M_{2n+1}, K)\)-snake and the size of \(S_{2n+1}\). Recall that if \(C\) is an \((2n + 1, M, K)\)-snake then \(M \leq \frac{|S_{2n+1}|}{2}\), and we conjecture that optimal size is \(M = (2n + 1)! - 2n + 1\). Thus, it is desirable to obtain a value \(D_{2n+1}\) close to half as much as possible. In our recursive construction \(M_{2n+1} = ((2n + 1)(2n) - 1)M_{2n-1}\). Thus, we have

\[
D_3 = \frac{1}{2},
\]

\[
\prod_{n=2}^{\infty} \frac{D_{2n+1}}{D_{2n-1}} = \frac{12\sqrt{\pi}}{5(1 + \sqrt{5})\Gamma\left(\frac{1}{4}(5 - \sqrt{5})\right)\Gamma\left(\frac{1}{4}(1 + \sqrt{5})\right)}.
\]
which implies that
\[
\lim_{n \to \infty} D_{2n+1} = \frac{1}{2} \cdot \frac{12\sqrt{\pi}}{5(1 + \sqrt{5})\Gamma\left(\frac{1}{4}(5 - \sqrt{5})\right)\Gamma\left(\frac{1}{4}(1 + \sqrt{5})\right)} 
\approx 0.4338.
\]

This computation can be done by any mathematical tool, e.g., WolframAlpha. This improves on the construction described in [13], which yields
\[
M_{2n+1} = (2n + 1)(2n - 1)M_{2n-1} \quad \text{and} \quad \lim_{n \to \infty} D_{2n+1} = \frac{1}{\sqrt{\pi n}}.
\]

### 3.5 A Direct Construction based on Necklaces

In this section we describe a direct construction to form a \((2n+1, M_{2n+1}, \mathcal{K})\)-snake. First, we describe a method to partition the classes which were used before into subclasses that are similar to necklaces. Next, we show how subclasses from different classes are merged into disjoint chains. Finally, we present a hypergraph and a graph in which we have to search for certain trees to form our desired \(\mathcal{K}\)-snake which we believe is of maximum length. Such \(\mathcal{K}\)-snakes were found in \(S_7\) and \(S_9\).

We present a direct construction for a \((2n + 1, M_{2n+1}, \mathcal{K})\)-snake, \(C_{2n+1}\). The goal is to obtain \(M_{2n+1} = \frac{(2n+1)!}{2} - (2n-1)\), and hence \(D_{2n+1} = 1 - \frac{1}{(2n)!}\). We believe that there is always a \((2n + 1, M_{2n+1}, \mathcal{K})\)-snake with \(M_{2n+1} = \frac{(2n+1)!}{2} - (2n - 1)\) and there is no such \(\mathcal{K}\)-snake with more codewords. We are making a slight change in the framework discussed in Section 3.3. First, all the permutations of \(A_{2n+1}\) are partitioned into \((2n + 1)(2n)\) classes according to the last ordered two elements. We denote by \([x, y]\) the class of all even permutations in which the last ordered pair in the permutation is \((x, y)\).

Each class is further partitioned into subclasses according to the cyclic order of the first \(2n - 1\) elements in the permutations, i.e., in each class \([x, y]\), the \(\frac{(2n-1)!}{2}\) permutations are partitioned into \(\frac{(2n-2)!}{2}\) disjoint subclasses. This implies that \((P1)\) and \((P2)\) are satisfied for both classes and subclasses. Let’s denote each one of the subclasses by \([\alpha] - [x, y]\) where \(\alpha\) is the cyclic order of the first \(2n - 1\) elements in the permutations of the subclass. Let \(\alpha_1, \alpha_2\) be two permutations over \([2n + 1] \setminus \{x, y\}\). If \(\alpha_1\) and \(\alpha_2\) have the same cyclic order, we denote it by \(\alpha_1 \simeq \alpha_2\), otherwise \(\alpha_1 \not\simeq \alpha_2\). Note that if \(\alpha_1 \simeq \alpha_2\) then \([\alpha_1] - [x, y] = [\alpha_2] - [x, y]\). For example \([1, 2, 3] - [4, 5]\) represents the
subclass with the permutations \([1, 2, 3, 4, 5], [3, 1, 2, 4, 5], \) and \([2, 3, 1, 4, 5]\).

Let \(L_{2n+1}\) be the set of all classes, and let \(T = t_{2n-1}^{2n-1}\) be the S-skeleton of the \(K\)-snakes in \(L_{2n+1}\). Note that a \(K\)-snake generated by \(T\) spans exactly all the permutations in one subclass. Hence (P3) and (P4) are immediately implied for both classes and subclasses. Such a \(K\)-snake will be called a *necklace*. The slight change in the framework is that instead of one \(K\)-snake, each class contains \(\left(\frac{(2n-2)!}{2}\right)\) \(K\)-snakes, all of them having the same S-skeleton.

The necklaces (subclasses) \([\alpha] - [x, y]\) are similar to necklaces on \(2n-1\) elements. Joining the necklaces into one large \(K\)-snake might be similar to the join of cycles from the pure cycling register of order \(2n - 1\), PCR_{2n-1}, into one cycle, which is also known as a de Bruijn sequence [2, 4]. There are two main differences between the two types of necklaces. The first one is that in de Bruijn sequences the necklaces do not represent permutations, but words of a given length over some finite alphabet. The second is that there is rather a simple mechanism to join all the necklaces into a de Bruijn sequence. We would like to have such a mechanism to join as many as possible necklaces from all the classes into one \(K\)-snake.

Let \(T_{2n+1}\) be the nearly spanning tree constructed by Theorem 1. By repeated application of Lemma 4 according to the hyperedges of \(T_{2n+1}\) starting from a necklace in the class \([1, 2]\) we obtain a \(K\)-snake which contains exactly one necklace from each class \([x, y] \neq [2, 1]\). Such a \(K\)-snake will be called a *chain*. If the chain contains the necklace \([\alpha] - [1, 2]\), we will denote it by \(c[\alpha]\). For two permutations \(\alpha_1\) and \(\alpha_2\) over \([2n+1] \setminus \{1, 2\}\) such that \(\alpha_1 \simeq \alpha_2\) we have \(c[\alpha_1] = c[\alpha_2]\). Note that there is a unique way to merge the three necklaces which correspond to a hyperedge of \(T_{2n+1}\), and hence there is no ambiguity in \(c[\alpha]\) (even so the order of the joins is not unique). Note also that the transitions sequence of two distinct chains is usually different. The number of permutations in a chain is \((2n+1)(2n-1)(2n-1)\). The following lemma is an immediate consequence of Lemma 4.

**Lemma 5.** Let \([x, y], [y, z], \) and \([z, x]\) be three classes, and let \(\alpha\) be a permutation of \([2n+1] \setminus \{x, y, z\}\). The necklaces \([\alpha, z] - [x, y], [\alpha, y] - [z, x], \) and \([\alpha, x] - [y, z]\) can be merged together, where \(\alpha, z\) is the sequence formed by concatenation of \(\alpha\) and \(z\).

**Lemma 6.** Let \([x, y], [y, z], \) and \([z, x]\) be three classes. All the subclasses in these classes can be partitioned into disjoint sets, where each set contains
exactly one necklace from each of the above three classes. The necklaces of each set can be merged together into one $K$-snake.

Proof. For each permutation $\alpha$ over $[2n+1] \setminus \{x, y, z\}$, the necklaces $[\alpha, z] - [x, y]$, $[\alpha, y] - [z, x]$, and $[\alpha, x] - [y, z]$ can be merged by Lemma 5. Thus, all the subclasses in these classes can be partitioned into disjoint sets. 

Corollary 4. The permutations of all the classes except for $[2, 1]$ can be partitioned into disjoint chains.

By Corollary 4 we construct $\frac{(2n-2)!}{2}$ disjoint chains which span $A_{2n+1}$, except for all the even permutations of the class $[2, 1]$. Recall that we have the same number, $\frac{(2n-2)!}{2}$, of $[2, 1]$-necklaces, which span all the permutations of the class $[2, 1]$. Now, we need a method to merge all these chains and necklaces, except for one necklace from the class $[2, 1]$, into one $K$-snake $C_{2n+1}$. Note that for $2n + 1 = 5$ we have only one chain. Thus, this chain is the final $K$-snake $C_5$. This $K$-snake is exactly the same $K$-snake as the one generated by the recursive construction in Section 3.4.

Lemma 7. Let $x$ be an integer such that $3 \leq x \leq 2n + 1$, let $\alpha$ be a permutation of $[2n+1] \setminus \{x, 2, 1\}$, and assume that the permutations $[\alpha, 1, x, 2]$ and $[\alpha, 2, 1, x]$ are contained in two distinct chains. We can merge these two chains via the necklace $[\alpha, x] - [2, 1]$.

Proof. Let $c_1$ be the chain which contains the permutation $\pi_1 = [\alpha, 1, x, 2]$, $c_2$ be the chain which contains the permutation $\pi_2 = [\alpha, 2, 1, x]$, and $\eta$ be the necklace which contains the permutation $\pi_3 = [\alpha, x, 2, 1]$. Note that all the chains contain only the $p$-transitions $t_{2n+1}$ and $t_{2n-1}$. The permutation $t_{2n+1}(\pi_1)$ appears in $c_2$, the permutation $t_{2n+1}(\pi_2)$ appears in $\eta$, and the permutation $t_{2n+1}(\pi_3)$ appears in $c_1$. Therefore, $\pi_1$, $\pi_2$, and $\pi_3$ are followed by $t_{2n-1}$ in $c_1$, $c_2$, and $\eta$, respectively. Let $\sigma_i$, $i \in \{1, 2\}$, be a transitions sequence such that $\sigma_i \circ t_{2n-1}$ is the transitions sequence of $c_i$, and therefore $t_{2n-1}(\sigma_i(\pi_i)) = \pi_i$. By Lemma 1 we have $\sigma_1 \sim t_{2n-1}^{2n-2} \sim \sigma_2$. Similarly to Lemma 4, by replacing the transition $t_{2n-1}$ which follows $\pi_3$ in $\eta$, with $t_{2n+1} \circ \sigma_1 \circ t_{2n+1} \circ \sigma_2 \circ t_{2n+1}$, we merge $c_1$, $c_2$ and $\eta$ into a $K$-snake. Thus, we have
\[ \pi_3 = [a_1, a_2, \ldots, a_{2n-2}, x, 2, 1] \]

\[ \downarrow t_{2n+1} \]

\[ [1, a_1, a_2, \ldots, a_{2n-2}, x, 2] \]

\[ \downarrow \sigma_1 \leftrightarrow t_{2n-1}^{2n-2} \]

the chain \( c_1 \)

\[ \pi_1 = [a_1, a_2, \ldots, a_{2n-2}, 1, x, 2] \]

\[ \downarrow t_{2n+1} \]

\[ [2, a_1, a_2, \ldots, a_{2n-2}, 1, x] \]

\[ \downarrow \sigma_2 \leftrightarrow t_{2n-1}^{2n-2} \]

the chain \( c_2 \)

\[ \pi_2 = [a_1, a_2, \ldots, a_{2n-2}, 2, 1, x] \]

\[ \downarrow t_{2n+1} \]

return to the necklace \( \eta \)

\[ t_{2n-1}(\pi_3) = [x, a_1, a_2, \ldots, a_{2n-2}, 2, 1] \]

For each \( x, 3 \leq x \leq 2n + 1 \), and for each permutation \( \alpha \) of \([2n+1] \setminus \{x, 1, 2\}\), the merging of two distinct chains which contain the permutations \([\alpha, 1, x, 2]\) and \([\alpha, 2, 1, x]\) via the necklace \([\alpha, x] - [2, 1]\) as described in Lemma 7, will be denoted by \(M[x]\)-connection. Note that if \( x \in \{3, 4, 5\} \) then the permutations \([\alpha, 1, x, 2]\) and \([\alpha, 2, 1, x]\) are contained in the same chain. Thus, there are no \(M[3]\)-connections, \(M[4]\)-connections, or \(M[5]\)-connections.

Lemma 7 suggests a method to join all the chains and all the \([2, 1]\)-necklaces except one into a \(K\)-snake of length \(\frac{(2n+1)!}{2} - (2n - 1)\). This should be implemented by \(\frac{(2n-2)!}{2} - 1\) iterations of the merging suggested by Lemma 7. The current merging problem is also translated into a 3-graph problem (see Definition 1). Let \(\hat{H}_{2n+1} = (\hat{V}_{2n+1}, \hat{E}_{2n+1})\) be a 3-graph defined as follows.

\[ \hat{V}_{2n+1} = \{c[\alpha] : \alpha \text{ is a permutation of } [2n+1] \setminus \{1, 2\}\} \]

\[ \cup \{[\beta] - [2, 1] : \beta \text{ is a permutation of } [2n+1] \setminus \{1, 2\}\} \]

\[ \hat{E}_{2n+1} = \{c[\alpha_1], c[\alpha_2], [\beta] - [2, 1] : c[\alpha_1] \text{ and } c[\alpha_2] \text{ can be merged together via } [\beta] - [2, 1] \text{ by Lemma 7}\} \]
The vertices in $\hat{V}_{2n+1}$ are of two types, chains and $[2,1]$-necklaces. Each $e \in \hat{E}_{2n+1}$ contains three vertices, two chains and one necklace, which can be merged together by Lemma 7. Therefore, the edge will be signed by $M[x]$ as described before. Note that $\hat{E}_{2n+1}$ might contain parallel edges with different signs.

Let $\hat{T}_{2n+1} = (V_{T_{2n+1}}, E_{T_{2n+1}})$ be a nearly spanning tree in $\hat{H}_{2n+1}$. Note that such a nearly spanning tree must contain all the vertices in $\hat{V}_{2n+1}$ except for one $[2,1]$-necklace. If such a nearly spanning tree exists then by Lemma 7, we can merge all the chains via $[2,1]$-necklaces to form the $\mathcal{K}$-snake $C_{2n+1}$. This $\mathcal{K}$-snake contains all the permutations of $A_{2n+1}$ except for $2n - 1$ permutations which form one $[2,1]$-necklace.

The joins which are performed are determined by the edges of $\hat{T}_{2n+1}$. Note that there is a unique way to merge the three vertices which correspond to a hyperedge of $\hat{T}_{2n+1}$ signed by $M[x]$. Hence, by using the given spanning trees $T_{2n+1}$ and $\hat{T}_{2n+1}$, there is no ambiguity in $C_{2n+1}$ (even so the orders of the joins are not unique). However, different nearly spanning trees can yield different final $\mathcal{K}$-snakes. Note that the $\mathcal{K}$-snake $C_{2n+1}$ generated by this construction has only $t_{2n+1}$ and $t_{2n-1}$ p-transitions, where usually $t_{2n-1}$ is used. The p-transition $t_{2n-1}$ is the only transition in the $\mathcal{K}$-snake of the subclasses. On average 3 out of $4n - 2$ sequential p-transitions of $C_{2n+1}$ are the p-transition $t_{2n+1}$. A similar property exists when a de Bruijn sequence is generated from the necklaces of pure cycling register of order $n$ [2, 4].

Finding a nearly spanning tree $\hat{T}_{2n+1}$ is an open question. But, we found such trees for $n = 3$ and $n = 4$. We believe that a similar construction to the one which follows in the sequel for $n = 3$ and $n = 4$, exists for all $n > 4$.

**Conjecture 1.** For each $n \geq 2$, there exists a $(2n + 1, M_{2n+1}, \mathcal{K})$-snake, where $M_{2n+1} = \frac{(2n+1)!}{2} - (2n - 1)$ in which there are only $t_{2n-1}$ and $t_{2n+1}$ p-transitions.

**Example 3.** For $n = 3$, a $(7, 2515, \mathcal{K})$-snake is constructed by using the tree $T_7$ of Example 2, and the tree $\hat{T}_7$ defined below. $\hat{T}_7$ contains 12 chains, where each chain contains 41 necklaces. It also contains 11 $[2,1]$-necklaces and 11 hyperedges. Denote an edge in $\hat{H}_7$ by $\{(c_i, c_j, \eta_k), x\}$ where $M[x]$ is the sign of the edge. $\hat{T}_7$ is defined as follows.
The chains in $\hat{T}_7$:

$c_1 = [3, 4, 5, 6, 7] - [1, 2], \quad c_2 = [3, 4, 6, 7, 5] - [1, 2],

$c_3 = [3, 4, 7, 5, 6] - [1, 2], \quad c_4 = [3, 5, 4, 7, 6] - [1, 2],

$c_5 = [3, 5, 6, 4, 7] - [1, 2], \quad c_6 = [3, 5, 7, 6, 4] - [1, 2],

$c_7 = [3, 6, 4, 5, 7] - [1, 2], \quad c_8 = [3, 6, 5, 7, 4] - [1, 2],

$c_9 = [3, 6, 7, 4, 5] - [1, 2], \quad c_{10} = [3, 7, 4, 6, 5] - [1, 2],

$c_{11} = [3, 7, 5, 4, 6] - [1, 2], \quad c_{12} = [3, 7, 6, 5, 4] - [1, 2].$

The necklaces in $\hat{T}_7$:

$\eta_1 = [3, 4, 5, 7, 6] - [2, 1], \quad \eta_2 = [3, 4, 6, 5, 7] - [2, 1],

$\eta_3 = [3, 4, 7, 6, 5] - [2, 1], \quad \eta_4 = [3, 5, 4, 6, 7] - [2, 1],

$\eta_5 = [3, 5, 6, 7, 4] - [2, 1], \quad \eta_6 = [3, 5, 7, 4, 6] - [2, 1],

$\eta_7 = [3, 6, 4, 7, 5] - [2, 1], \quad \eta_8 = [3, 6, 5, 4, 7] - [2, 1],

$\eta_9 = [3, 6, 7, 5, 4] - [2, 1], \quad \eta_{10} = [3, 7, 4, 5, 6] - [2, 1],

$\eta_{11} = [3, 7, 5, 6, 4] - [2, 1].$

The edges in $\hat{T}_7$:

$e_1 = (\{c_{11}, c_6 , \eta_9 \}, 6), \quad e_2 = (\{c_6 , c_1 , \eta_2 \}, 6),$

$e_3 = (\{c_2 , c_{12} , \eta_{11} \}, 6), \quad e_4 = (\{c_{12} , c_7 , \eta_4 \}, 6),$

$e_5 = (\{c_5 , c_3 , \eta_3 \}, 6), \quad e_6 = (\{c_3 , c_4 , \eta_7 \}, 6),$

$e_7 = (\{c_9 , c_{10} , \eta_{10} \}, 6), \quad e_8 = (\{c_{10} , c_8 , \eta_5 \}, 6),$

$e_9 = (\{c_{12} , c_9 , \eta_8 \}, 7), \quad e_{10} = (\{c_9 , c_3 , \eta_1 \}, 7),$

$e_{11} = (\{c_2 , c_{11} , \eta_6 \}, 7).$

$\hat{H}_7$ contains another $[2, 1]$-necklace, $\eta_{12} = [3, 7, 6, 4, 5] - [2, 1]$, and the following additional edges:

$e_{12} = (\{c_1 , c_{11} , \eta_{12} \}, 6), \quad e_{13} = (\{c_7 , c_2 , \eta_1 \}, 6),$

$e_{14} = (\{c_4 , c_5 , \eta_8 \}, 6), \quad e_{15} = (\{c_8 , c_9 , \eta_6 \}, 6),$

$e_{16} = (\{c_{10} , c_2 , \eta_2 \}, 7), \quad e_{17} = (\{c_8 , c_1 , \eta_3 \}, 7),$

$e_{18} = (\{c_{11} , c_{10} , \eta_4 \}, 7), \quad e_{19} = (\{c_3 , c_{12} , \eta_5 \}, 7),$

$e_{20} = (\{c_6 , c_7 , \eta_7 \}, 7), \quad e_{21} = (\{c_4 , c_8 , \eta_9 \}, 7),$

$e_{22} = (\{c_1 , c_4 , \eta_{10} \}, 7), \quad e_{23} = (\{c_5 , c_6 , \eta_{11} \}, 7),$

$e_{24} = (\{c_7 , c_5 , \eta_{12} \}, 7).$

An additional different illustration of $\hat{H}_7$ is presented in the sequel (see Example 4).

For each $n \geq 3$, let $G_{2n+1} = (V_{2n+1}, E_{2n+1})$ be a multi-graph (with parallel edges) with labels and signs on the edges. The vertices of $V_{2n+1}$ represent the $\binom{2(n-2)}{2}$ chains and hence $|V_{2n+1}| = \frac{(2n-2)!}{2}$. There is an edge signed with $M[x]$, where $6 \leq x \leq 2n + 1$, between the vertex (chain) $c_1$ and
vertex (chain) $c_2$, if $c_1$ contains a permutation $[\alpha, 2, 1, x]$ and $c_2$ contains the permutation $[\alpha, 1, x, 2]$, where $c_1 \neq c_2$. The label on this edge is the necklace $[\alpha, x] - [2, 1]$. Note that the label on the edge is a necklace which can merge together the chains of its corresponding endpoints by $M[x]$-connection. Note also that the pair $\alpha, x$ might not be unique and hence the graph might have parallel edges. A tree in $G_{2n+1}$ which doesn't have two edges with the same label, will be called a chain tree. The following Lemma can be easily verified.

**Lemma 8.** There exists a nearly spanning tree in $\hat{H}_{2n+1}$ if and only if there exists a chain tree in $G_{2n+1}$.

Henceforth, $T_{2n+1}$ will be the nearly spanning tree constructed in Theorem 1, and the chains are constructed via $T_{2n+1}$.

**Definition 3.** Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two multi-graphs with labels and signs on the edges, where the set of the labels of $G_i$ denoted by $L_i, i \in \{1, 2\}$. We say that $G_1$ is isomorphic to $G_2$ if there exist two bijective functions $f : V_1 \rightarrow V_2$ and $g : L_1 \rightarrow L_2$, with the following property: $(u, v) \in E_1$ with the label $\eta$ and sign $M[x]$, if and only if $(f(u), f(v)) \in E_2$ with the label $g(\eta)$ and sign $M[x]$.

**Definition 4.** For each $n \geq 4$, a sub-graph of $G_{2n+1}$ which is isomorphic to $G_{2n-1}$ is called a component of $G_{2n+1}$, and denoted by $A = (V_A, L_A)$ where $V_A$ consists of the vertices (chains) of the component, $L_A$ consists of the labels ([2, 1]-necklaces) on the edges in the component. Note that $|V_A| = |L_A|$, i.e., the numbers of the distinct labels is equal to the number of the vertices.

**Definition 5.** Two components, $A = (V_A, L_A)$ and $B = (V_B, L_B)$, in $G_{2n+1}$ are called disjoint if $V_A \cap V_B = \emptyset$ and $L_A \cap L_B = \emptyset$, i.e., there is no a common vertex (chain) or a common label ([2, 1]-necklace) in $A$ and $B$.

**Lemma 9.** For each $n \geq 4$, $G_{2n+1}$ consists of $(2n-3)(2n-2)$ disjoint copies of isomorphic graphs to $G_{2n-1}$, called components. The edges between the vertices of two distinct components are signed only with $M[2n]$ and $M[2n+1]$.

**Proof.** The $M[x]$-connections are deduced by the tree $T_{2n+1}$, which was used for the construction of the chains. In particular, the path between the vertices $[1, x]$ and $[x, 2]$ in $T_{2n+1}$ determines the $M[x]$-connections in $G_{2n+1}$.
By Theorem 1, $T_{2n-1}$ is a sub-graph of $T_{2n+1}$. Therefore, for each $x, x \geq 3$, the path between the vertices $[1, x]$ and $[x, 2]$ in $T_{2n+1}$ is equal to the path between the vertices $[1, x]$ and $[x, 2]$ in $T_{2k+1}$ for each $x \leq 2k + 1 \leq 2n + 1$. The number of the vertices (chains) in $G_{2n+1}$ is equal to $\frac{(2n-2)!}{2}$, and each component contains $\frac{(2n-4)!}{2}$ vertices. Thus, $G_{2n+1}$ consists of $(2n-3)(2n-2)$ disjoint copies of isomorphic graphs to $G_{2n-1}$ connected by edges signed only with $M[2n]$ and $M[2n + 1]$. 

For each $n \geq 4$, let $\hat{G}_{2n+1} = (\hat{V}_{2n+1}, \hat{E}_{2n+1})$ be the component graph of $G_{2n+1}$. The vertices of $\hat{V}_{2n+1}$ represent the components of $G_{2n+1}$. There is an edge signed with $M[x], x \in \{2n, 2n+1\}$, between the vertices (components) $A$ and $B$, if the chain that contains the permutation $[\alpha, 2, 1, x]$ is contained in $A$, and the chain that contains the permutation $[\alpha, 1, x, 2]$ is contained in $B$. The label on this edge is the necklace $[\alpha, x] - [2, 1]$. We define $\hat{G}_7$ to be $G_7$, i.e., each component of $\hat{G}_7$ consists of exactly one chain (and also one distinct $[2, 1]$-necklace in order to follow the properties of $G_{2n+1}$).

**Definition 6.** A components spanning tree, $\hat{T}_{2n+1}$ is a spanning tree in $\hat{G}_{2n+1}$, where in the set of the labels of the tree’s edges, there are no two labels from the same component, i.e., each label in the set of the labels of the tree’s edges belongs to a different component.

**Example 4.** $\hat{G}_7$ is depicted in Figure 3.3, where the vertices numbers and the edges labels corresponds to the chains and the necklaces in Example 3, respectively. The vertical edges are signed with $M[6]$, while the horizontal edges are signed with $M[7]$. The double lines edges correspond to the edges of $\hat{T}_7$.

![Figure 3.3: The graph $\hat{G}_7$ and its component spanning tree $\hat{T}_7$](image)
Conjecture 2. For each component $A$ in $\hat{G}_{2n+1}$, $n \geq 3$, and for each label $\eta$ of $A$, there exists a components spanning tree, where there is no edge in the tree with the label $\eta$.

Conjecture 2 implies Conjecture 1, i.e.,

Theorem 3. If Conjecture 2 is true then for each $n \geq 2$, there exists a $(2n + 1, M_{2n+1}, K)$-snake, where $M_{2n+1} = \frac{(2n+1)!}{2} - (2n - 1)$ in which there are only $t_{2n-1}$ and $t_{2n+1}$ p-transitions.

Conjecture 2 was verified by computer search for $n = 3$ and $n = 4$. By using Conjecture 2 recursively, for each $n \geq 3$, and for each necklace $\eta$ in class $[2, 1]$, we can construct a chain tree $T$ in $\hat{G}_{2n+1}$, which doesn’t include $\eta$ as a label on an edge in $T$.

Corollary 5. There exist a $(7, 2515, K)$-snake and a $(9, 181433, K)$-snake, and hence $\lim_{n \to \infty} \frac{M_{2n+1}}{|S_{2n+1}|} \approx 0.4743$.

Note that the ratio $\lim_{n \to \infty} \frac{M_{2n+1}}{|S_{2n+1}|}$ would be improved, if there exists a $(2m + 1, \frac{(2m+1)!}{2} - (2m - 1), K)$-snake for some $m > 4$.

Conjecture 3. The $(2n - 3)(2n - 2)$ components in $\hat{G}_{2n+1}$ can be arranged in a $(2n - 3) \times (2n - 2)$ grid. The edges which are sign by $M[2n]$ define $2n - 2$ cycles of length $2n - 3$. Each cycle contains the vertices of exactly one column, and is called an $M[2n]$-cycle. The edges which are sign with $M[2n + 1]$ are between two components in different columns, and they also define $2n - 2$ cycles of length $2n - 3$. Such a cycle will be called an $M[2n+1]$-cycle. Each multi-edge between two components has $\frac{(2n-4)!}{2}$ parallel edges (the number of chains in the component). Parallel edges have the same sign $x$, $x \in \{2n, 2n + 1\}$, but different labels (i.e., $M[x]$-connection, but with different $[2, 1]$-necklaces).

Example 5. An illustration for the structure of $\hat{G}_{2n+1}$ for $n = 3$ is presented in Example 4, and for $n = 4$ is depicted in Figure 3.4. In $\hat{G}_9$ there are 30 components, where each component is isomorphic to $\hat{G}_7$ (thus, it contains 12 chains and 12 $[2, 1]$-necklaces).
3.6 Conclusions and Future Research

Gray codes for permutations using the operation push-to-the-top and the Kendall’s \( \tau \)-metric were discussed. We have presented a framework for constructing snake-in-the-box codes for \( S_n \). The framework for the construction yield a recursive construction with large snakes. A direct construction to obtain snakes which might be optimal in length was also presented. Several questions arise from our discussion and they are considered for current and future research.

1. Complete the direct construction for snakes of length \( \frac{(2n+1)!}{2} - 2n + 1 \) in \( S_{2n+1} \).

2. Can a snake in \( S_{2n+1} \) have size larger than \( \frac{(2n+1)!}{2} - 2n + 1 \)?

3. Prove or disprove that the length of the longest snake in \( S_{2n} \) is not longer than the length of the longest snake in \( S_{2n-1} \).

4. Examine the questions in this paper for the \( \ell_\infty \) metric.

Acknowledgment

The authors would like to thank the anonymous reviewers for their careful reading of the paper. Especially, one of the reviewers pointed out on the good ratio \( \lim_{n \to \infty} \frac{M_{2n+1}}{|S_{2n+1}|} \approx 0.4338 \) compared to one in [13].
Bibliography


errors in the rank-modulation scheme,” IEEE Transactions on Informa-

[10] M. Kendall and J. D. Gibbons, Rank Correlation Methods, New York:


levels problem,” Journal of Combinatorial Theory, Series A, vol. 70,

5483, August 2012.
Chapter 4

On the Capacity of Write-Once Memories

Michal Horovitz and Eitan Yaakobi

A Note

After Paper [20], which is introduced in this chapter, was published, a mistake was found in Figure 4.3 by Mr. Gilad Baruch. In the following, the fixed version is presented.

Abstract

Write-once memory (WOM) is a storage device consisting of $q$-ary cells that can only increase their value. A WOM code is a coding scheme which allows to write multiple times to the memory without decreasing the levels of the cells. In the conventional model, it is assumed that the encoder can read the memory state before encoding, while the decoder reads only the memory state after encoding. However, there are three more models in this setup which depend on whether the encoder and decoder are informed or uninformed with the previous state of the memory. These four models were first introduced by Wolf et al., where they extensively studied the WOM capacity in these models for the binary case. In the non-binary setup only the model, in which the encoder is informed and the decoder is not, was studied by Fu.
and Vinck. In this paper, we first present constructions of WOM codes in the models where the encoder is uninformed with the memory state (that is, the encoder cannot read the memory prior to encoding). We then study the capacity regions and maximum sum-rates of non-binary WOM codes for all four models. We extend the results by Wolf et al. and show that the capacity regions for the models in which the encoder is informed and the decoder is informed or uninformed in both the $\epsilon$-error and the zero-error cases are all identical. We also find the $\epsilon$-error capacity region in case the encoder is uninformed and the decoder is informed and show that, in contrary to the binary case, it is a proper subset of the capacity region in the first two models. Several more results on the maximum sum-rate are presented as well.

4.1 Introduction

Write-once memory (WOM) is a storage medium consisting of cells that can only increase their level. WOM codes were first introduced by Rivest and Shamir [19] and were designed to record data more than once in a WOM. Examples of such storage media are punch cards and optical disks, and more recently flash memories. The goal in designing a WOM code is to maximize the total number of bits which are written to the memory in $t$ writes, while the cells can only increase their level. The rate on each write is the ratio between the number of bits stored in the memory and the number of cells, and the sum-rate is the sum of all individual rates. The capacity region of the WOM is the set of all achievable rate tuples.

It is usually assumed that the encoder can read the current state of the cells before programing, while the decoder has access only to the state of the cells after programming but not before that. This is the most practical model in which the encoder reads the memory before encoding, and the decoder reads the memory only after the encoding ends. However, there are four models of WOM which depend on whether the encoder and decoder on each write are informed with the previous state of the memory before encoding [26].

The case where the encoder is informed with the previous state of the memory is called Encoder Informed (EI) and otherwise Encoder Uninformed (EU). The cases of Decoder Informed (DI) and Decoder Uninformed (DU)
are defined similarly. For shorthand, we refer to these four models as follows: model 1 - EI:DI, model 2 - EI:DU, model 3 - EU:DI, and model 4 - EU:DU. Note that all these models can be seen as a special case of coding for communication over a discrete memoryless channel (DMC) with state, where the state is known/unknown to the encoder and decoder; for more details see [25].

The model which was mostly studied in the literature is model 2 due to its practical relevance; see e.g. [3, 4, 10, 11, 13, 25, 27]. From the information-theoretic point of view, model 1 is the easiest one, while the most difficult one is model 4 since in model 4, both the encoder and decoder cannot read the memory before a new message is programmed. However, model 4 has significant practical advantage as it provides fast programming by saving an additional read before a write. Additionally, models 3 and 4 can be used for the construction of RIO codes, which are designed for fast programming and reading in flash memories [20, 28].

The binary case of these four models was rigorously defined and studied by Wolf et al. in [26]. The authors studied the capacity regions and maximum sum-rates for the four models in this case. In particular, they calculated the $\epsilon$-error and the zero-error capacity regions in models 1 and 2 and showed that they are all identical and thus the maximum sum-rates in these cases as well. They also showed that this is the capacity region for model 3 for the $\epsilon$-error case. Note that the zero-error capacity region and the maximum sum-rate in model 3 are still unknown. The $\epsilon$-error capacity region for model 4 was partially solved by an achievable region (i.e., it is not known whether it is a tight region), however, it was still possible to calculate its maximum sum-rate [26]. For example, for two-write binary WOM in model 4, the maximum sum-rate is 1.3881 and for three writes, it is 1.600.

Much less is known in the non-binary case, where only the zero-error capacity region for model 2 was calculated by Fu and Vinck [8] and the maximum sum-rate was shown to be $\log\left(\frac{q-1+t}{q-1}\right)$. In [10], Gabrys and Dolecek investigated the rates achieved in each write in capacity achieving WOM codes in model 2, and the distribution of symbols in such WOM codes. They also studied the fixed-rate case, where the rates in all writes are equal. Noisy WOM in model 2 was studied by Heegrad [13] and by Wang and Kim [25]. In [25], the authors investigated the capacity region and maximum sum-rate

\footnote{All logarithms in this paper are taken according to base 2.}
in this model, where in [13] a more general case was considered. In this paper we assume that the WOM is noiseless.

We study all four models. First, constructions for WOM codes are proposed for models 3 and 4, the models in which the encoder is uninformed. For model 4, where the decoder is uninformed with the previous state, we show how codes in the $Z$ channel provide constructions of binary WOM codes. This result is extended for non-binary WOM codes in which codes correcting non-binary asymmetric errors are used. Similarly in model 3, erasure-correcting codes are invoked for such constructions, and in the non-binary setup we use codes for handling asymmetric erasures. For the non-binary case in these two models we use codes for the Manhattan distance.

We also study the capacity regions of non-binary WOM for models 1, 2, and 3, and generalize some of the results by Wolf et al. for the non-binary setup. We first show that the same property of binary WOM in models 1 and 2 holds for non-binary as well. That is, the $\epsilon$-error and the zero-error capacity regions for models 1 and 2 are all the same. Then, we notice and show that in contrast to the binary case, model 3 for non-binary does not behave the same as models 1 and 2 and its $\epsilon$-error capacity region is a proper subset of the capacity regions in models 1 and 2. Furthermore, its maximum sum-rate is also smaller than the one for models 1 and 2. We derive more results to get upper and lower bounds on the maximum sum-rates in models 3 and 4.

The rest of this paper is organized as follows. In Section 4.2, we formally define all four possible models of WOM and discuss existing results. We follow by presenting constructions of WOM codes for model 4 in Section 4.3 and for model 3 in Section 4.4. In Section 4.5, we prove that the capacity regions of non-binary WOM in the first two models, both for the $\epsilon$-error and the zero-error cases, are equal. Thus these four regions are all equal to the capacity region which was studied in [8] for non-binary WOM in the zero-error case of model 2. In Section 4.6, we calculate the $\epsilon$-error capacity region of non-binary WOM in model 3, which in Section 4.7, is proved to be a proper subset of the capacity region in models 1 and 2. Additionally, Section 4.7 includes results regarding the maximum sum-rates in the four models. Lastly, Section 4.8 concludes the paper and lists some open problems.
4.2 Definitions and Preliminaries

In this section we formally define the models studied in the paper and review the related previous results. The memory consists of \( n \) q-ary cells, where initially all of them are in the zero state. On each write, it is only possible to increase the level of each cell. For a positive integer \( q \), the set \( \{0, \ldots, q-1\} \) is denoted by \([q]\). A vector \( \mathbf{c} \in [q]^n \) will be called also a cell-state vector. The vector \( \mathbf{c} = \max(\mathbf{c}_1, \mathbf{c}_2) \) is given by \( c_i = \max\{c_{1,i}, c_{2,i}\} \) for all \( i \in [n] \). For two vectors \( \mathbf{x}, \mathbf{y} \in [q]^n \), we say that \( \mathbf{x} \leq \mathbf{y} \) if for all \( i \in [n] \), \( x_i \leq y_i \) and \( \mathbf{x} < \mathbf{y} \) is defined similarly. For a vector of probabilities \( \mathbf{p} = (p_1, \ldots, p_n) \), \( H(\mathbf{p}) \) denotes the entropy function, \( H(\mathbf{p}) = -\sum_{i=1}^{n} p_i \log p_i \).

There are four families of WOM codes which were defined first in [26]. We follow the definition presented in [15] for both the zero-error and the \( e \)-error cases.

**Definition 7.** A q-ary \( t \)-write WOM code with error probability vector \( \mathbf{p}_e = (p_{e1}, \ldots, p_{et}) \), denoted by \([n,t; M_1, \ldots, M_t]_{[q]}^{\mathbf{p}_e} \), for \( k \in \{1, 2, 3, 4\} \), is a coding scheme comprising of \( n \) q-ary cells and is defined by \( t \) encoding and decoding maps \( \mathcal{E}_i, \mathcal{D}_i \). For the map \( \mathcal{E}_i, \operatorname{Im}(\mathcal{E}_i) \) is the image of the map. By definition \( \operatorname{Im}(\mathcal{E}_0) = \{(0, \ldots, 0)\} \), and for \( i, 1 \leq i \leq t, \operatorname{Im}^*(\mathcal{E}_i) = \{\max(\mathbf{c}_1, \ldots, \mathbf{c}_i) : \mathbf{c}_j \in \operatorname{Im}(\mathcal{E}_j), 1 \leq j \leq i\} \). For a message \( m \) we denote by \( \text{Ind}_m(x) \) the indicator function, where \( \text{Ind}_m(x) = 0 \) if \( m = x \), otherwise \( \text{Ind}_m(x) = 1 \). The encoding and decoding maps are defined as follows:

1. If \( k = 1 \) (encoder and decoder informed - EI:DI) then for each \( i \)
   
   \[
   \mathcal{E}_i : [M_i] \times \operatorname{Im}(\mathcal{E}_{i-1}) \rightarrow [q]^n, \\
   \mathcal{D}_i : \{(\mathcal{E}_i(m, \mathbf{c}), \mathbf{c}) : m \in [M_i], \mathbf{c} \in \operatorname{Im}(\mathcal{E}_{i-1})\} \rightarrow [M_i]
   \]

   such that for all \( (m, \mathbf{c}) \in [M_i] \times \operatorname{Im}(\mathcal{E}_{i-1}) \), \( \mathcal{E}_i(m, \mathbf{c}) \geq \mathbf{c} \), and

   \[
   \sum_{(m, \mathbf{c}) \in [M_i] \times \operatorname{Im}(\mathcal{E}_{i-1})} \Pr(m) \Pr(\mathbf{c}) \cdot \text{Ind}_m(\mathcal{D}_i(\mathcal{E}_i(m, \mathbf{c}), \mathbf{c})) \leq p_{e_i}.
   \]

2. If \( k = 2 \) (encoder informed, decoder uninformed - EI:DU) then for each \( i \)

   \[
   \mathcal{E}_i : [M_i] \times \operatorname{Im}(\mathcal{E}_{i-1}) \rightarrow [q]^n, \mathcal{D}_i : \operatorname{Im}(\mathcal{E}_i) \rightarrow [M_i]
   \]

   such that for all \( (m, \mathbf{c}) \in [M_i] \times \operatorname{Im}(\mathcal{E}_{i-1}) \), \( \mathcal{E}_i(m, \mathbf{c}) \geq \mathbf{c} \), and

   \[
   \sum_{(m, \mathbf{c}) \in [M_i] \times \operatorname{Im}(\mathcal{E}_{i-1})} \Pr(m) \Pr(\mathbf{c}) \cdot \text{Ind}_m(\mathcal{D}_i(\mathcal{E}_i(m, \mathbf{c}))) \leq p_{e_i}.
   \]
3. If \(k=3\) (encoder uninformed, decoder informed - EU:DI) then for each \(i\)
\[
\mathcal{E}_i : [M_i] \rightarrow [q]^n, \mathcal{D}_i : Im^*(\mathcal{E}_i) \times Im^*(\mathcal{E}_{i-1}) \rightarrow [M_i]
\]
such that
\[
\sum_{(m,c)\in[M_i]\times Im^*(\mathcal{E}_{i-1})} Pr(m)Pr(c) \cdot Ind_m (\mathcal{D}_i (\max\{c, \mathcal{E}_i(m)\}, c)) \leq p_{e_i}.
\]
4. If \(k=4\) (encoder and decoder uninformed - EU:DU) then for each \(i\)
\[
\mathcal{E}_i : [M_i] \rightarrow [q]^n, \mathcal{D}_i : Im^*(\mathcal{E}_i) \rightarrow [M_i]
\]
such that
\[
\sum_{(m,c)\in[M_i]\times Im^*(\mathcal{E}_{i-1})} Pr(m)Pr(c) \cdot Ind_m (\mathcal{D}_i (\max\{c, \mathcal{E}_i(m)\})) \leq p_{e_i}.
\]
If \(p_{e_i} = 0\) for all \(1 \leq i \leq t\), then the code is called a zero-error WOM code
and is denoted by \([n, t; M_1, \dotsc, M_t]^{\otimes, \epsilon}\).

If \(q = 2\) then the WOM code is called a binary WOM code, and \(q\) is usually omitted from the notation, otherwise it is called a non-binary WOM code. The rate of the \(i\)-th write is the ratio between the number of written bits and the total number of cells, i.e., \(R_i = \frac{\log M_i}{n}\), and the sum-rate of the WOM code is the sum of the individual rates on all writes, i.e., \(R_t^{sum} = \sum_{i=1}^t R_i\). A rate tuple \(\mathbf{r} = (R_1, \dotsc, R_t)\) is called \(\epsilon\)-error achievable in model \(k, k \in \{1, 2, 3, 4\}\), if for all \(\epsilon > 0\) there exists an \([n, t; M_1, \dotsc, M_t]^{\otimes, \epsilon}\) WOM code with error probability vector \(\mathbf{p}_e = (p_{e_1}, \dotsc, p_{e_t}) \leq (\epsilon, \dotsc, \epsilon)\), such that \(\frac{\log M_i}{n} \geq R_i - \epsilon\). The rate tuple \(\mathbf{r}\) will be called zero-error achievable if for all \(1 \leq i \leq t\), \(p_{e_i} = 0\). The capacity region of \(q\)-ary \(t\)-write WOM is the set of all the achievable rate tuples. Let \(C_{q,t}^{\otimes, \epsilon}, C_{q,t}^{\otimes, \epsilon, z}\) denote the capacity region, and \(R_{q,t}^{\otimes, \epsilon}, R_{q,t}^{\otimes, \epsilon, z}\) denote the maximum sum-rate of \(q\)-ary \(t\)-write WOM in model \(k\) for the \(\epsilon\)-error, the zero-error case, respectively.

In [26], Wolf et al. studied the binary case, and proved that the capacity regions of binary \(t\)-write WOM in the first two models (EI models) are equal, both for the \(\epsilon\)-error and the zero-error cases. They also showed that this region is the capacity region of model 3 for the \(\epsilon\)-error case. That is, the following holds for all \(t \geq 1\):
\[
C_{2,t}^{\otimes, \epsilon} = C_{2,t}^{\otimes, \epsilon, z} = C_{2,t}^{\otimes, \epsilon, z} = C_{2,t}^{\otimes, \epsilon} = C_{2,t}^{\otimes, \epsilon},
\]

\[52\]
where

\[ C_{2,t} = \left\{ (R_1, \ldots, R_t) \mid R_1 \leq h(p_1), \\
R_2 \leq h(p_2)(1 - p_1), \ldots, \\
R_{t-1} \leq h(p_{t-1}) \prod_{i=1}^{t-2}(1 - p_i), \\
R_t \leq \prod_{i=1}^{t-1}(1 - p_i), \\
\text{where } 0 \leq p_1, \ldots, p_{t-1} \leq 1/2 \right\}. \] (4.2)

It is also known that the maximum sum-rate in all these cases is \( \log(t + 1) \). Note that the zero-error capacity region of model 3, i.e. \( C_{3,z,t}^{c} \), is still unknown, even for the binary case.

In the non-binary case, only model 2 was studied by Fu and Vinck [8]. They calculated the zero-error capacity region \( C_{q,t}^{z} \), and found that the maximum sum-rate in this model, denoted by \( R_{q,t}^{z} \), is \( \log \left( \frac{q-1+t}{q-1} \right) \). In this paper we show that the capacity regions of models 1 and 2 for both the \( \epsilon \)-error and the zero-error cases are all the same, and thus \( \log \left( \frac{q-1+t}{q-1} \right) \) is the maximum sum-rate in these four cases. One can readily conclude that this is an upper bound on the sum-rate also for the other models. However, we prove that in models 3 and 4 this bound is not tight.

In Section 4.3 and Section 4.4 we show how codes in the \( Z \) channel, and the binary erasure channel (BEC) provide constructions of binary WOM codes in model 4 and model 3, respectively. The \( Z \) channel is a channel with binary inputs and outputs in which only a zero can change to a one and not vice versa. These are called asymmetric errors. The BEC is a channel with binary inputs and outputs in which a bit value can be erased.

For each channel (\( Z \) channel or BEC), we use the following two models: the coding theory model, which is used for zero-error WOM codes, and the information theory model which is applied for the \( \epsilon \)-error case. The coding theory model mimics an adversarial channel, where the adversary knows the entire codeword prior to transmission and can corrupt up to \( \lceil pn \rceil \) locations to each specific transmission, where \( p \in [0, 1] \). This is the worst-case noise model studied in coding theory. In the information theory model, the errors are generated in an independent identical distribution, with probability \( p \). We call \( p \) the channel error probability. For more details about these two models, see for example [7].

We say that a length-\( n \) code \( C \) with \( M \) codewords is an \( (n, M, \tau, p_e)_{BEC} \).
erasure-correcting code, if it can correct at most any \( \tau \) erasures with decoding error probability \( p_e \). The decoding error probability, \( p_e \), is defined as the probability of decoding incorrectly the transmitted message, where the erasure vector is chosen independently uniformly from the set of all the vectors of Hamming weight at most \( \tau \). Note that \( p_e \) can be defined as maximal or average decoding error probability where the maximum and average measures are computed over the set of the messages, since both of these problems have the same capacity in discrete memoryless channel (DMC) \[6, pp. 194, 204\]. If \( p_e = 0 \) then we omit this parameter, and we have an \((n, M, \tau)_{BEC} \) erasure-correcting code, which is capable of correcting any \( \tau \) erasures. An \((n, M, \tau, p_e)_Z \) asymmetric-error-correcting code, and \((n, M, \tau)_Z \) asymmetric-error-correcting code are defined similarly. Note, that \( p_e \) is usually defined to be the decoding error probability where each bit is corrupted independently uniformly with probability \( p \). For \( \tau = \lceil pn \rceil \), this definition is equivalent to our definition described above, with respect to the capacity results. We use our definition to simplify the construction and to clarify the connection between the two models of the used channel.

Let \( K \) be the \( Z \) channel or the BEC with channel error probability \( p \). It is said that \( R \) is an achievable rate in the information theory model of channel \( K \), if for each \( \epsilon > 0 \) there exists an \((n, M, \tau, p_e)_K \) error-correcting code consisting of \( M \geq 2^{n(R-\epsilon)} \) codewords of length \( n \), which is capable of correcting \( \tau = \lceil pn \rceil \) errors which occurred by \( K \) with decoding error probability \( p_e < \epsilon \). Similarly, \( R \) is an achievable rate in the coding theory model of a channel \( K \), if for each \( \epsilon > 0 \) there exists an \((n, M, \tau)_K \) error-correcting code with the same parameters except for \( p_e = 0 \).

### 4.3 Constructions for model 4 – The EU:DU Model

In this section we study constructions of WOM codes in model 4. We first present some known results \[26\] about the capacity region and the maximum sum-rate of binary WOM in the \( \epsilon \)-error case.

In subsection 4.3.1 we study the binary two-write case. We show a very simple construction for the zero-error case using only two cells which already achieves a significantly high sum-rate. Then, using codes for the \( Z \) channel we give a more general construction of binary WOM codes in this model. We use codes in the information theory model in the \( Z \) channel, in order to
construct WOM codes for the $\epsilon$-error case, which obtain each point in the achievable region presented in [26] and thus the maximum sum-rate $R_{2,2}^{(3),\epsilon}$ as well. By the same ideas, codes in the coding theory model are applied to construct WOM codes in the zero-error case. Based on the two-write constructions, we then show in subsection 4.3.2 a recursive construction of binary multiple-write WOM codes. The generalization for non-binary two-write WOM is presented in Appendix A.

The capacity region of a binary $t$-write WOM in model 4 for the $\epsilon$-error case was studied in [26]. The following region was shown to be achievable

$$\tilde{C}_t = \left\{ (R_1, \ldots, R_t) \mid R_1 \leq h(p_1), \right.$$  
$$R_2 \leq h(p_1p_2) - p_2 h(p_1), \ldots,$$  
$$R_{t-1} \leq h(\prod_{j=1}^{t-1} p_j) - p_{t-1} h(\prod_{j=1}^{t-2} p_j),$$  
$$R_t \leq h(\prod_{j=1}^{t} p_j) - p_t h(\prod_{j=1}^{t-1} p_j),$$

where $0 \leq p_1, \ldots, p_t \leq 1$, \hspace{1cm} (4.3)

that is, $\tilde{C}_t \subseteq C_{2,t}^{(3),\epsilon}$. However, it is not known if the converse holds as well, i.e., whether every achievable rate tuple belongs to this region. Here, $1 - p_i$ is the probability for programming a bit on the $i$-th write. For example, for $t = 2$ we get

$$\tilde{C}_2 = \left\{ (R_1, R_2) \mid R_1 \leq h(p_1), \right.$$  
$$R_2 \leq h(p_1p_2) - p_2 h(p_1),$$

where $0 \leq p_1, p_2 \leq 1$. \hspace{1cm} (4.4)

Even though, this capacity region is not necessarily a tight region, the authors in [26] could still give an achievable upper bound on the sum-rate in this model. Specifically, the maximum sum-rate, denoted by $P_t$ in [26], was shown to be given by

$$R_{2,t}^{(3),\epsilon} = \sup_{0 \leq p_1, \ldots, p_t \leq 1} \left\{ h(\prod_{j=1}^{t} p_j) + \sum_{i=2}^{t} ((1 - p_i) h(\prod_{j=1}^{i-1} p_j)) \right\}. \hspace{1cm} (4.5)$$

Furthermore it was shown that for all $t \geq 1$, $R_{2,t}^{(3),\epsilon} \leq \frac{\pi^2}{6 \ln 2} \approx 2.37$, and $\lim_{t \to \infty} R_{2,t}^{(3),\epsilon} = \frac{\pi^2}{6 \ln 2} \approx 2.37$. Table 4.1 presents the values of $R_{2,t}^{(3),\epsilon}$ for $2 \leq t \leq 5$, along with the probabilities vectors $p = (p_1, p_2, \ldots, p_t)$ that maximize this term.

Fig. 4.1 illustrates the capacity regions for binary two-write WOM in all the models. It demonstrates that $\tilde{C}_2$ (see Equation (4.4)), which is an achievable region for binary WOM in model 4 for the $\epsilon$-error case (inner
Table 4.1: Maximum Sum-Rates in Model 4

<table>
<thead>
<tr>
<th>$t$</th>
<th>$R_{2,t}^{(i),e}$</th>
<th>$p = (p_1, p_2, \ldots, p_t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.3881</td>
<td>(0.665, 0.4169)</td>
</tr>
<tr>
<td>3</td>
<td>1.600</td>
<td>(0.7475, 0.59, 0.395)</td>
</tr>
<tr>
<td>4</td>
<td>1.7356</td>
<td>(0.80, 0.69, 0.57, 0.39)</td>
</tr>
<tr>
<td>5</td>
<td>1.9695</td>
<td>(0.85, 0.75, 0.675, 0.60, 0.40)</td>
</tr>
</tbody>
</table>

line), is a proper subset of $C_{2,2}$ (see Equation (4.2)), the capacity region of binary WOM in all the other models, (outer curve).

Figure 4.1: A comparison between $C_{2,2}$ – the capacity regions for binary two-write WOM in models 1, 2, and 3, and $\tilde{C}_2$ – an achievable region for binary two-write WOM in model 4. The points are rates of specific constructions of zero-error WOM codes in model 4.
4.3.1 Binary Two-Write WOM Codes

Let us start with a simple two-write WOM code construction.

Example 6. In this example, we show a construction of a binary $[2, 2; 3, 2]^{\oplus, z}$ WOM code. On the first write a ternary symbol is written according to the encoding map

$$
0 \rightarrow (0, 0), \quad 1 \rightarrow (0, 1), \quad 2 \rightarrow (1, 0).
$$

Then, on the second write one more bit is written. If the bit value is zero the memory state does not change, that is the cells are programmed with $(0, 0)$. Otherwise, the cells are programmed to the $(1, 1)$ state. The decoding on each write is clear from the encoding. The sum-rate of this construction is

$$
\frac{\log 3 + 1}{2} \approx 1.29.\quad \text{By concatenation, one can construct for each positive integer } n, \ a \ [2n, 2; 3^n, 2^n]^{\oplus, z} \text{ WOM code with the same sum-rate.}
$$

According to Table 4.1, the maximum sum-rate of two-write WOM codes in the $\epsilon$-error case is $R_{2, 2}^{\oplus, \epsilon} = 1.388$, which is an upper bound for the zero-error case. Thus, this very simple example already achieves sum-rate which is fairly close to the upper bound. We show how to close on this gap for the $\epsilon$-error case by using codes for the $Z$ channel.

The construction of binary two-write WOM code we next propose is based on a reduction to the $Z$ channel. We use two models of the channel, the coding theory model, which is invoked for constructing zero-error WOM codes, and the information theory model which is applied for the $\epsilon$-error case, and achieves the capacity region $\tilde{C}_2$ and thus the maximum sum-rate $R_{2, 2}^{\oplus, \epsilon}$.

Recall that a length-$n$ code $C$ with $M$ codewords is an $(n, M, \tau, p_e)_Z$ asymmetric-error-correcting code if it can correct at most $\tau$ asymmetric errors with decoding error probability $p_e$.

Theorem 4. Let $C$ be an $(n, M, \tau, p_e)_Z$ asymmetric-error-correcting code. Then there exists an $[n, 2; M_1, M_2]^{\oplus, p_e}$ WOM code with error probability vector $p_e = (0, p_e)$, where $M_1 = \sum_{i=0}^{\tau} \binom{n}{i}$ and $M_2 = M$. If $p_e = 0$ then the constructed WOM code is a zero-error WOM code.

Proof. The proof will consist of describing the encoding and decoding maps of the WOM code. On the first write, $M_1$ messages are written by simply programming at most $\tau$ cells. We assume here, and later on, that there is a
mapping between the set $[M_1]$ and the set of all binary vectors of Hamming weight at most $\tau$. This mapping defines also the decoding of this write.

Let $E, D$ be the encoding, decoding map of the asymmetric-error-correcting code $C$, respectively. The encoder on the second write receives a message $m \in [M_2]$ to be encoded to the memory and programs the cells with the vector $E(m)$, given by applying the encoding map of $C$. We denote the cell-state vector after the first write by $c_1$. Since the vector $c_1$ is already programmed to the memory, the cell-state vector on the second write becomes $c_2 = \max\{c_1, E(m)\}$. The decoder on the second write applies the decoding map of $C$ on $c_2$. We have the following three observations:

1. $E(m)$ is a codeword in $C$,
2. $c_2 \geq E(m)$,
3. $d_H(E(m), c_2) \leq w_H(c_1) \leq \tau$.

That is, the cell-state vector $c_2$ is the outcome of at most $\tau$ asymmetric errors with respect to the codeword $E(m)$. Since the code $C$ is capable of correcting at most $\tau$ asymmetric errors with decoding error probability $p_e$, we have that $D(c_2) = m$ with probability at least $1 - p_e$, as required.

Note that Example 6 is a special case of Theorem 4, in which the code $C$ is a $(2, 2, 1)_Z$ asymmetric-error-correcting code. That is, the code is of length two, contains the two codewords $(0, 0)$ and $(1, 1)$ and it can correct a single asymmetric error.

The capacity of the $Z$ channel in the information theory model was well studied in the literature before [22, 24]. Let $\alpha$ be the probability for occurrence of 1 in the codewords, and $p$ be the crossover $0 \rightarrow 1$ probability, then the capacity was shown to be

$$h((1 - \alpha)(1 - p)) - (1 - \alpha)h(p).$$

Thus, the capacity of the $Z$ channel with the crossover $0 \rightarrow 1$ probability $p$ equals to

$$\text{cap}(Z) = \max_{0 \leq \alpha \leq 1} \{h((1 - \alpha)(1 - p)) - (1 - \alpha)h(p)\}$$

$$= \log \left( 1 + (1 - p)p^{p/(1 - p)} \right).$$
which is achieved for
\[ \alpha = 1 - \frac{1}{(1 - p)(1 + 2^{h(p)}/(1 - p))}, \]

In [26], Wolf et al. proved that it is possible to achieve all points in the region \( \tilde{C}_2 \) by random coding. The next theorem proves this fact by using the construction from Theorem 4, and using capacity achieving codes in the Z channel in the information model. Theorems 4 and 5 provide more explicit constructions and important insight about the connection to the Z channel. Note that all the capacity achievable WOM codes are for the \( \epsilon \)-error case.

**Theorem 5.** For any \( r \in \tilde{C}_2 \), \( r \) is \( \epsilon \)-error achievable by the construction from Theorem 4.

**Proof.** We prove that for any \((R_1, R_2) \in \tilde{C}_2\), and \( \epsilon > 0 \) there exists an \([n, 2; 2^{nR_1'}, 2^{nR_2'}]\) WOM code, constructed by Theorem 4, with error probability vector \( p_e = (0, p_e) \leq (\epsilon, \epsilon) \), and \( R_1' \geq R_1 - \epsilon, \ R_2' \geq R_2 - \epsilon \).

Let \( p_1, p_2 \in [0, 1] \) be such that \( R_1 \leq h(p_1) \) and \( R_2 \leq h(p_1 p_2) - p_2 h(p_1) \). Let \( p = 1 - p_1, \alpha = 1 - p_2, \) and \( \epsilon > 0 \). Based on the existence of capacity achieving codes for the Z channel, there exists an \((n, M, \tau, p_e)\) Z asymmetric-error-correcting code \( C \), such that \( \tau = [pn] \) if \( p \in [0, 0.5]^2 \) and otherwise \( \tau = [pn], p_e \leq \epsilon, \) and
\[ \log \frac{M}{n} \geq h((1 - \alpha)(1 - p)) - (1 - \alpha)h(p) - \epsilon. \]

According to the construction from Theorem 4, there exists an \([n, 2; M_1, M_2]\) WOM code with error probability vector \( p_e = (0, p_e) \), where \( M_1 = \sum_{i=0}^{\tau} \binom{n}{i} \) and \( M_2 = M \). Based on Lemma 4.8 in [17]
\[ \sum_{i=0}^{\tau} \binom{n}{i} \geq \frac{1}{n + 1} 2^{nh(\frac{\tau}{n})}. \]

\[^2\]If \( \tau = 0.5 \), we assume, without loss of generality, that \( n \) is even
Therefore, for $n$ large enough, the rates of this WOM code satisfy
\[
R'_1 = \frac{\log(\sum_{i=0}^{\tau} \binom{n}{i})}{n} \geq h\left(\frac{\tau}{n}\right) - \frac{\log(n+1)}{n} \\
\geq h(p) - \frac{\log(n+1)}{n} \geq R_1 - \epsilon,
\]
and
\[
R'_2 = \frac{\log M_2}{n} \geq h((1-\alpha)(1-p)) - (1-\alpha)h(p) - \epsilon \\
= h(p_1p_2) - p_2h(p_1) - \epsilon \geq R_2 - \epsilon.
\]

As an immediate result from Theorem 5, we conclude that there exists a family of binary two-write WOM codes in model 4 for the $\epsilon$-error case which achieve the maximum sum-rate $R_{4,\epsilon}^2$.

Although the $Z$ channel can provide us with capacity achieving codes for two writes for the $\epsilon$-error case, yet it is not easy to find specific WOM codes with high sum-rates, mostly because the problem of finding such codes in the $Z$ channel is still far to be solved.

By the same techniques, we can use codes for the coding theory model, which provide $(n,M,\tau)_Z$ asymmetric-error codes with decoding error probability zero, in order to construct zero-error WOM codes. However, the capacity in the coding theory model is unknown, and an upper-bound on the capacity is given by $\log \left(\frac{(\tau+1)2^n}{\sum_{j=0}^{\tau} \binom{n}{j}}\right)$ [1, 2]. Hence, if $\tau \approx pn$, and $n$ goes to infinity, then the maximum sum-rate is upper-bounded by $1-h(p)$. Thus, unfortunately, the maximum sum-rate of this construction for the zero-error case where $\tau \approx pn$, approaches 1 asymptotically.

We used some of the existing code constructions for asymmetric errors [12], and found WOM codes with the following parameters for the zero-error case, as described in Table 4.2. The rates of these WOM codes are marked also in the plot of Fig. 4.1.

Note that the best sum-rate that we could find remains 1.29 which is achieved by the WOM code from Example 6 using two cells. The problem of closing this gap with specific WOM code constructions still remains an
Table 4.2: An \([n, 2; M_1, M_2]^{\mathfrak{O},\tau}_Z\) WOM Code Constructed by an \((n, M, \tau)_Z\) Asymmetric-Error-Correcting Code

<table>
<thead>
<tr>
<th>((n, M, \tau)_Z)</th>
<th>(M_1)</th>
<th>(M_2)</th>
<th>(R^\text{sum}_2 - \text{sum-rate})</th>
</tr>
</thead>
<tbody>
<tr>
<td>((5, 2, 4)_Z)</td>
<td>31</td>
<td>2</td>
<td>1.1908</td>
</tr>
<tr>
<td>((6, 2, 4)_Z)</td>
<td>57</td>
<td>2</td>
<td>1.1388</td>
</tr>
<tr>
<td>((6, 4, 3)_Z)</td>
<td>42</td>
<td>4</td>
<td>1.23205</td>
</tr>
</tbody>
</table>

open problem. The extension of this construction to two-write non-binary WOM codes appears in Appendix A.

4.3.2 Binary \(t\)-Write WOM Codes

Given a binary two-write WOM code in model 4, we can construct binary \(t\)-write WOM codes for all \(t\). We accomplish this goal by a recursive construction which is proved in the next theorem.

**Theorem 6.** If \(C_t\) is an \([n, t; M_1, M_2, \ldots, M_t]^{\mathfrak{O},p_e}_Z\) WOM code, then there exists a \([2n, t + 1; 3^n, M_1, M_2, \ldots, M_t]^{\mathfrak{O},p'_e}_Z\) WOM code, \(C_{t+1}\), where \(p'_e = (0, p_{e1}, \ldots, p_{et})\)

**Proof.** The proof will consist of describing the encoding and decoding maps of the \((t + 1)\)-write WOM code \(C_{t+1}\). On the first write of \(C_{t+1}\), we invoke the first write of the two-write WOM code from Example 6. Thus, the \(2n\) cells are divided into pairs, where each pair can be programmed to one of the following vectors \((0, 0), (0, 1),\) or \((1, 0)\), resulting with \(3^n\) messages.

For the next writes, each pair of two cells represents one logical cell, where the pair \((1, 1)\) represents a logical value one and the other three pairs represent a logical value zero. Thus, on the \(i\)-th write, \(2 \leq i \leq t + 1,\) \(M_{i-1}\) messages can be encoded, decoded by using the encoder, decoder, of the \((i - 1)\)-st write of \(C_t\) over the \(n\) logical cells, respectively.

By applying the construction of Theorem 6 recursively, and using a two-write WOM code with sum-rate \(R^\text{sum}_2\) as the parameter of the recursion,
the sum-rate of the obtained $t$-write WOM code equals

$$
\mathcal{R}_t^{\text{sum}} = log_3 \cdot \sum_{i=1}^{t-2} (2^{-i}) + 2^{2-t} \cdot \mathcal{R}_2^{\text{sum}} \\
= log_3 + 2^{2-t} \cdot (\mathcal{R}_2^{\text{sum}} - log_3). 
$$

Thus, the value $\mathcal{R}_t^{\text{sum}}$ approaches $log_3$ as $t$ gets large enough. In Table 4.3, we present the results for the values of $\mathcal{R}_t^{\text{sum}}$ in case that $\mathcal{R}_2^{\text{sum}} = 1.29$ from Example 6, and for $\mathcal{R}_2^{\text{sum}} = \mathcal{R}_{2,t}^{\text{3,} \epsilon} = 1.388$, which is the maximum value for $\mathcal{R}_2^{\text{sum}}$ and can be asymptotically achieved by Theorem 5. These results are compared with the upper bound on the sum-rate, given by the value of $\mathcal{R}_{2,t}^{\text{3,} \epsilon}$.

Table 4.3: A Comparison between Values of $\mathcal{R}_t^{\text{sum}}$ and $\mathcal{R}_{2,t}^{\text{3,} \epsilon}$

<table>
<thead>
<tr>
<th>$t$</th>
<th>$\mathcal{R}_2^{\text{sum}} = 1.29$</th>
<th>$\mathcal{R}_2^{\text{sum}} = 1.38$</th>
<th>$\mathcal{R}_{2,t}^{\text{3,} \epsilon}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1.4387</td>
<td>1.4824</td>
<td>1.6000</td>
</tr>
<tr>
<td>4</td>
<td>1.5112</td>
<td>1.5337</td>
<td>1.7356</td>
</tr>
<tr>
<td>5</td>
<td>1.5480</td>
<td>1.5593</td>
<td>1.9695</td>
</tr>
</tbody>
</table>

The second and the third columns contain the values of $\mathcal{R}_t^{\text{sum}}$. The second column’s values are achieved by zero-error codes using the binary two-write WOM code with sum-rate 1.29 from Example 6. The third column’s values are attained for the $\epsilon$-error case using capacity achieving binary two-write WOM codes.

### 4.4 Constructions for model 3 – The EU:DI Model

In this section we describe some constructions of WOM codes in model 3. First, we recall some known results [26] regarding the capacity region and the maximum sum-rate of binary WOM in the $\epsilon$-error case in this model. Then, we study the binary two-write case. We show a very simple construction for the zero-error case. We also give a more general construction of binary WOM codes in this model, by using codes for the binary erasure channel (BEC).

As stated before in (4.1), the capacity regions of binary $t$-write WOM in model 1 and model 2 in both the $\epsilon$-error and the zero-error cases, and the
capacity region of binary $t$-write WOM in model 3 in the $\epsilon$-error case are all identical [26]. This capacity region, denoted by $C_{2,t}$, is presented in (4.2), and the maximum sum-rate in these cases is $\log(t + 1)$.

We start with a very simple construction for a zero-error WOM code using only three cells which is derived from a similar construction by Rivest and Shamir for model 2 [19].

**Example 7.** In this example, we show a construction of a binary $[3, 2; 4, 4]^{(2,2)}$ WOM code, achieving sum-rate $4/3$. Table 4.4 describes the encoders’ maps. The encoding map of the first write defines also the decoding of this write.

<table>
<thead>
<tr>
<th>Message</th>
<th>First write</th>
<th>Second write</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>110</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>101</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>011</td>
</tr>
</tbody>
</table>

We denote the cell-state vector after the first, second write by $c_1, c_2$, respectively. Let $u$ be the vector which was encoded in the second write. The decoder input is $c_1$ and $c_2 = \max\{c_1, u\}$. It is possible to verify the following equation.

$$u = \begin{cases} 
(0, 0, 0) & \text{if } w_H(c_2) \leq 1 \\
(c_2) & \text{if } w_H(c_2) = 2 \\
(c_2 + c_1) & \text{if } w_H(c_2) = 3
\end{cases}$$

Thus, the decoder can reveal $u$, and the message of the second write is decoded by the second column in Table 4.4.

We next give a general construction for binary two-write WOM in this model. This construction is based on a reduction to the BEC. We construct $\epsilon$-error WOM codes by codes in the information theory model of the BEC, which achieve the capacity region $C_{2,2}^{(2,\epsilon)}$, and thus the maximum sum-rate $\log 3$ as well, while for the zero-error case, we invoke codes in the coding theory model.

The construction is specified directly by codes correcting erasures. Recall that a length-$n$ code $\mathcal{C}$ with $M$ codewords is an $(n, M, \tau, p_e)_{BE}\text{C}$ erasure-correcting code if it can correct at most $\tau$ erasures with decoding error.
probability $p_e$. In particular, codes with minimum Hamming distance $\tau + 1$
can correct $\tau$ erasures with $p_e = 0$.

**Theorem 7.** Let $C$ be an $(n, M, \tau, p_e)$ BEC erasure-correcting code. Then
there exists an $[n, 2; M_1, M_2]^{\Theta, p_e}$ WOM code with decoding error probability
$p_e = (0, p_e)$, where $M_1 = \Sigma_{i=0}^{\tau}(n)$ and $M_2 = M$. If $p_e = 0$ then the
constructed code is zero-error WOM code.

**Proof.** The proof will consist of describing the encoding and decoding maps
of the WOM code. On the first write $M_1$ messages are written by simply
programming at most $\tau$ cells. Let $E, D$ be the encoding, decoding maps
of the erasure-correcting code $C$, respectively. The encoder on the second write
receives a message $m \in [M_2]$ to be encoded to the memory and programs
the cells with the vector $E(m)$. We denote the cell-state vector after the
first, second write by $c_1, c_2$, respectively. Thus, $c_2 = \max\{c_1, E(m)\}$. Let
$S = \{i : c_{1,i} = 1\}$. We have the following four observations:

1. $E(m)$ is a codeword in $C$,
2. $c_2 \geq E(m)$,
3. $d_H(E(m), c_2) \leq w_H(c_1) = |S| \leq \tau$,
4. the set $S$ is known to the decoder.

That is, the cell-state vector $c_2$ is the outcome of at most $\tau$ erasures in the
codeword $E(m)$ where the set of the erasures’ locations is $S$. Since the code
$C$ is capable of correcting at most $\tau$ erasures, we have that $D(c_2) = m$ with
probability at least $1 - p_e$, as required.

Note that Example 7 is a special case of Theorem 7, in which the code $C$
is a $(3, 4, 1)_{BEC}$ erasure-correcting code. That is, the code is of length three,
contains the following four codewords $(0, 0, 0), (1, 1, 0), (1, 0, 1), (0, 1, 1)$, and
can correct a single erasure.

The next theorem proves that by capacity achieving codes in the BEC
in the information model, it is possible to achieve all points in the region
$C_{2,2} = C_{2,2}^{\epsilon}$ through the construction from Theorem 7 for the $\epsilon$-error case.
Recall that the capacity of the BEC in the information theory model is
known to be $1 - p$, where $p$ is the erasure probability [6, pp. 189].

64
Theorem 8. For any \( r \in C_{2,2} \), \( r \) is achievable by the construction form Theorem 7.

Proof. We prove that for any \((R_1, R_2) \in C_{2,2}\), and \( \epsilon > 0 \) there exists an \([n, 2; 2^n R_1', 2^n R_2']^{(3)} p_e\) WOM code, constructed by Theorem 7, with error probability vector \( p_e = (0, p_{e2}) \leq (\epsilon, \epsilon) \), and \( R_1' \geq R_1 - \epsilon, R_2' \geq R_2 - \epsilon \).

Let \( p \in [0, 0.5] \) be such that \( R_1 \leq h(p) \) and \( R_2 \leq 1 - p \). By capacity achieving codes in the BEC in the information theory model, for \( p \) and \( \epsilon > 0 \), there exists an \((n, M, \tau, p_e)_{BEC}\) erasure-correcting code \( C \), such that \( \tau = \lceil pn \rceil \), \( p_e < \epsilon \), and

\[
\frac{\log M}{n} \geq 1 - p - \epsilon.
\]

According to the construction from Theorem 7, there exists an \([n, 2; M_1, M_2]^{(3)} p_e\) WOM code with error probability vector \( p_e = (0, p_e) \leq (\epsilon, \epsilon) \), where \( M_1 = \sum_{i=0}^{\tau} \binom{n}{i} \) and \( M_2 = M \). Using Lemma 4.8 in [17], as in Theorem 5, the rates of this WOM code satisfy

\[
R_1' = \frac{\log (\sum_{i=0}^{\tau} \binom{n}{i})}{n} \geq R_1 - \epsilon,
\]

and \( R_2' = \frac{\log M_2}{n} \geq 1 - p - \epsilon \) for \( n \) large enough. \( \square \)

By the same techniques, we can use codes for the coding theory model, which provide \((n, M, \tau)_{BEC}\) erasure-correcting codes with decoding error probability zero, in order to construct zero-error WOM codes in model 3. However, the capacity in the coding theory model is unknown. The best known achievable scheme for the coding model corresponds to codes suggested by Gilbert and Varshamov [9, 23] which achieve a rate of \( 1 - h(p) \) where \( pn \) is the maximum number of erased indices. Thus, unfortunately, this lower-bound doesn’t provide sum-rate greater than 1. On the other hand, the best known upper bound is the MRRW (McEliece-Rodemich-Rumsey-Welch) bound obtained as the solution of an LP [16]. This bound, called the second MRRW bound, strengthens the following three bounds: the Hamming (sphere-packing) \( 1 - h(\frac{p}{2}) \), the Elias-Bassalygo and the first MRRW bound \( h(\frac{1}{2} - \sqrt{p(1-p)}) \) for \( 0 < p < \frac{1}{2} \). This upper-bound provides us a limit on the power of this construction for the zero-error case. In particular, according to this bound the maximum sum-rate in this construction is upper bounded by 1.18. Note that by the WOM code in Example 7,
we have already managed to achieve sum-rate 1.33. The extension of this construction to non-binary WOM codes appears in Appendix B.

### 4.5 Capacity Region of Models 1 and 2 – EI Models

In this section we follow the derivations from [8] and [26] to prove equality between the capacity regions of $q$-ary $t$-write WOM in models 1 and 2 (EI models), and additionally to show equality between the $\epsilon$-error and the zero-error capacity regions in these models. That is, we extend the result for models 1 and 2 stated in (4.1), for the non-binary case and show that for all $q$ and $t$,

$$
C_{q,t}^{z} = C_{q,t}^{e} = C_{q,t}^{\epsilon} = C_{q,t}^{\epsilon, z}.
$$

(4.6)

Clearly, $C_{q,t}^{z} \subseteq C_{q,t}^{e} \subseteq C_{q,t}^{\epsilon}$, and $C_{q,t}^{z} \subseteq C_{q,t}^{\epsilon, z} \subseteq C_{q,t}^{\epsilon}$. Thus, in order to establish the equalities in (4.6), it is enough to prove that $C_{q,t}^{\epsilon, z} \subseteq C_{q,t}^{\epsilon}$.

Let us introduce several more notations to be used in this section. For $1 \leq i \leq t$, and $0 \leq j_1 \leq j_2 \leq q - 1$, let $p_{i,j_1 \rightarrow j_2}$ be the conditional probability of writing the symbol $j_2$ on the $i$-th write given that the cell is in state $j_1$, and $p_{i,j}$ is the probability vector $p_{i,j} = (p_{i,j \rightarrow j_1}, p_{i,j \rightarrow (j+1)}, \ldots, p_{i,j \rightarrow (q-1)})$. If $j_1 > j_2$, then $p_{i,j_1 \rightarrow j_2} = 0$, and $p_{i,(q-1) \rightarrow (q-1)} = 1$, and $p_{i,q-1} = (p_{i,(q-1) \rightarrow (q-1)})$ is a probability vector of length 1. We let $Q_{i,j}$ denote the probability that a cell’s state after $i$ writes is $j$, and $Q_i = (Q_{i,0}, Q_{i,1}, \ldots, Q_{i,q-1})$. Note that $Q_i$ is a function of $p_{i,j_1}, p_{i,j_2}, \ldots, p_{i,j}, j \in [q]$, and can be computed recursively for all $i \geq 1$ and $j \in [q]$ as follows:

$$
Q_{0,j} = \begin{cases} 
1, & \text{if } j = 0, \\
0, & \text{else}
\end{cases}
$$

(4.7)

$$
Q_{i,j} = \sum_{k=0}^{j} Q_{i-1,k} p_{i,k \rightarrow j}.
$$

We define the following region $C_{q,t}$ in order to show that it is the capacity region of models 1 and 2. This region can be readily verified to be equivalent to the one presented by Fu and Vinck [8] for the zero-error case, though its
representation was more general.

\[ C_{q,t} = \{(R_1, R_2, \ldots, R_t) | \forall 1 \leq i \leq t : \]
\[ R_i \leq \sum_{j=0}^{q-2} Q_{i-1,j} H(p_{i,j}), \]
\[ \forall 1 \leq i \leq t, j \in [q] : \]
\[ p_{i,j} \text{ is a probability vector,} \]
\[ Q_{i,j} \text{ is defined in (4.7)} \}. \tag{4.8} \]

Note that for \( i = t \), the upper bound \( \log(q - j) \) of \( H(p_{t,j}) \) can be achieved. So, one can also write \( R_t \leq \sum_{j=0}^{q-2} Q_t-1,j \log(q - j) \).

For example, the capacity region of binary \( t \)-write WOM in models 1 and 2 is presented in (4.2), where for \( 1 \leq i \leq t \), \( p_i \) in (4.2) is the probability to write symbol 1 on the \( i \)-th write, which was denoted in \( C_{q,t} \) by \( p_{i,0\rightarrow 1} \). Therefore \( 1 - p_i \) equals \( p_{i,0\rightarrow 0} \), and \( Q_{i,0} = \prod_{j=1}^{i}(1 - p_j) \).

In addition, the capacity region of 3-ary two-write WOM in models 1 and 2 is

\[ C_{3,2} = \{(R_1, R_2) | R_1 \leq H(p_{1,0\rightarrow 0}, p_{1,0\rightarrow 1}, p_{1,0\rightarrow 2}), \]
\[ R_2 \leq p_{1,0\rightarrow 0} \cdot \log 3 + p_{1,0\rightarrow 1} \cdot \log 2, \]
\[ \text{where } p_{1,0} = (p_{1,0\rightarrow 0}, p_{1,0\rightarrow 1}, p_{1,0\rightarrow 2}) \text{ is a probability vector} \}. \]

Since Fu and Vinck [8] have already showed that \( C_{q,t} = C_{q,t}^{(2,2)} \), to complete the proof of (4.6) we are only required to prove that \( C_{q,t}^{(2,2)} \subseteq C_{q,t} \), i.e. the converse part. In this proof and in the rest of the paper, we denote by \( X_i \) the vector written on the \( i \)-th write to the memory, and by \( Y_i \) the cell-state vector after the \( i \)-th write. We let \( Y_0 \) be the zero vector.

**Theorem 9** (Converse part). If there exists an \( [n, t; M_1, \ldots, M_t]_q^{(2,2)}p_e \) WOM code, where \( p_e = (p_{e1}, \ldots, p_{et}) \), then

\[ \left( \frac{\log M_1}{n} - \epsilon_1, \frac{\log M_2}{n} - \epsilon_2, \ldots, \frac{\log M_t}{n} - \epsilon_t \right) \in C_{q,t}, \]

where \( \epsilon_i = H(p_{ei}) + p_{ei} \log(M_i)/n \).

**Proof.** Let \( S_1, \ldots, S_t \) be independent random variables, where \( S_i \) is uni-
formally distributed over the messages set \([M_i]\). The data processing yields the following Markov chain:

\[ S_i|Y_{i-1} \rightarrow X_i|Y_{i-1} \rightarrow Y_i|Y_{i-1} \rightarrow \hat{S}_i|Y_{i-1} \]

and therefore, \(I(X_i; Y_i|Y_{i-1}) \geq I(S_i; \hat{S}_i|Y_{i-1})\). Additionally,

\[
I(S_i; \hat{S}_i|Y_{i-1}) = H(S_i|Y_{i-1}) - H(S_i|\hat{S}_i, Y_{i-1}) \\
\geq H(S_i) - H(S_i|\hat{S}_i) \\
\geq \log(M_i) - H(p_{e_i}) - p_{e_i} \log(M_i).
\]

The first inequality follows from the independence of \(Y_{i-1}\) and \(S_i\) which implies that \(H(S_i|Y_{i-1}) = H(S_i)\), and from the fact that conditioning does not increase the entropy. The second inequality follows from Fano’s inequality [6, pp. 38] \(H(S_i|\hat{S}_i) \leq H(p_{e_i}) + p_{e_i} \log(M_i)\). Let \(L\) be an index random variable, which is uniformly distributed over the index set \([n]\). Since \(L\) is independent of all other random variables we get

\[
\frac{1}{n} I(X_i; Y_i|Y_{i-1}) \leq \frac{1}{n} H(Y_i|Y_{i-1}) \\
\leq \frac{1}{n} \sum_{k=0}^{n-1} H(Y_{i,k}|Y_{i-1,k}) \\
\leq H(Y_{i,L}|Y_{i-1,L}, L) \\
\leq \sum_{j=0}^{q-1} Pr(Y_{i-1,L} = j)H(Y_{i,L}|Y_{i-1,L} = j) \\
\leq \sum_{j=0}^{q-2} Pr(Y_{i-1,L} = j)H(Y_{i,L}|Y_{i-1,L} = j),
\]

where steps (a) and (c) follow from the fact that entropy of a vector is not greater than the sum of the entropies of its components, and conditioning does not increase the entropy. Step (b) follows from the fact that

\[
H(Y_{i,L}|Y_{i-1,L}, L) = \sum_{k=0}^{n-1} Pr(L = k)H(Y_{i,k}|Y_{i-1,L}, L = k) \\
= \frac{1}{n} \sum_{k=0}^{n-1} H(Y_{i,k}|Y_{i-1,k}),
\]

and step (d) follows from \(H(Y_{i,L}|Y_{i-1,L} = q - 1) = 0\).
Now, we set \( p_{i,j_1 \rightarrow j_2} = Pr(Y_{i,L} = j_2 | Y_{i-1,L} = j_1) \) and thus conclude that

\[
Q_{i,j} \triangleq Pr(Y_{i,L} = j) = \sum_{k=0}^{\tilde{j}} Pr(Y_{i,L} = j, Y_{i-1,L} = k) Pr(Y_{i-1,L} = k) = \sum_{k=0}^{\tilde{j}} p_{i,k \rightarrow j} Q_{i-1,k},
\]

and

\[
\frac{\log(M_i)}{n} - \epsilon_i \leq \frac{1}{n} I(X_i;Y_i|Y_{i-1}) \leq \frac{1}{n} \sum_{j=0}^{q-2} Pr(Y_{i-1,L} = j) H(Y_{i,L}|Y_{i-1,L} = j) = \frac{1}{n} \sum_{j=0}^{q-2} Q_{i-1,j} H(p_{i,j \rightarrow j_1}, \ldots, p_{i,j \rightarrow (q-1)}),
\]

where \( \epsilon_i = \frac{H(p_{ei}) + p_{ei} \log(M_i)}{n} \), and the converse part is implied.

Later on, we denote the capacity region \( C_{q,t} \) by \( C_{\triangleq q,t} \), and the maximum sum-rate of this region by \( R_{\triangleq q,t} \). Recall that \( R_{\triangleq q,t} = \log \left( \frac{q-1+t}{q-1} \right) \) [8].

### 4.6 Capacity Region of model 3 – EU:DI Model

In this section we study the \( \epsilon \)-error capacity of model 3. The binary case of this model was proved in [26], and was shown to be the same as the capacity \( C_{2,t} \). We observe in our analysis that this property no longer holds for the non-binary case. Note that the capacity region in the zero-error case in this model is still unknown even for the binary case.

The programming probabilities in this case are not defined as in models 1 and 2, simply because the encoder can no longer read the memory state prior to encoding on each write. We let \( p_{i,j} \) be the probability of writing the symbol \( j \) on the \( i \)-th write. Denote the probability vector \( p_{i,j} \) for \( 1 \leq i \leq t \) and \( j \in [q-1] \) to be

\[
p_{i,j} = \left( \Sigma_{k=0}^{j} p_{i,k}, p_{i,j+1}, \ldots, p_{i,q-1} \right).
\]

We let \( Q_{i,j} \) be the probability of a cell’s state to be in level \( j \) after the \( i \)-th write, and \( Q_i = (Q_{i,0}, Q_{i,1}, \ldots, Q_{i,q-1}) \). Note that \( Q_i \) is a function of \( p_{i,j} \) which is calculated recursively for all \( i \geq 1 \) and \( j \in [q] \) as follows:

\[
Q_{0,j} = \begin{cases} 1, & \text{if } j = 0, \\ 0, & \text{else} \end{cases} \quad (4.10)
\]

\[
Q_{i,j} = Q_{i-1,j}(\sum_{k=0}^{j-1} p_{i,k}) + p_{i,j}(\sum_{k=0}^{j} Q_{i-1,k}).
\]

69
We define the region $\hat{C}_{q,t}$, and then prove that $\hat{C}_{q,t} = C_{q,t}^{(3)}$, 

$$
\hat{C}_{q,t} = \left\{ (R_1, R_2, \ldots, R_t) \mid \forall i \leq t : 
R_i \leq \sum_{j=0}^{q-2} Q_{i-1,j} H(p_{i,j}), \\
\forall 1 \leq i \leq t, j \in [q] : 
p_{i,j} \text{ is a probability vector as defined in (4.9), and} 
Q_{i,j} \text{ is defined in (4.10)} \right\}.
$$

For example, the capacity region of 3-ary 2-write WOM is

$$
\widehat{C}_{3,2} = \left\{ (R_1, R_2) \mid R_1 \leq H(p_{1,0}, p_{1,1}, p_{1,2}), \\
R_2 \leq p_{1,0} H(p_{2,0}, p_{2,1}, p_{2,2}) \\
+ p_{1,1} H(p_{2,0} + p_{2,1}, p_{2,2}) \\
\text{where } 0 \leq p_{i,j}, \sum_{j=0}^{2} p_{i,j} = 1 \right\}.
$$

**Theorem 10.** $\hat{C}_{q,t}$ is the capacity region of $\epsilon$-error $q$-ary $t$-write WOM in model 3, i.e., $\hat{C}_{q,t} = C_{q,t}^{(3),\epsilon}$.

The proof of Theorem 10 consists of two parts, which are presented in the following two sub-sections.

### 4.6.1 Proof of Direct Part – Theorem 10

For the direct part, we prove that for each $\epsilon > 0$, and $(R_1, R_2, \ldots, R_t) \in \hat{C}_{q,t}$ there exists an $[n, t, M_1, \ldots, M_t]^{3} p_e$ WOM code, where $p_e = (p_{e_1}, \ldots, p_{e_t}) \leq (\epsilon, \ldots, \epsilon)$, and $\frac{\log M_t}{n} \geq R_i - \epsilon$ for all $1 \leq i \leq t$. We use the well-known random channel-coding theorem [6, pp. 200]. We describe the encoding and decoding on each write.

On the $i$-th write, the encoder writes the symbol $j$ with probability $p_{i,j}$, i.e., for $k \in [n]$:

$$
Pr(X_i = j) = p_{i,j}.
$$

It is readily verified that that for $k \in [n]$: $Pr(Y_i = j) = Q_{i,j}$, where $Q_{i,j}$ is defined in (4.10). The decoder on the $i$-th round knows both $Y_{i-1}, Y_i$. Thus, the $i$-th write presents a DMC with input $X_i$, output $Z_i = (Y_{i-1}, Y_i)$, and transition probabilities, $Pr(Y_i | X_i, k)$, determined by the probability $Q_{i-1,j}$. Let $x_i = X_{i,k}$, $y_{i-1} = Y_{i-1,k}$, $y_i = Y_{i,k}$ for some index $k$. By the random coding theorem, for $n$ large enough, we can have highly reliable transmission with $p_{e_i} \leq \epsilon$ and
provided rate $\mathcal{R}_i = I(x_i; z_i) - \epsilon$. That is, the following region is achievable

$$\left\{ (\mathcal{R}_1, \ldots, \mathcal{R}_t) | \mathcal{R}_i \leq I(x_i; z_i = (y_{i-1}, y_i)) \right\},$$

where

$$I(x_i; z_i) = H(z_i = (y_{i-1}, y_i)) - H(z_i|x_i) = H(y_{i-1}|y_{i-1} - H(z_i = (y_{i-1}, y_i)|x_i)$$

$$(a) = H(Q_{i-1}) + H(y_i|y_{i-1}) - H(Q_{i-1})$$

$$= H(y_i|y_{i-1})$$

$$(b) = \sum_{j=0}^{q-1} Pr(y_{i-1} = j) H(y_i|y_{i-1} = j)$$

$$(c) = \sum_{j=0}^{q-2} Q_{i-1,j} H(p_{i,j}).$$

Step (a) follows from $H((y_{i-1}, y_i)|x_i) = H(y_{i-1}|x_i)$ since $y_i$ is a function of $x_i, y_{i-1}$, and $H(y_{i-1}|x_i) = H(y_{i-1})$ because $y_{i-1}$ and $x_i$ are independent. Also, $H(y_i|y_{i-1} = q - 1) = 0$ implies (b), and (c) is provided by

$$Pr(y_i = \ell|y_{i-1} = j) = \begin{cases} \sum_{k=0}^{j} p_{i,k}, & \text{if } \ell = j \\ p_{i,\ell}, & \text{if } \ell > j \\ 0, & \text{else} \end{cases}$$

Thus, this region is exactly $\hat{C}_{q,t}$, and we have $\hat{C}_{q,t} \subseteq C_{q,t}^{\oplus,\epsilon}.$

4.6.2 Proof of Converse Part – Theorem 10

In this section we prove the converse part of the capacity region. We use the same techniques as in model 1. Thus, many details, which are identical to the proof of Theorem 9, are omitted.

(Converse part of Theorem 10). If there exists an $(n, t; M_1, \ldots, M_t)_q^{\oplus, P_e}$ WOM code, where $p_e = (p_{e1}, \ldots, p_{et})$, then

$$\left( \frac{\log M_1}{n} - \epsilon_1, \frac{\log M_2}{n} - \epsilon_2, \ldots, \frac{\log M_t}{n} - \epsilon_t \right) \in \hat{C}_{q,t},$$

where $\epsilon_i = H(p_{ei}) + p_{ei} \log(M_i)$.

Proof. Let $S_1, \ldots, S_t$ be independent random variables as defined in the
proof of Theorem 9. We can follow the same steps as in this proof to get

\[
I(X_i; Y_i | Y_{i-1}) \geq I(S_i; \hat{S}_i | Y_{i-1}) \geq \log(M_i) - H(p_{e_i}) - p_{e_i} \log(M_i)
\]

and

\[
\frac{1}{n} I(X_i; Y_i | Y_{i-1}) \leq \sum_{j=0}^{q-2} Pr(Y_{i-1,L} = j) H(Y_{i,L} | Y_{i-1,L} = j)
\]

where \( L \) is an index random variable, which is uniformly distributed over the index set \([n]\), and is independent of all other random variables. The random variables \( X_{i,L} \) and \( Y_{i-1,L} \) are independent in model 3, and \( Y_{i,L} = \max\{X_{i,L}, Y_{i-1,L}\} \). Therefore

\[
Pr(Y_{i,L} = \ell | Y_{i-1,L} = j) = \begin{cases} 
\sum_{k=0}^{j} Pr(X_{i,L} = k), & \text{if } \ell = j \\
Pr(X_{i,L} = \ell), & \text{if } \ell > j \\
0, & \text{else}
\end{cases}
\]

Now, by choosing \( p_{i,j} \triangleq Pr(X_{i,L} = j) \), we can conclude that

\[
\frac{\log(M_i)}{n} - \epsilon_i \leq \frac{1}{n} I(X_i; Y_i | Y_{i-1}) \leq \sum_{j=0}^{q-2} Pr(Y_{i-1,L} = j) H(Y_{i,L} | Y_{i-1,L} = j) = \sum_{j=0}^{q-2} Q_{i-1,j} H(p_{i,j})
\]

where \( p_{i,j} = (\sum_{k=0}^{j} p_{i,k}, p_{i,k+1}, \ldots, p_{i,q-1}) \) defined in (4.9), and \( Q_{i,j} \) defined in (4.10), and \( \epsilon_i = \frac{H(p_{e_i}) + p_{e_i} \log(M_i)}{n} \). Thus, the converse part is implied. \( \square \)

4.7 Comparison between the Capacities of the Models

The main goal of this section is to compare between the capacity regions of models 1 and 2 (EI models), and model 3 (EU:DI model). We prove that for all \( q > 2 \) and \( t \geq 2 \), the \( \epsilon \)-error capacity region of \( q \)-ary \( t \)-write WOM in model 3 is a subset of the interior of the capacity region of \( q \)-ary \( t \)-write WOM in models 1 and 2. This implies that \( C_{q,t}^{1,\epsilon} \subseteq C_{q,t}^{2,\epsilon} \) and \( R_{q,t}^{1,\epsilon} < R_{q,t}^{2,\epsilon} \), however, we will see that the difference between \( R_{q,t}^{1,\epsilon} \) and \( R_{q,t}^{2,\epsilon} \) is upper bounded by a constant which depends only on \( q \) but not on \( t \). We also examine the probabilities which attain the maximum sum-rate. These
results are demonstrated in Fig. 4.2 and Fig. 4.3. In addition, finding the capacity region of model 4 is an open question, even for the binary case. However, in [26] it was proved that the maximum sum-rate of binary t-write WOM in model-4 is finite where \( t \to \infty \), while in the other models it goes to infinity. We generalize this result for all \( q > 2 \).

Even though the next lemma is a straightforward property we use its proof in deriving the other results in this section.

**Lemma 10.** If \( \mathbf{r} \in \hat{C}_{q,t} \) then \( \mathbf{r} \in C_{q,t} \), i.e., \( \hat{C}_{q,t} \subseteq C_{q,t} \).

**Proof.** The claim is derived simply by

\[
\hat{C}_{q,t} = C_{q,t}^\hat{\psi} \subseteq C_{q,t}^\psi = C_{q,t}.
\]

However we present another proof, which examines the probabilities for which \( \mathbf{r} \) is attained in \( C_{q,t} \).

Let \( \mathbf{r} = (R_1, R_2, \ldots, R_t) \in \hat{C}_{q,t} \) where \( \mathbf{r} \) is achieved by probabilities \( p_{i,j} \), and define

\[
\hat{p}_{i,j_1 \to j_2} = \begin{cases} 
\sum_{k=0}^{j_2} p_{i,k}, & \text{if } j_2 = j_1 \\
p_{i,j_2}, & \text{if } j_2 > j_1 \\
0, & \text{else}
\end{cases}
\]

Thus, one can readily verify that \( \mathbf{r} \) is achieved in \( C_{q,t} \) for probabilities \( \hat{p}_{i,j_1 \to j_2} \). It is possible to show that the probabilities to have symbol \( j \) in a cell after the \( i \)-th write, denoted by \( Q_{i,j} \), are equal in both regions, \( C_{q,t} \) and \( \hat{C}_{q,t} \).

**Lemma 11.** If \( \mathbf{r} = (R_1, R_2, \ldots, R_t) \in \hat{C}_{q,t} \) and \( \mathbf{0} \neq (R_1, R_2, \ldots, R_{t-1}) \), then there exists \( \mathbf{r}' = (R'_1, R'_2, \ldots, R'_t) \in \hat{C}_{q,t} \) such that \( \mathbf{r} \leq \mathbf{r}' \) and \( \mathbf{r}' \) is attained by \( p'_{i,j} \), \( 1 \leq i \leq t, j \in [q] \), with \( Q'_{i-1,0}, Q'_{i-1,1} > 0 \) where \( Q'_{i-1,0}, Q'_{i-1,1} \) are defined in (4.10).

**Proof.** Denote by \( p_{i,j} \), \( 1 \leq i \leq t, j \in [q] \), the parameters for which \( \mathbf{r} \) is attained \( \hat{C}_{q,t} \). We prove that the exist a set of parameters \( p'_{i,j} \), \( 1 \leq i \leq t, j \in [q] \), which correspond to a rate vector \( \mathbf{r}' \) such that \( \mathbf{r}' \geq \mathbf{r} \). Let \( j \) be the smallest number such that \( R_j > 0 \). Note that \( j < i \) since \( \mathbf{0} \neq (R_1, R_2, \ldots, R_{i-1}) \). We can choose \( p'_{1,0} = p'_{2,0} = \ldots = p'_{j-1,0} = 1 \) provides \( R'_1 = R'_2 = \ldots = R'_{j-1} = 0 \). Note the entropy of a vector is determined by the values of its components disregarding their order. Thus, on the \( j \)-th
write, we can choose the probability vector \((p_{j,0}, p_{j,1}, \ldots, p_{j,q-1})\) in non-increasing order which provides \(\mathcal{R}'_{j} \geq \mathcal{R}_{j}\), and \(p'_{j,0} \geq p'_{j,1} > 0\) since \(\mathcal{R}_{j} > 0\). For the next writes, let \(\ell > j\), if \(p_{\ell,0} = 0\), then let \(k\) be the smallest number in \([q]\) such that \(p_{\ell,k} > 0\). If \(k = 1\), then \(p'_{\ell,0} = p'_{\ell,1} = \frac{p_{\ell,1}}{2}\) else, \(k > 1\), then \(p'_{\ell,0} = p'_{\ell,1} = \frac{p_{\ell,k}}{\ell}\). It can be easily verified the \(\mathcal{R}'_{t} \geq \mathcal{R}_{t}\) for all \(\ell > j\), and \(Q'_{t,0}, Q'_{t,1} > 0\) for all \(i' \geq j\), and in particular \(Q'_{t-1,0}, Q'_{t-1,1} > 0\) as required. □

In the next three lemmas, we will show that \(\mathcal{C}_{q,t}\) is a subset of the interior of \(C_{q,t}\).

**Lemma 12.** If \(\mathbf{r} = (\mathcal{R}_{1}, \mathcal{R}_{2}, \ldots, \mathcal{R}_{t}) \in \mathcal{C}_{q,t}\) and \(\mathbf{0} \neq (\mathcal{R}_{1}, \mathcal{R}_{2}, \ldots, \mathcal{R}_{t-1})\), then there exists an \(\epsilon > 0\) such that \(\mathbf{r}_{t-\epsilon} \in \mathcal{C}_{q,t}\).

**Proof.** If there exists an \(\epsilon > 0\) such that \(\mathbf{r}_{t-\epsilon} \in \mathcal{C}_{q,t}\), then by Lemma 10, the claim is obviously true. Otherwise, by Lemma 11, there exists \(\mathbf{r}' = (\mathcal{R}'_{1}, \mathcal{R}'_{2}, \ldots, \mathcal{R}'_{t}) \in \mathcal{C}_{q,t}\) such that \(\mathbf{r} \leq \mathbf{r}'\) and \(\mathbf{r}'\) is attained by \(p_{i,j} > q_{i,j}\), where \(Q_{t-1,0}, Q_{t-1,1} > 0\) are defined in (4.10). We can assume that \(\mathcal{R}'_{t} = \mathcal{R}_{t}\), and \(\forall \epsilon > 0:\ \mathbf{r}'_{t-\epsilon} \notin \mathcal{C}_{q,t}\) (else, there exists an \(\epsilon > 0\) such that \(\mathbf{r}_{t-\epsilon} \in \mathcal{C}_{q,t}\), and this is the first case). Thus, we can conclude that \(\mathcal{R}'_{t}\) is equal to

\[
\max_{\mathbf{p}'_{t,0}, \ldots, \mathbf{p}'_{t,q-1}} \left\{ \sum_{j=0}^{q-2} Q_{t-1,j} H \left( \left( \sum_{k=0}^{j} p'_{t,k}, p'_{t,j+1}, \ldots, p'_{t,q-1} \right) \right) \right\},
\]

where \(p'_{t,j}\) is a probability to write symbol \(j\) on the \(t\)-th write, and \(Q_{t-1,j}\) is defined in (4.10). By the proof of Lemma 10, \(\mathbf{r}' \in \mathcal{C}_{q,t}\) implies \(\mathbf{r}' \in C_{q,t}\), where \(Q_{t-1,j}\), the probabilities to have symbol \(j\) after the \((t-1)\)-th write, are equals in both regions, \(C_{q,t}\) and \(\mathcal{C}_{q,t}\). Let us define \(\hat{\mathcal{R}}_{t} \triangleq \sum_{j=0}^{q-2} Q_{t-1,j} \log(q-j)\). Clearly \((\mathcal{R}'_{1}, \mathcal{R}'_{2}, \ldots, \mathcal{R}'_{t}) \in C_{q,t}\), and we prove now that \(\mathcal{R}'_{t} < \hat{\mathcal{R}}_{t}\). Note that for each \(\mathbf{x}\), a probability vector of length \(m\),

\[
H \left( x_{0}, x_{1}, \ldots, x_{m-1} \right) \leq \log(m),
\]

with equality if and only if \(x_{i} = \frac{1}{m}\) for all \(i \in [m]\). Additionally, \(\mathcal{R}'_{t} = \hat{\mathcal{R}}_{t}\) if
Lemma 14. If for all $j \in [q-2]$

$$Q_{t-1,j} H \left( \sum_{k=0}^{j} p_{t,k}, p_{t,j+1}, \ldots, p_{t,q-1} \right) = Q_{t-1,j} \log(q-j).$$

Recall that $Q_{t-1,0}, Q_{t-1,1} > 0$. Thus, from $j = 0$ we derive $p_{t,k} = \frac{1}{q}$ for all $k \in [q]$, and similarly from $j = 1$ we have $p_{t,k} = \frac{1}{q-1}$ for all $2 \leq k \in [q]$, and $p_{t,0} + p_{t,1} = \frac{1}{q}$. These two conditions cannot hold simultaneously. Thus, we conclude that $R_t = R_t^t < \hat{R}_t$.

Therefore, one can readily verify that there exists $\epsilon > 0$ such that $r_{t,\epsilon} \in C_{q,t}$.

We can extend Lemma 12 so it is possible to increase each coordinate in a rate tuple.

**Lemma 13.** If $r = (R_1, R_2, \ldots, R_i, \ldots, R_t) \in \hat{C}_{q,t}$ and $0 \neq (R_1, R_2, \ldots, R_{t-1})$, then there exists an $\epsilon > 0$ such that $r_{i,\epsilon} = (R_1, R_2, \ldots, R_i + \epsilon, \ldots, R_t) \in C_{q,t}$.

**Proof.** If there exists $\epsilon > 0$ such that $r_{i,\epsilon} \in \hat{C}_{q,t}$, then by Lemma 10, the claim is obviously true. Otherwise we prove it by induction of $i$, where the base case is $i = t$ which was proved in Lemma 12. By induction hypothesis, for all $\ell \in \{i+1, i+2, \ldots, t\}$ there exists $\epsilon_\ell > 0$ such that $r_{\ell,\epsilon_\ell} \in C_{q,t}$.

Thus, by time-sharing technique, for $\epsilon' = \min \{\epsilon_\ell/(t-i)\}_{\ell=i+1}^t$ we have $r_1 = (R_1, \ldots, R_i, R_{i+1} + \epsilon', R_{i+2} + \epsilon', \ldots, R_t + \epsilon') \in C_{q,t}$. By Lemma 11, there exists $r' = (R'_1, R'_2, \ldots, R'_t) \in \hat{C}_{q,t}$ such that $r \leq r'$ attained by $p_{i,j}$, $1 \leq i \leq t$, $j \in [q]$, where $Q_{i-1,0}, Q_{i-1,1} > 0$. By the same techniques as in the previous lemma, we can prove that there exists $\epsilon'' > 0$ such that $r_2 = (R'_1, R'_2, \ldots, R'_{t-1}, R'_t + \epsilon'', 0, \ldots, 0) \in C_{q,t}$. Using $r_1, r_2 \in C_{q,t}$ and time sharing method, we can conclude that there exists an $\epsilon > 0$, such that $r \leq r^* = (R'_1, R'_2, \ldots, R'_{t-1}, R'_t + \epsilon, R'_{t+1}, \ldots, R_t^t) \in C_{t,q}$, and therefore $r_{i,\epsilon} \in C_{t,q}$.

Based on Lemma 13 by using time-sharing technique, we conclude the following lemma.

**Lemma 14.** If $(R_1, R_2, \ldots, R_t) \in \hat{C}_{q,t}$ then there exists an $\epsilon > 0$ such that $(R_1 + \epsilon, R_2 + \epsilon, \ldots, R_t + \epsilon) \in C_{q,t}$.
Figure 4.2: A comparison between the capacity regions $C_{3,2}$ (models 1 and 2 – the outer line) and $\hat{C}_{3,2}$ (model 3 – the inner line).

Finally, the next corollary summarizes the discussion above.

**Corollary 6.** For all $q > 2$ and $t \geq 2$, $C_{q,t}^{3,\epsilon} \subseteq C_{q,t}^{\epsilon}$ and $R_{q,t}^{3,\epsilon} < R_{q,t}^{\epsilon}$.

To illustrate the results in Corollary 6, we show in Fig. 4.2, the capacity regions of $C_{3,2}, \hat{C}_{3,2}$ which compare between models 1 and 2 and model 3.

Fu and Vinck [8] found the probabilities which attain the maximum sum-rate for models 1 and 2. Even though we cannot carry the same analysis for model 3, we can still have the following result for the probabilities that achieve the maximum sum-rate in this case.

**Lemma 15.** Assume that $r \in \hat{C}_{q,t}$ achieves the maximum sum-rate. Then, there exist probabilities $p_{i,j}$ for $1 \leq i \leq t$, $j \in [q]$ which correspond to the rate tuple $r$ and $p_{t,0} = p_{t,1}$.

Proof. According to Lemma 11, if $r \in \hat{C}_{q,t}$ then there exists $r' = (R'_1, R'_2, \ldots, R'_t) \in \hat{C}_{q,t}$ such that $r \leq r'$ and $r'$ attained by $p_{i,j}$, $1 \leq i \leq t$, $j \in [q]$, where $Q_{t-1,0}, Q_{t-1,1} > 0$. Note that $r = r'$ since $r$ is maximum.
sum-rate point, and
\[
\mathcal{R}_t = \left\{ \sum_{j=0}^{q-2} Q_{t-1,j} H \left( \sum_{k=0}^{j} p_{t,k}, p_{t,j+1}, \ldots, p_{t,q-1} \right) \right\},
\]

All the summands in \( \mathcal{R}_t \) equation except for the first, use only the value \( p_{t,0} + p_{t,1} \), while the first summand achieves maximum for \( p_{t,0} = p_{t,1} \). Since \( r \) is a maximum sum-rate point, we can conclude that \( p_{t,0} = p_{t,1} \).

Even though we do not know the exact maximum sum-rate for non-binary WOM in models 3 and 4, it is still possible to derive a lower and upper bound in order to have better estimations on these values. The following result is proved by similar techniques from [11].

**Lemma 16.** For all \( q \geq 2, t \geq 1, k \in \{3, 4\}, x \in \{z, \epsilon\} \):

\[
(q-2)\mathcal{R}_{2, \lfloor \frac{t}{q-1} \rfloor}^x + \mathcal{R}_{2, t-(q-2) \lfloor \frac{t}{q-1} \rfloor}^x \leq \mathcal{R}_{q,t}^x \leq (q-1)\mathcal{R}_{2, t}^x.
\]

**Proof.** The left inequality proved by using a construction described in [11]. Once we have binary WOM codes in model \( k \), \( C_1 \) for \( t_1 = \lfloor \frac{t}{q-1} \rfloor \) writes and \( C_2 \) for \( t_2 = t - (q-2) \lfloor \frac{t}{q-1} \rfloor \) writes, we can construction a \( q \)-ary \( t \)-write WOM code by using the \( q \) levels “layer by layer”. The \( t \) rounds are divided to \( q-1 \) stages where each stage (except the last) contains \( t_1 \) writes. On the first stage, only 0, 1 symbols are used according to \( C_1 \), on the next stage only the 1, 2 levels are used according to \( C_1 \) by mapping 0 \( \mapsto 1, 1 \mapsto 2 \), and so on. The last binary WOM code that will be used is \( C_2 \) contains \( t - (q-2) \lfloor \frac{t}{q-1} \rfloor \) writes.

The proof of the right inequality consists of a reduction. Given \( C \), an \([n, t; M_1, \ldots, M_{q}]^\oplus_{p_e} (1 \leq k \leq 4)\) WOM code, with sum-rate \( \mathcal{R}_{q,t}^{\text{sum}} = \sum_{i=1}^{t} \frac{\log(M_i)}{n} \), we can construct \( C' \), an \([n(q-1), t; M_1, \ldots, M_{q}]^\oplus_{p_e} \) WOM code, with sum-rate \( \mathcal{R}_{q,t}^{\text{sum}} = \frac{\sum_{i=1}^{t} \log(M_i)}{n(q-1)} = \frac{\mathcal{R}_{q,t}^{\text{sum}}}{q-1} \). Thus, we have

\[
\mathcal{R}_{q,t}^{\text{sum}} = (q-1)\mathcal{R}_{2, t}^{\text{sum}} \leq (q-1)\mathcal{R}_{q,t}^x,
\]

where \( x = z \) if \( p_e = 0 \), otherwise \( x = \epsilon \).

Now we describe the encoding and decoding maps \( C' \). Let \( f : [q] \rightarrow \{0, 1\}^{q-1} \) be the mapping, where \( j \rightarrow 1^j0^{q-1-j} \), and for \( v \in [q]^n \), \( f(v) \) is the
concatenating the values \( f(v_1), f(v_2), \ldots, f(v_n) \). Denote by \( \mathcal{V} \) the image of \( f \), i.e., \( \mathcal{V} = \{0^{q-1-j}1^j : j \in [q]\} \). Note that \( f : [q] \to \mathcal{V} \) is bijective, and so invertible.

Let \( \mathcal{E}_i \) and \( \mathcal{D}_i \) be the encoding and decoding maps of \( C \) of the \( i \)-th write, \( 1 \leq i \leq t \). The encoder on the \( i \)-th write of \( C' \) receives a message \( m \in [M_i] \) to be encoded to the memory and programs the cells with the vector \( f(\mathcal{E}_i(m)) \), given by applying \( f \) on the result of the encoding map \( \mathcal{E}_i \). We denote the cell-state vector after the \( i \)-write by \( c_i \). It is easy to verify that \( m = \mathcal{D}_i(f^{-1}(c_i)) \) with probability at least \( 1 - p_{e_i} \).

By Corollary 6 we have that \( R_{q,t}^{\Omega,\epsilon} < R_{q,t}^{\Omega,2} \) for each \( q > 2 \) and \( t > 1 \). However, in the sequel, we state that the difference between these maximum sum-rates is bounded by a constant.

**Lemma 17.** For all \( q \) and \( t \) the following holds

\[
\mathcal{D}_{q,t} \triangleq R_{q,t}^{\Omega,2} - R_{q,t}^{\Omega,\epsilon} \leq (q - 1) \log(q - 1),
\]

that is the difference between the maximum sum-rates of an \( \epsilon \)-error \( q \)-ary \( t \)-write WOM in model 3 and a \( q \)-ary \( t \)-write WOM in models 1 and 2 is bounded by \( \mathcal{D}_{q,t} \leq (q - 1) \log(q - 1) \) for each \( t \).

**Proof.** On one hand

\[
R_{q,t}^{\Omega,2} = \log \left( \frac{q - 1 + t}{q - 1} \right) \leq (q - 1) \log(t + 1),
\]

and on the other hand, by Lemma 16,

\[
R_{q,t}^{\Omega,\epsilon} \geq (q - 1) \log \left( \left\lfloor \frac{t}{q-1} \right\rfloor + 1 \right).
\]

Thus, we have

\[
\mathcal{D}_{q,t} \leq (q - 1)(\log(t + 1) - \log(\left\lfloor \frac{t}{q-1} \right\rfloor + 1)) \\
\leq (q - 1) \log(q - 1).
\]

\( \Box \)

In Fig. 4.3, we compare between the maximum sum-rates in these models for \( q = 3, 4 \).
As a consequence of Lemma 16 we conclude with the following corollary, which illustrates the huge difference between the maximum sum-rates of model 4 and the others.

**Corollary 7.** For all $q \geq 2$, $R_{q,t}^{4,\epsilon} \leq (q - 1) \cdot 2.37$ and therefore $\lim_{t \to \infty} R_{q,t}^{4,\epsilon} \leq (q - 1) \cdot 2.37 < \infty$.

**Proof.** By Lemma 16, $R_{q,t}^{4,\epsilon} \leq (q - 1)R_{2,t}^{3,\epsilon}$. The maximum sum-rate, $R_{2,t}^{3,\epsilon}$, is given by (4.5) [26], where for all $t \geq 1$, $R_{2,t}^{3,\epsilon} \leq \frac{\pi^2}{6 \ln 2} \approx 2.37$, and $\lim_{t \to \infty} R_{2,t}^{3,\epsilon} = \frac{\pi^2}{6 \ln 2} \approx 2.37$. Thus, we have $\forall q: R_{q,t}^{4} \leq (q - 1)R_{2,t}^{3,\epsilon} \leq (q - 1) \cdot 2.37$. $\square$

### 4.8 Conclusion and Open Problems

In this paper, we studied constructions and the capacity region of write-once memories. We first presented constructions of WOM codes for models 3 (EU:DI) and 4 (EU:DU). We then studied the capacity region and maximum sum-rate of non-binary WOM for all four models both for the zero-error and the $\epsilon$-error cases. While the results in the paper expand the state of
the art knowledge on write-once memories, there are still several interesting problems which are left open. Some of them are summarized as follows:

1. Calculating the zero-error capacity region and maximum sum-rate in model 3 (the EU:DI model) for all $q \geq 2$ and $t \geq 2$.

2. Calculating the zero-error and the $\epsilon$-error capacity regions and maximum sum-rate in these two cases, in model 4 (the EU:DU model) for all $q \geq 2$ and $t \geq 2$. Note that only the $\epsilon$-error maximum sum-rate for binary $t$-write WOM is known [26].

3. Find zero-error and $\epsilon$-error WOM code constructions for models 3 and 4, both for binary and non-binary. In particular, it is left open to improve the zero-error construction from Example 6 for model 4 with sum-rate 1.29 and the zero-error construction from Example 7 for model 3 with sum-rate 1.33.

Acknowledgment

The authors thank the anonymous reviewer and the associate editor Prof. Jun Chen for their valuable comments and suggestions, which have contributed for the clarity of the paper and its presentation.

Appendix A  Non-Binary Two-Write in Model 4

In order to construct non-binary WOM codes in model 4, we follow an analogy to the one from Theorem 4 which uses codes in the $Z$ channel for the binary setup. More specifically, we consider here non-binary asymmetric errors, which can only increase the level of each cell. That is, if a $q$-ary cell is stored with the value $b \in [q]$ and a $q$-ary asymmetric error has occurred, then only the values $\{b + 1, \ldots, q - 1\}$ can be received. Note that the $Z$ channel is a special case of this model for $q = 2$. Given two vectors $u$, $v \in [q]^n$, the Manhattan distance between $u$ and $v$, $d_M(u, v)$, is defined to be $d_M(u, v) = \sum_{i=0}^{n-1}|u_i - v_i|$. The Manhattan weight of $u$, $w_M(u)$ is defined to be $w_M(u) = d_M(u, 0) = \sum_{i=0}^{n-1}u_i$.

Let $u \in [q]^n$ be a stored cell-state vector and $u + e \in [q]^n$ be the received vector where $e \geq 0$ is the error vector. We say that a length-$n$ code $C$ over $q$-ary symbols with $M$ codewords is an $(n, M, \tau)_q$ asymmetric-error-correcting
code if it can correct any asymmetric error vector with Manhattan weight at most \( \tau \). We denote the number of \( q \)-ary length-\( n \) vectors of Manhattan weight at most \( \tau \) by \( B_q(n, \tau) \). Using the inclusion-exclusion principle, we conclude

\[
B_q(n, \tau) = \min\{n, \lfloor \frac{\tau}{q} \rfloor \} \sum_{k=0}^{\min\{n, \lfloor \frac{\tau}{q} \rfloor \}} (-1)^k \binom{n + \tau - kq}{k, n - k, \tau - kq}.
\]

Note that a length-\( n \) code over \( q \)-ary symbols with \( M \) codewords and minimum Manhattan distance \( d \) is an \((n, M, \lfloor \frac{d-1}{2} \rfloor)\)q asymmetric-error-correcting code.

The Lee metric is more investigated than the Manhattan metric, and may be used here. Given two vectors \( u, v \in [q]^n \), the Lee distance between \( u \) and \( v \), \( d_L(u, v) \), is defined to be

\[
d_L(u, v) = \sum_{i=0}^{n-1} \min\{|u_i - v_i|, q - |u_i - v_i|\}.
\]

Note that for each two vectors \( u, v \in [q]^n \), it holds that \( d_L(u, v) \leq d_M(u, v) \).

Therefore, a code \( C \) with minimum Lee distance \( d \), has minimum Manhattan distance \( d' \), where \( d' \geq d \).

**Theorem 11.** Let \( C \) be an \((n, M, \tau)\)q asymmetric-error-correcting code. Then there exists an \([n, 2; M_1, M_2]^{(\frac{d-1}{2})}\)q WOM code, where \( M_1 = B_q(n, \tau) \) and \( M_2 = M \).

**Proof.** The proof will consist of describing the encoding and decoding maps of the WOM code. On the first write \( M_1 \) messages can be written by simply programming vectors with Manhattan weight at most \( \tau \).

Let \( E, D \) be the encoding, decoding map of the error-correcting code \( C \), respectively. Then, the encoder of the second write receives a message \( m \in [M_2] \) to be encoded to the memory and programs the cells by applying the encoding map \( E(m) \) of \( C \). Let \( c_1 \) and \( c_2 \) be the memory cell-state vectors after the first and the second write, respectively. Denote by \( e \) the error vector \( c_2 - E(m) \). Note that \( c_2 = \max\{c_1, E(m)\} \), and \( e \leq c_1 \) We have the following three observations:

1. \( E(m) \) is a codeword in \( C \),
2. \( c_2 \geq E(m) \),
3. \( d_M(E(m), c_2) = w_M(e) \leq w_M(c_1) \leq \tau \).

That is, the cell-state vector \( c_2 \) is the outcome of the error vector \( e \). Since the code \( C \) is capable of correcting an asymmetric error vector with Manhattan weight at most \( \tau \), we have that \( D(c_2) = m \), as required. \( \square \)
Even though the result from Theorem 11 provides us with a specific construction of non-binary WOM codes in model 4, it is not clear what the asymptotic result of this construction is. That is, given $n$ large enough what the best choice to choose the value of $\tau$ is. Furthermore, codes in the Manhattan distance are not easy to construct, and our attempts to find efficient codes in the Lee metric for this construction were unsuccessful. We also note that we are not limited to using this type of asymmetric non-binary error-correcting codes. We could use codes which correct limited magnitude errors [5] as long as the programmed cell-state vectors on the first write have the same limited magnitude.

### Appendix B Non-binary Two-Write in Model 3

In order to construct non-binary WOM codes in model 3, we follow an analogy to the one from Theorem 7 which uses codes in the erasure channel for the binary setup. The techniques are similar to those described in model 4, Appendix A. More specifically, we consider here non-binary erasures. That is, if a $q$-ary cell is stored with the value $b \in [q]$ and an asymmetric erasure has occurred, then a value from the set $\{b, \ldots, q-1\}$ can be received with the erasure mark. Thus, if the value $a \in [q]$ is received with the erasure mark, then the correct value is in $[a+1]$. Note that the binary erasure channel is a special case of this model for $q = 2$.

Let $u \in [q]^n$ be a stored cell-state vector and $v \in [q]^n$ be the received vector with erasures locations $S \subseteq [n]$. Denote by $e \in [q]^n$ the erasure vector, i.e., $e_i = v_i$ if $i \in S$, and otherwise $e_i = 0$. We say that a length-$n$ code $C$ over $q$-ary symbols with $M$ codewords is an $(n, M, \tau)_q$ asymmetric-erasure-correcting code if it can correct any asymmetric erasure vector with Manhattan weight at most $\tau$. Note that a length-$n$ code over $q$-ary symbols with $M$ codewords and minimum Manhattan distance $d$ is an $(n, M, d-1)_q$ asymmetric-erasure-correcting code.

**Theorem 12.** Let $C$ be an $(n, M, \tau)_q$ asymmetric-erasure-correcting code. Then there exists an $[n, 2; M_1, M_2]^{\text{3D}}_q$ WOM code, where $M_1 = B_q(n, \tau)$ and $M_2 = M$.

**Proof.** The proof will consist of describing the encoding and decoding maps of the WOM code. On the first write $M_1$ messages can be written by simply
programming vectors with Manhattan weight at most $\tau$. Let $E, D$ be the encoding, decoding map of the code $C$, respectively. Then, the encoder of the second write receives a message $m \in [M_2]$ to be encoded to the memory and programs the cells by applying the encoding map $E(m)$ of $C$. Let $c_1$ and $c_2$ be the memory cell-state vector after the first, second write, respectively. We have that $c_2 = \max\{c_1, E(m)\}$. The input to the decoder on the second write is $c_1$ and $c_2$. Let $S$ be the set of the erasures' locations, i.e., $S = \{i : c_{1,i} = c_{2,i} > 0\}$, and $e$ be the erasure vector. We have the following four observations.

1. $E(m)$ is a codeword in $C$,
2. $c_2 \geq E(m)$,
3. $w_M(e) = \sum_{i \in S} c_{2,i} = \sum_{i \in S} c_{1,i} \leq w_M(c_1) \leq \tau$,
4. the set $S$ is known to the decoder.

That is, the cell-state vector $c_2$ is the outcome of an erasure vector with Manhattan weight at most $\tau$ where the erasures locations are known to the decoder. Since the code $C$ is capable to correct an erasure vector of Manhattan weight $\tau$, there exists exactly one codeword $c \in C$ which match to $c_2$ on the indices in $[n] \setminus S$ and $c \leq c_2$. Thus, we have that $D(c_2) = m$, as required.

The disadvantages of the construction in Theorem 12 are essentially the same as described in Appendix A for model 4. Recall that a length-$n$ code over $q$-ary symbols with $M$ codewords and Manhattan distance $d$, is an $(n, M, \lfloor d - 1/2 \rfloor)_q$ asymmetric-error-correcting code, and an $(n, M, d - 1)_q$ asymmetric-erasure-correcting code. Thus, given such a code, the construction in Theorem 12 for model 3, yields a better sum-rate than the construction from Theorem 11 for model 4. Nevertheless, we couldn’t find efficient codes also for this model, by the same reasons.
Bibliography


Part II

Unpublished Journal Papers
Chapter 5

Local Rank Modulation for Flash Memories

Michal Horovitz and Tuvi Etzion

5.A Unpublished Full Version

Abstract

Local rank modulation scheme was suggested for representing information in flash memories in order to overcome drawbacks of rank modulation. For $0 < s \leq t \leq n$ with $s$ divides $n$, an $(s, t, n)$-LRM scheme is a local rank modulation scheme where the $n$ cells are locally viewed cyclically through a sliding window of size $t$ resulting in a sequence of small permutations which requires less comparisons and less distinct values. The gap between two such windows equals to $s$. In this work, encoding, decoding, and asymptotic enumeration of the $(1, t, n)$-LRM scheme is studied.

5.A.1 Introduction

Flash memory is a non-volatile technology that is both electrically programmable and electrically erasable. It incorporates a set of cells maintained at a set of levels of charge to encode information. While raising the charge level of a cell is an easy operation, reducing the charge level requires
the erasure of the whole block to which the cell belongs. For this reason charge is injected into the cell over several iterations. Such programming is slow and can cause errors since cells may be injected with extra unwanted charge. Other common errors in flash memory cells are due to charge leakage and reading disturbance that may cause charge to move from one cell to its adjacent cells. In order to overcome these problems, the novel framework of rank modulation was introduced in [3]. In this setup, the information is carried by the relative ranking of the cells’ charge levels and not by the absolute values of the charge levels. Denote the charge level in the ith cell by \( c_i \), \( 0 \leq i < n \), and \( c = (c_0, c_2, \ldots, c_{n-1}) \) is the sequence of the charges in \( n \) cells. A codeword in this scheme is the permutation defined by the order of the charge levels, from the highest one to the lowest one, e.g. if \( n = 5 \) and \( c = (3, 5, 2, 7, 10) \) then the permutation, i.e., the codeword in the rank modulation scheme, is \([5, 4, 2, 1, 3]\). This allows for more efficient programming of cells, and coding by the ranking of the cells’ charge levels is more robust to charge leakage than coding by their actual values. The push-to-the-top operation is a basic minimal cost operation in the rank modulation scheme by which a single cell has its charge level increased such that it will be the highest of the set. Lot of research on the rank modulation scheme was done since its introduction less than ten years ago, see [1, 2] and references therein.

A drawback of the rank modulation scheme is the need for a large number of comparisons when reading the induced permutation. Furthermore, distinct \( n \) charge levels are required for a group of \( n \) cells. The local rank modulation scheme was suggested [2] in order to overcome these problems. In this scheme, the \( n \) cells are locally viewed through a sliding window, resulting in a sequence of permutations for a much smaller number of cells which requires less comparisons and less distinct values. For \( 0 < s \leq t \leq n \), where \( s \) divides \( n \), the \((s, t, n)\)-LRM scheme, defined in [2, 5], is a local rank modulation scheme over \( n \) physical cells, where \( t \) is the size of each sliding window and \( s \) is the gap between two such windows. In this scheme the permutations are over \( \{1, 2, \ldots, t\} \), i.e., from \( S_t \), and the push-to-the-top operation merely raises the charge level of the selected cell above those cells which are comparable with it. We say a sequence with \( \frac{n}{2} \) permutations from \( S_t \) is an \((s, t, n)\)-LRM scheme realizable if it can be demodulated to a sequence of charges in \( n \) cells under the \((s, t, n)\)-LRM scheme. Except for the
degenerate case where \( s = t = n \), not every sequence is realizable.

The \((1, 2, n)\)-LRM scheme was defined in [2] in order to get the simplest hardware implementation. All demodulated sequences of permutations in this scheme are realizable, except for the two sequences of permutations in which all permutations are the same. Hence, \(2^n - 2\) sequences of permutations are realizable in this scheme. But, since only two permutations are used in this scheme, it follows that this scheme is relatively very weak. Therefore, we are interested in \((1, t, n)\)-LRM schemes, where \( t \geq 3 \).

In this paper we focus on the \((1, t, n)\)-LRM schemes for \( t \geq 3 \), and suggest a demodulation method for these schemes. The \((1, t, n)\)-LRM scheme is a local rank modulation scheme over \( n \) physical cells, where the size of each sliding window is \( t \), and each cell starts a new window. Since the size of a sliding window is \( t \), demodulated sequences of permutations in this scheme contain \( t! \) permutations. Therefore, we need \( t! \) symbols to present the demodulated sequences of permutations.

Let \( s = (s_1, s_2, \ldots, s_{t!}) \) be an order of the \( t! \) permutations from \( S_t \), and \( \Sigma = \{1, 2, \ldots, t!\} \) be an alphabet where \( i \) represents the permutation \( s_i \). A sequence \( \alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) \) over the alphabet \( \Sigma \) is called a base-word in the \((1, t, n)\)-LRM scheme, and it is realizable, if there exists a sequence of charge levels \( c = (c_0, c_1, \ldots, c_{n-1}) \), such that for each \( i, 0 \leq i < n \), \( \alpha_i \) represents the permutation induced by \( c_i, c_{i+1}, \ldots, c_{i+t-1} \), where indices are taken modulo \( n \) for the charge levels vector \( c \) and for the base-word \( \alpha \).

In this paper a mapping method, in which each a base-word \( \alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) \) over the alphabet of size \( t! \), is mapped to a codeword \( g = (g_0, g_1, \ldots, g_{n-1}) \) over an alphabet of size \( t \), will be presented. A codeword is called legal if there exists a realizable base-word which is mapped to it. We have to make sure that two distinct realizable base-words are mapped into two distinct legal codewords. Note again, that the indices in the base-words, charge levels, and the codewords are taken modulo \( n \).

Let \( M_t \) be the number of legal codewords in the \((1, t, n)\)-LRM scheme. Since a symbol in a codeword is from an alphabet with \( t \) letters, it follows that \( M_t \leq t^n \). But, this upper bound is not tight since there exist illegal codewords. We prove that this upper bound on \( M_t \) is asymptotically tight, i.e. \( \lim_{n \to \infty} \frac{M_t}{t^n} = 1 \).

The rest of this paper is organized as follows. The encoding, decoding and asymptotic enumeration of the \((1, 3, n)\)-LRM scheme is presented in
Section 5.A.2. Generalizations, especially for the enumeration technique for the \( (1, t, n) \)-LRM scheme, \( t > 3 \), is given in Section 5.A.3. In Section 5.A.4 conclusion and problems for future research are presented.

5.A.2 The \( (1, 3, n) \)-LRM scheme

In the \( (1, 3, n) \)-LRM scheme the size of each sliding window is 3. Therefore, an alphabet of size 3! is required to present the demodulated sequences of permutations.

\[
\begin{align*}
  s_1 &= [1, 2, 3] & s_2 &= [1, 3, 2] \\
  s_3 &= [2, 1, 3] & s_4 &= [3, 1, 2] \\
  s_5 &= [2, 3, 1] & s_6 &= [3, 2, 1]
\end{align*}
\]

The alphabet of the base-words is \( \Sigma = \{1, 2, \ldots, 6\} \), where the symbol \( \ell \) represents the permutation \( s_\ell \). Let \( \alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) \) be a base-word. Note that the last two cells which determine \( \alpha_i \), \( 0 \leq i < n \), are the first two cells which determine \( \alpha_{i+1} \), i.e., the permutation related to \( \alpha_{i+1} \) is obtained from \( \alpha_i \) by the following way. The symbol 1 in the permutation related to \( \alpha_i \) is omitted, the symbols 2, 3 in the permutation are replaced with 1, 2, respectively, and a new symbol 3 is inserted before 1, 2, between them, or after both of them. Therefore, given \( \alpha_i \), there are exactly 3 options for \( \alpha_{i+1} \).

Let \( \Sigma^1 = \{1, 3, 5\} \) and \( \Sigma^2 = \{2, 4, 6\} \) be a partition of \( \Sigma \) into the even and the odd symbols, respectively. Note that for each \( \Sigma^i, i \in \{1, 2\} \), the permutations related to the symbols in \( \Sigma^i \) agree on the order of cells 2 and 3. Therefore, they also agree on the three possibilities of their succeeding permutation. Denote the set of symbols of these succeeding permutations by \( \tilde{\Sigma}^i \). It is readily verified that \( \tilde{\Sigma}^1 = \{1, 2, 4\} \) and \( \tilde{\Sigma}^2 = \{3, 5, 6\} \).

The base-word \( \alpha \) is mapped to a codeword \( g = (g_0, g_1, \ldots, g_{n-1}) \) over the alphabet \( \{0, 1, 2\} \). Given the charge levels \( c_i, c_{i+1}, c_{i+2} \), the permutation \( \alpha_i \) is uniquely determined. If we are given now also the charge level \( c_{i+3} \), then its rank among \( c_{i+1}, c_{i+2} \) uniquely determines \( g_{i+1} \). Therefore, \( \alpha_{i+1} \) can be deduced from \( \alpha_i \) and \( g_{i+1} \) instead of \( c_{i+1}, c_{i+2}, c_{i+3} \). The relations between \( \alpha_{i-1}, \alpha_i, \) and \( g_i \), where \( 0 \leq i < n \), are presented in Table 5.1. This table induces a mapping from the realizable base-words to the codewords. As mentioned before, given \( \alpha_{i-1} \), there are three options for \( \alpha_i \). In all these options the sub-permutation of \( \{1, 2\} \) is the same, and the difference is the
index of the symbol 3 in the permutation related to $\alpha_i$. Thus, $g_i$ represents the index of the symbol 3 in this permutation and it is equal to the number of symbols which are to the right of the symbol 3 in the permutation related to $\alpha_i$. In other words, $g_i$ represents the relation between $c_{i+2}$, the charge level in cell $i + 2$, and the charge levels in the two cells which proceed it, i.e., $c_i$ and $c_{i+1}$.

\[
\begin{array}{|c|c|c|c|}
\hline
\alpha_{i-1} \in \Sigma^1 & \alpha_i = 1 & \alpha_i = 2 & \alpha_i = 4 \\
\hline
\alpha_{i-1} \in \Sigma^2 & \alpha_i = 3 & \alpha_i = 5 & \alpha_i = 6 \\
\hline
\end{array}
\]

Table 5.1: The encoding key of the $(1, 3, n)$-LRM scheme.

Note that there might exist non-realizable base-words which are mapped to codewords by this method. A base-word $\alpha$, which can be mapped to a codeword by this method, satisfies only the dependencies between $\alpha_i$ and $\alpha_{i+1}$, $0 \leq i < n$, but it still can be non-realizable. The $n$ cells are viewed cyclically, i.e., the charge levels of the last two cells, $c_{n-2}$ and $c_{n-1}$, are compared with the charge level in the first cell, $c_0$, to determine $\alpha_{n-2}$. The same works for the three charge levels $c_{n-1}$, $c_0$, and $c_1$ to determine $\alpha_{n-1}$. Therefore, there might exists a non-realizable dependency between the charge levels in the last two cells and the charge levels in the first two cells. Such a non-realizable base-word will be called a cyclically non-realizable base-word.

**Example 8.** The following base-words are cyclically non-realizable. Recall, that a codeword is called legal if there exists a realizable base-word which is mapped to it.

- $(6, 6, \ldots, 6)$ - the charge levels are increased cyclically, which is impossible. This base-word is mapped to the illegal codeword $(2, 2, \ldots, 2)$.

- $(2, 5, 2, 5, \ldots, 2, 5)$ where $n$ is even - the charge level of each cell is between the charge levels of the two cells which proceed it, where the charge levels are taken cyclically. This base-word is mapped to the illegal codeword $(1, 1, \ldots, 1)$. 

93
• $(1,1,\ldots,1,2,3)$ - the prefix $(1,1,\ldots,1)$ means that the charge levels always decrease, and therefore $c_{n-1} < c_{n-2} < c_1 < c_0$. Hence, in this case the only possible permutations for $\alpha_{n-2}$ and $\alpha_{n-1}$ are $[3,1,2] = s_4 \quad \text{and} \quad [2,3,1] = s_5$, respectively. That is, the only realizable base-word which completes the prefix $(1,1,\ldots,1)$ is $(1,1,\ldots,1,4,5)$. Thus, $(1,1,\ldots,1,2,3)$ is a non-realizable base-word, which is mapped by Table 5.1 to the illegal codeword $(0,0,\ldots,0,1,0)$. The realizable base-word $(1,1,\ldots,1,4,5)$ is mapped to the legal codeword $(0,0,\ldots,0,2,1)$.

**Theorem 13.** Table 5.1 provides a one-to-one mapping between the realizable base-words and the legal codewords.

**Proof.** Obviously, each base-word is mapped to exactly one codeword. Now, we prove that the other direction is also true, i.e. given a legal codeword $g$, there is a unique base-words which is mapped to $g$. By Example 8, $(1,1,\ldots,1)$ is an illegal codeword. Hence, given a legal codeword $g = (g_0,g_1,\ldots,g_{n-1})$, there exists $0 \leq i < n$, such that $g_i \in \{0,2\}$. If $g_i = 0$ then by Table 5.1 we have $\alpha_i \in \{1,3\}$, i.e., $\alpha_i$ is odd. Therefore, given $g_{i+1}$, the permutation $\alpha_{i+1}$ is determined by an entry in the first row of Table 5.1, where the column is chosen by the value of $g_{i+1}$. Similarly, if $g_i = 2$ then $\alpha_i \in \{4,6\}$, i.e., $\alpha_i$ is even. Hence, $\alpha_{i+1}$ is determined by an entry in the second row of Table 5.1, where the column is chosen by the value of $g_{i+1}$. Now, it is easy to determine the symbols of the base-word $\alpha_{i+2}, \alpha_{i+3}, \ldots, \alpha_{i+n-1}, \alpha_{i+n} = \alpha_i$ one by one from Table 5.1 in this cyclic order. 

Note that decoding a given codeword $g$ to a base-word $\alpha$ does not guarantee that $g$ is legal. For some illegal codewords the decoding procedure fails, while for the others it succeeds without a notification about the illegality of the input $g$. Let $i, 0 \leq i < n$, be the starting point of the decoding algorithm as described in the proof of Theorem 13. At the first step of the algorithm, $\alpha_i$ has two options $(\{1,3\}$ if $g_i = 0$ and $\{4,6\}$ if $g_i = 2$, as implied by Table 5.1). At the last step, if $\alpha_i$ is not equal to one of these two optional initial
values, which was chosen in the first step, then we conclude that the given codeword is illegal. However, the algorithm may decode some cyclically non-realizable base words without realizing that it is an illegal codeword. For example, the procedure decodes the cyclically non-realizable base-word \( \alpha = (1,1,\ldots,1) \) from the illegal codeword \( g = (0,0,\ldots,0) \). Therefore, given such a codeword, it would be interesting to decide efficiently, is it a legal codeword or not? This question will be considered towards the end of this section.

To enumerate the number of legal codewords in the \((1,3,n)\)-LRM scheme we need another concept to describe the permutation defined by the current last two charge levels \( c_{i-1} \) and \( c_i \) and the rank of each one of them among \( c_0 \) and \( c_1 \).

Given a prefix of a codeword \((g_0,g_1,\ldots,g_{i-2})\) obtained by the unknown charge levels \( c_0,c_1,\ldots,c_i \), the ranking among the charge levels in the \( i \)-th cell, \( c_i \), and the first two cells, \( c_0 \) and \( c_1 \), might have a few options (at most three). These options will be denoted by 0, 1 and 2, where 0 represents that \( c_i \) is lower than \( c_0 \) and \( c_1 \), 1 represents that \( c_i \) is between them, and 2 represents that \( c_i \) is higher than both of them. For each \( i \), \( 3 \leq i < n \), consider the following two properties regarding \( c_{i-1} \) and \( c_i \):

(Q.1) the permutation \( \pi_i \) induced by \( c_{i-1} \) and \( c_i \) (either \([1,2]\) or \([2,1]\)).

(Q.2) the set of all possible pairs \((x,y)\), \( x,y \in \{0,1,2\} \), where \( x \) represents the relation between \( c_{i-1} \) and the first two charge levels, \( c_0,c_1 \), and \( y \) represents the relation between \( c_i \) and the first two charge levels, \( c_0,c_1 \).

Note, that not all the nine pairs \((x,y)\) can be obtained for a given permutation defined by (Q.1). A maximum of six pairs from the nine pairs can be obtained and this will be proved in Section 5.A.3 (see Lemma 19). We call each set of properties defined by (Q.1) and (Q.2) a state, and the state at index \( i \) (for \( c_{i-1} \) and \( c_i \)) will be denoted by \( P_i \). For the computation of the states, only the codeword \( g \) is known, while neither the charge levels nor the permutations defined by them, from which it was computed, are known. Let’s denote by \( \pi_i \), \( 1 \leq i < n \), the permutation defined by \( c_{i-1} \) and \( c_i \).
Example 9. Assume that the prefix of the codeword is $g' = (g_0, g_1, \ldots, g_{n-3}) = (2, 2, \ldots, 2)$, i.e. the charge levels are increased between any two consecutive charge levels (with a possible exception between $c_0$ and $c_1$ and an exception cyclically since $c_{n-1} > c_0$), that is, $c_0$ or $c_1$ are the smallest charge levels, $c_{n-2}$ and $c_{n-1}$ are the largest, and $c_{n-2} < c_{n-1}$. Hence, $\pi_{n-1} = [2, 1]$, and both $c_{n-2}$ and $c_{n-1}$ are larger than $c_0$ and $c_1$. Therefore, $P_{n-1} = ([2, 1], \{(2, 2)\})$. There are two possible base-words depending on the ranking between $c_0$ and $c_1$. If $c_1 < c_0$, i.e. $\pi_1 = [1, 2]$, then the base-word is $\alpha = (6, 6, \ldots, 6, 3, 2)$ and the codeword is $g = (2, 2, \ldots, 2, 0, 1)$. If $c_1 > c_0$, i.e. $\pi_1 = [2, 1]$, then the base-word is $\alpha = (4, 6, 6, \ldots, 6, 3, 1)$ and the codeword is $g = (2, 2, \ldots, 2, 0, 0)$.

Example 10. Assume that the prefix of the codeword is $g' = (1, 1, \ldots, 1, 0, 2)$ which implies that for all $j$, $2 \leq j \leq i-2$, the charge level $c_j$ is between $c_{j-1}$ and $c_{j-2}$. Therefore, with a simple induction on $j$, we conclude that $c_{i-3}$ and $c_{i-2}$ are between $c_0$ and $c_1$. Thus, $g_{i-3} = 0$ implies that $c_{i-1}$ might be between $c_0$ and $c_1$ or smaller than both of them, and $g_{i-2} = 2$ implies that $c_i$ might be between $c_0$ and $c_1$ or larger than both of them. We conclude that $P_i = ([2, 1], \{(0, 1), (0, 2), (1, 1), (1, 2)\})$.

Lemma 18. If $P_i$ and $g_{i-1}$ are given, for some $3 \leq i < n-1$, then $P_{i+1}$ is uniquely determined.

Proof. $P_i$ is characterized by the permutation $\pi_i$ in (Q.1) and the pairs in (Q.2). The permutation $\pi_i$ is defined by $c_{i-1}$ and $c_i$; and $g_{i-1}$ defines the ranking of $c_{i+1}$ among $c_{i-1}$, $c_i$. Hence, the permutation defined by $c_i$ and $c_{i+1}$ is uniquely determined and property (Q.1) for $P_{i+1}$ is well defined. Let $(x, y)$ be a possible rankings pair of $c_{i-1}$ and $c_i$ among $c_0$, $c_1$. We have to show that all the possible values for $(x, y)$ imply all the possible rankings of $c_i$ and $c_{i+1}$ among $c_0$, $c_1$. In other words, for a given $(x, y)$ the possible set of rankings $(y, z)$, $0 \leq z \leq 2$, of $c_i$ and $c_{i+1}$ among $c_0$, $c_1$ is uniquely determined. Hence, to complete the proof it is sufficient to show that all the possible rankings of $c_{i+1}$ among $c_0$, $c_1$, are uniquely determined. The permutation defined by $c_0$ and $c_1$ does not affect on the ranking of $c_{i+1}$.
among $c_0$ and $c_1$. But, to write the relations between these three charge levels we will assume w.l.o.g. that $c_0 > c_1$. We distinguish between three cases according to the value of $g_{i-1}$.

**Case 1:** If $g_{i-1} = 0$ then $c_{i+1} < c_i$ and $c_{i+1} < c_{i-1}$.

If $y = 0$ then $c_1 < c_0$, hence $c_{i+1} < c_1 < c_0$, which implies that $(0, 0)$ is the only possible pair in $(Q.2)$ obtained by $(x, y)$.

If $y = 1$ then $c_1 < c_i$ and hence $c_{i+1} < c_i < c_0$. Now, we have to consider $x$. If $x = 0$ then $c_{i+1} < c_{i-1} < c_1$, which implies that $(1, 0)$ is the possible pair. If $x = 1$ or $x = 2$ then $c_{i+1}$ can be lower or higher than $c_1$, which implies that $(1, 0)$ and $(1, 1)$ are the possible pairs in $(Q.2)$ obtained in this case.

If $y = 2$ then $c_1 < c_0 < c_i$ and hence we have to consider $x$. If $x = 0$ then $c_{i+1} < c_{i-1} < c_1$, which implies that $(2, 0)$ is the possible pair. If $x = 1$ then $c_{i+1}$ can be lower or higher than $c_1$ but it is lower than $c_0$, which implies that $(2, 0)$ and $(2, 1)$ are the possible pairs. If $x = 2$ then $c_{i+1}$ can be lower than $c_1$, between $c_0$ and $c_1$, or higher than $c_0$, which implies that $(2, 0)$, $(2, 1)$, and $(2, 2)$ are the possible pairs.

**Case 2:** If $g_{i-1} = 1$ then $c_{i+1}$ is between $c_i$ and $c_{i-1}$. By (Q.1) we know exactly the order between the three charge levels. We continue and analyse as in Case 1.

**Case 3:** If $g_{i-1} = 2$ then $c_i < c_{i+1}$ and $c_{i-1} < c_{i+1}$. This case is dual to Case 1 and hence has a dual analysis.

**Corollary 1.** If $P_i = P_j$ for some $3 \leq i < j < n-1$ and $g_{i-1} = g_{j-1}$ then $P_{i+1} = P_{j+1}$.

A state which has the maximum number of pairs, i.e. six pairs, in property $(Q.2)$ will be called a complete state. In other words, $P_i$ is a complete state if the last two charge levels, $c_{n-2}$ and $c_{n-1}$, are independent on the charge levels of the first two cells, $c_0$ and $c_1$. It is not difficult to verify that there exist exactly two complete states:

1. $S_1 : ([1, 2], \{(0, 0), (1, 0), (1, 1), (2, 0), (2, 1), (2, 2)\})$.

2. $S_2 : ([2, 1], \{(0, 0), (0, 1), (0, 2), (1, 1), (1, 2), (2, 2)\})$.

We are only interested in complete states since non-complete states might lead to a relatively small number of legal codewords. The non-complete
states and their related codewords will be omitted in the computations of the number of legal codewords which follows. Given \( g_{i-1} \), the succeeding state \( P_{i+1} \) of a state \( P_i \) which is a complete state, is given in Table 5.2. It should be noted that if \( P_i \) is a complete state then also \( P_{i+1} \) is a complete state.

<table>
<thead>
<tr>
<th>( P_i )</th>
<th>( g_{i-1} )</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_1 )</td>
<td>( S_1 )</td>
<td>( S_2 )</td>
<td>( S_2 )</td>
<td></td>
</tr>
<tr>
<td>( S_2 )</td>
<td>( S_1 )</td>
<td>( S_1 )</td>
<td>( S_2 )</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2: Succeeding states for complete states in the \((1, 3, n)\)-LRM scheme.

Given \( \pi_1 \) and \( g' = (g_0, g_1, \ldots, g_{n-4}, g_{n-3}) \), the sub-base-word \((\alpha_0, \alpha_1, \ldots, \alpha_{n-3})\) of a realizable base-word which corresponds to \( \pi_1 \) and \( g' \) is determined unambiguously. But, there exist a few possible assignments for \( \alpha_{n-2} \) and \( \alpha_{n-1} \), which correspond to the possible assignments for \( g_{n-2} \) and \( g_{n-1} \). These assignments are determined by the state \( P_{n-1} \) and the permutation \( \pi_1 \). Each assignment provides a distinct realizable base-word which is represented by the state \( P_{n-1} \) and the permutation \( \pi_1 \).

Recall, that only complete states will be considered in the computations. Table 5.3 presents the number of possible pairs \((g_{n-2}, g_{n-1})\) which can complete a given prefix of a codeword \((g_0, \ldots, g_{n-3})\) to a legal codeword, correspond to a complete state \( P_{n-1} \) in the \((1, 3, n)\)-LRM scheme.

Example 11. Let \( \pi_1 = [1, 2] \) and \( P_{n-1} = S_1 \), i.e. \( c_1 < c_0 \) and \( \pi_{n-1} = [1, 2] \), which implies that \( c_{n-1} < c_{n-2} \). We distinguish now between the six possible pairs \((x, y)\) of (Q.2) related to \( S_1 \).

1. If \((x, y) = (0, 0)\) then \( c_{n-1} < c_{n-2} < c_1 < c_0 \) which implies that \( \alpha_{n-2} = [3, 1, 2] = s_4, \alpha_{n-1} = [2, 3, 1] = s_5, g_{n-2} = 2, \) and \( g_{n-1} = 1. \)
2. If \((x, y) = (1, 0)\) then \( c_{n-1} < c_1 < c_{n-2} < c_0 \) which implies that \( \alpha_{n-2} = [3, 1, 2] = s_4, \alpha_{n-1} = [2, 3, 1] = s_5, g_{n-2} = 2, \) and \( g_{n-1} = 1. \)
3. If \((x, y) = (1, 1)\) then \( c_1 < c_{n-1} < c_{n-2} < c_0 \) which implies that \( \alpha_{n-2} = [3, 1, 2] = s_4, \alpha_{n-1} = [2, 1, 3] = s_3, g_{n-2} = 2, \) and \( g_{n-1} = 0. \)
4. If \((x, y) = (2, 0)\) then \(c_{n-1} < c_1 < c_0 < c_{n-2}\) which implies that 
\(\alpha_{n-2} = [1, 3, 2] = s_2, \, \alpha_{n-1} = [2, 3, 1] = s_5, \, g_{n-2} = 1, \text{ and } g_{n-1} = 1.\)

5. If \((x, y) = (2, 1)\) then \(c_1 < c_{n-1} < c_0 < c_{n-2}\) which implies that 
\(\alpha_{n-2} = [1, 3, 2] = s_2, \, \alpha_{n-1} = [2, 1, 3] = s_3, \, g_{n-2} = 1, \text{ and } g_{n-1} = 0.\)

6. If \((x, y) = (2, 2)\) then \(c_1 < c_0 < c_{n-1} < c_{n-2}\) which implies that 
\(\alpha_{n-2} = [1, 2, 3] = s_1, \, \alpha_{n-1} = [1, 2, 3] = s_1, \, g_{n-2} = 0, \text{ and } g_{n-1} = 0.\)

Thus, the 5 possible pairs \((\alpha_{n-2}, \alpha_{n-1})\) and \((g_{n-2}, g_{n-1})\) are given in the following table

<table>
<thead>
<tr>
<th>((x, y))</th>
<th>((\alpha_{n-2}, \alpha_{n-1}))</th>
<th>((g_{n-2}, g_{n-1}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>((0, 0), (1, 0))</td>
<td>((4, 5))</td>
<td>((2, 1))</td>
</tr>
<tr>
<td>((1, 1))</td>
<td>((4, 3))</td>
<td>((2, 0))</td>
</tr>
<tr>
<td>((2, 0))</td>
<td>((2, 5))</td>
<td>((1, 1))</td>
</tr>
<tr>
<td>((2, 1))</td>
<td>((2, 3))</td>
<td>((1, 0))</td>
</tr>
<tr>
<td>((2, 2))</td>
<td>((1, 1))</td>
<td>((0, 0))</td>
</tr>
</tbody>
</table>

**Example 12.** Let \(\pi_1 = [2, 1]\) and \(P_{n-1} = S_1, i.e. c_0 < c_1\) and \(\pi_{n-1} = [1, 2]\), which implies that \(c_{n-1} < c_{n-2}\). We distinguish now between the six possible pairs \((x, y)\) of \((Q.2)\) related to \(S_1\).

1. If \((x, y) = (0, 0)\) then \(c_{n-1} < c_{n-2} < c_0 < c_1\) which implies that 
\(\alpha_{n-2} = [3, 1, 2] = s_4, \, \alpha_{n-1} = [3, 2, 1] = s_6, \, g_{n-2} = 2, \text{ and } g_{n-1} = 2.\)

2. If \((x, y) = (1, 0)\) then \(c_{n-1} < c_0 < c_{n-2} < c_1\) which implies that 
\(\alpha_{n-2} = [1, 3, 2] = s_2, \, \alpha_{n-1} = [3, 2, 1] = s_6, \, g_{n-2} = 1, \text{ and } g_{n-1} = 2.\)

3. If \((x, y) = (1, 1)\) then \(c_0 < c_{n-1} < c_{n-2} < c_1\) which implies that 
\(\alpha_{n-2} = [1, 2, 3] = s_1, \, \alpha_{n-1} = [3, 1, 2] = s_4, \, g_{n-2} = 0, \text{ and } g_{n-1} = 2.\)

4. If \((x, y) = (2, 0)\) then \(c_{n-1} < c_0 < c_1 < c_{n-2}\) which implies that 
\(\alpha_{n-2} = [1, 3, 2] = s_2, \, \alpha_{n-1} = [3, 2, 1] = s_6, \, g_{n-2} = 1, \text{ and } g_{n-1} = 2.\)

5. If \((x, y) = (2, 1)\) then \(c_0 < c_{n-1} < c_1 < c_{n-2}\) which implies that 
\(\alpha_{n-2} = [1, 2, 3] = s_1, \, \alpha_{n-1} = [3, 1, 2] = s_4, \, g_{n-2} = 0, \text{ and } g_{n-1} = 2.\)

6. If \((x, y) = (2, 2)\) then \(c_0 < c_1 < c_{n-1} < c_{n-2}\) which implies that 
\(\alpha_{n-2} = [1, 2, 3] = s_1, \, \alpha_{n-1} = [1, 3, 2] = s_2, \, g_{n-2} = 0, \text{ and } g_{n-1} = 1.\)
Thus, the 4 possible pairs $(\alpha_{n-2}, \alpha_{n-1})$ and $(g_{n-2}, g_{n-1})$ are given in the following table:

<table>
<thead>
<tr>
<th>$(x, y)$</th>
<th>$(\alpha_{n-2}, \alpha_{n-1})$</th>
<th>$(g_{n-2}, g_{n-1})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0)</td>
<td>(4, 6)</td>
<td>(2, 2)</td>
</tr>
<tr>
<td>(1, 0), (2, 0)</td>
<td>(2, 6)</td>
<td>(1, 2)</td>
</tr>
<tr>
<td>(1, 1), (2, 1)</td>
<td>(1, 4)</td>
<td>(0, 2)</td>
</tr>
<tr>
<td>(2, 2)</td>
<td>(1, 2)</td>
<td>(0, 1)</td>
</tr>
</tbody>
</table>

Note, that some different pairs $(x, y)$ of property (Q.2) in state $P_{n-1}$ yield the same pair $(g_{n-2}, g_{n-1})$ for a given $\pi_1$. Note also that in Table 5.3, the sum of values in each row and in each column equals to 9 = 3².

Table 5.3: The number of possible pairs $(g_{n-2}, g_{n-1})$ which can complete a given prefix of a codeword $(g_0, \ldots, g_{n-3})$ to a legal codeword, correspond to a complete state $P_{n-1}$, given $\pi_1$, in the $(1,3,n)$-LRM scheme.

<table>
<thead>
<tr>
<th>$P_{n-1}$</th>
<th>$\pi_1$</th>
<th>[1, 2]</th>
<th>[2, 1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>$S_2$</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Theorem 14. If $M_3$ is the number of legal codewords in the $(1,3,n)$-LRM scheme then $\lim_{n \to \infty} \frac{M_3}{3^n} = 1$.

Proof. Consider a prefix of a codeword $g' = (g_0, g_1, \ldots, g_{n-4}, g_{n-3})$ which contains the subsequence $(g_{i-3}, g_{i-2}, g_{i-1}, g_i) = (2, 2, 0, 0)$. We will prove now that the charge levels $c_{i+1}$ and $c_{i+2}$ are independent of the charge levels $c_0$ and $c_1$.

1. $g_{i-3} = 2$ is equivalent to $c_{i-1} > \max\{c_{i-2}, c_{i-3}\}$, and
2. $g_{i-2} = 2$ is equivalent to $c_i > \max\{c_{i-1}, c_{i-2}\} = c_{i-1}$.

In other words, $c_i > c_{i-1} > c_{i-2}$ and $c_{i-1} > c_{i-3}$.

3. $g_{i-1} = 0$ is equivalent to $c_{i+1} < \min\{c_i, c_{i-1}\} = c_{i-1}$, and
4. $g_i = 0$ is equivalent to $c_{i+2} < \min\{c_{i+1}, c_i\} = c_{i+1}$.
Hence, the only constraints in the rankings between the charge levels in \( \{c_{i-3}, c_{i-2}, \ldots, c_{i+2}\} \) are given by:

1. \( c_{i-2} < c_{i-1} < c_i \),
2. \( c_{i-3} < c_{i-1} < c_i \),
3. \( c_{i+2} < c_{i+1} < c_{i-1} < c_i \).

Therefore, there are no constraints on the rankings between each one of the charge levels \( c_{i+1}, c_{i+2} \) and the charge levels \( c_{i-3}, c_{i-2} \). Thus, the charge levels \( c_{i+1}, c_{i+2} \) are independent of the charge levels \( c_0, c_1 \), which implies that \( P_{i+2} \) is a complete state. Furthermore, since \( c_{i+2} < c_{i+1} \), it follows that \( P_{i+2} = S_1 \) and by Table 5.2 we have that \( P_{n-1} \) must also be a complete state (\( S_1 \) or \( S_2 \)).

By using the well known Perron-Frobenius Theorem [1, 4], we can compute the asymptotic behavior of the number of sub-codewords of length \( n - 2 \) over the alphabet \( \{0, 1, 2\} \) which don’t include \( (2, 2, 0, 0) \) as a subsequence. First, an automata whose states accept all the sequences over \( \{0, 1, 2\} \) which don’t contain the subsequence \( (2, 2, 0, 0) \) is given, where the automata is in state \( q_i, 0 \leq i \leq 3 \), if the input sequence, before the current symbol of the sequence ends with the prefix of length \( i \) from \( (2, 2, 0, 0) \).

![Automata Diagram]

In the matrix which represents this automata (its state diagram), the value in cell \( (q_i, q_j) \) is the number of the labels on the outgoing edge from \( q_i \) to \( q_j \).

<table>
<thead>
<tr>
<th></th>
<th>( q_0 )</th>
<th>( q_1 )</th>
<th>( q_2 )</th>
<th>( q_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_0 )</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( q_1 )</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( q_3 )</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

101
The largest real eigenvalue of this matrix is 2.777. Therefore, by the Perron-Frobenius Theorem, we conclude that the number of sub-codewords of length $n - 2$ which don't contain $(2, 2, 0, 0)$ as a subsequence, tends to $2.777^n - 2$, when $n$ tends to $\infty$. Hence, the number of sub-codewords of length $n - 2$ which contain $(2, 2, 0, 0)$ as a subsequence, $M'$, tends to $3^n - 2 - 2.777^n - 2$. By Table 5.3 we can conclude that the number of legal codewords in the $(1, 3, n)$-LRM scheme is $M_3 \geq 9M'$, and therefore $\lim_{n \to \infty} \frac{M_3}{3^n} = 1$. 

Theorem 14 will be generalized for $t > 3$ in the next section. But, the last part of the proof which applies the Perron-Frobenius Theorem will be replaced by a different method (which can be used also for $t = 3$).

As written before, given a codeword $g$, we want to determine if $g$ is legal or not. First, we apply the decoding algorithm to obtain a base-word $\alpha$ which corresponds to $g$. If the decoding algorithm fails, then $g$ is an illegal codeword. However, the decoding algorithm might produce a cyclically non-realizable base-word $\alpha$. Note, that by the decoding algorithm, the dependencies between $\alpha_i$ and $\alpha_{i+1}$ are preserved for all $0 \leq i < n$. Thus, the only case in which $\alpha$ is non-realizable is related to the dependencies of the first two charge levels and the last two charge levels. These dependencies are implies by considering the consecutive permutations from $\alpha_0$, $\alpha_1$, $\alpha_2$, and so on up to $\alpha_{n-4}$ and $\alpha_{n-3}$. These dependencies can be inconsistent when we continue and consider the dependencies of the charge levels implied by the consecutive permutations $\alpha_{n-2}$ and $\alpha_{n-1}$, i.e. $\alpha$ is cyclically non-realizable base-word.

Thus, the question is how to indicate that a base-word is cyclically non-realizable. To answer this question, we use the states. Given a codeword $g = (g_0, g_1, \ldots, g_{n-1})$, and $\alpha_0$ (which is computed by the decoding procedure), we can determine $P_3$. Then, by Lemma 18, we can compute $P_{n-1}$. Note, that $\alpha_0$ determines the permutation of the first three cells and $P_{n-1}$ (see (Q.1)) determines the permutation of the last two elements. Additionally, $P_{n-1}$ (see (Q.2)) determines exactly all the possible ranking for the charge levels of each one of the last two cells among the first two cells. Thus, from $\alpha_0$ and $P_{n-1}$ we can conclude if $\alpha$ is cyclically realizable, i.e., if $g$ is legal. This complexity of this procedure is $O(n)$. 

102
5.A.3 The \((1, t, n)\)-LRM Scheme for \(t \geq 3\)

Some of the results in Section 5.A.2 can be generalized to the \((1, t, n)\)-LRM Scheme, \(t \geq 4\). The ideas will be described in details in this section, where the examples were given for \(t = 3\) in Section 5.A.2. Moreover, some of the proofs for \(t = 3\) were given in details and can be considered as examples for the general case \(t \geq 3\). Nevertheless, it should be emphasis that some of the generalizations are more complicated for \(t = 4\) and become impractical as \(t\) is increased.

In the \((1, t, n)\)-LRM scheme the size of each sliding window is \(t\). Therefore, to present the demodulated sequences of permutations the alphabet of the base-words has size \(t!\). The \(n\) charge levels form a sequence \(c = (c_0, c_1, \ldots, c_{n-1})\). Given \(t\) consecutive charge levels, \(c_i, c_{i+1}, \ldots, c_{i+t-1}\), where indices are taken modulo \(n\), the corresponding permutation \(\alpha_i\), from \(S_t\), is uniquely determined by the order of these \(t\) consecutive charge levels. Therefore, the \(n\) charge levels define a sequence of permutations \(\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})\). The position of the symbol \(t\) in the permutation \(\alpha_i\) determines the value of \(g_i\), i.e., \(g_i = j\), \(0 \leq j \leq t-1\), if \(t\) is in position \(t-j\) in the permutation. In other words, \(g_i\) is the ranking of \(c_{i+t-1}\) among \(c_i, c_{i+1}, \ldots, c_{i+t-1}\), i.e. \(g_i = 0\) if \(c_{i+t-1}\) is the lowest charge level, \(g_i = 1\) if only one charge level is below \(c_{i+t-1}\), and so on, where finally \(g_i = t-1\) if \(c_{i+t-1}\) is the highest charge level. The consecutive values \(g_0, g_1, \ldots, g_{n-1}\) define the codeword \(g = (g_0, g_1, \ldots, g_{n-1})\). This means that given the last \(t-1\) charge levels \(c_{i+1}, c_{i+2}, \ldots, c_{i+t-1}\), a new charge level \(c_{i+t}\) combined with these \(t-1\) charge levels, defines the permutation \(\alpha_{i+1}\) and the new symbol \(g_{i+1}\) in the codeword \(g\). Therefore, the base-word \(\alpha\) defined by the charge level vector \(c\), uniquely determines the related codeword \(g\). Clearly, given a permutation \(\alpha_{i-1}\), not all the \(t!\) permutations of \(S_t\) can follow \(\alpha_{i-1}\) to serve as \(\alpha_i\). Only \(t\) permutations can be used for \(\alpha_i\) based on \(\alpha_{i-1}\), including \(\alpha_0\) which follows \(\alpha_{n-1}\). Note that a base-word \(\alpha\) might not be realizable, even if it meets the dependencies between \(\alpha_{i-1}\) and \(\alpha_i\). This might happen if there is no possible sequence of charge levels that can be demodulated from \(\alpha\) due to the dependencies between the first \(t-1\) charge levels, and the last \(t-1\) charge levels. If \(\alpha_i\) can follow \(\alpha_{i-1}\) for all \(0 \leq i < n\), then the base-word \(\alpha\) can be mapped to a codeword \(g\), but \(g\) might be illegal since the base-word \(\alpha\) is not realizable by a sequence of charge levels. Given
the suggested mapping between the base-words and the codewords, we are mainly interested in three related questions concerning the legal codewords of the $(1, t, n)$-LRM scheme:

1. Given a legal codeword $g$ over the alphabet $\{0, 1, \ldots, t - 1\}$, present an efficient method to find the base-word $\alpha$ mapped to $g$.

2. Given a codeword $g$ over the alphabet $\{0, 1, \ldots, t - 1\}$, present an efficient method to decide whether $g$ is legal.

3. Find the number of legal codewords in the $(1, t, n)$-LRM scheme.

The rest of this section will be devoted to solve some of these questions.

To obtain the original base-word from the given codeword would be easy if for some $i$, $\alpha_i$ is given or known (in fact the permutation related to $t - 1$ consecutive cells is sufficient to figure out the entire base-word from a known codeword, either legal or illegal). If no such permutation is known then the task becomes more complicated and we have to analyse the codeword based only on the mapping from the base-words to the codewords.

Given a prefix of a codeword $(g_0, g_1, \ldots, g_{i-t+1})$, $2t - 3 \leq i < n$, obtained by the unknown charge levels $c_0, c_1, \ldots, c_i$, the ranking among the charge levels in the $j$-th cell, $c_j$, $i - t + 2 \leq j \leq i$, and the first $t - 1$ cells, $c_0, c_1, \ldots, c_{i-2}$, might have a few options (at most $t$). These options will be denoted by 0, 1, up to $t - 1$, where 0 represents that $c_i$ is lower than $c_0, c_1, \ldots, c_{i-2}$, 1 represents that $c_i$ is higher than exactly one of them, and so on. For each $i$, $2t - 3 \leq i < n$, consider the following two properties regarding $c_{i-t+2}, c_{i-t+3}, \ldots, c_i$:

(Q.1) the permutation $\pi_i$ induced by $c_{i-t+2}, c_{i-t+3}, \ldots, c_i$ ($(t - 1)!$ possible permutations).

(Q.2) the set of all possible $(t - 1)$-tuples of rankings of the charge level $c_j$, for each $j$, $i - t + 2 \leq j \leq i$, among the charge levels $c_0, c_1, \ldots, c_{i-2}$.

The elements of the set defined in (Q.2) will be denoted by $(t - 1)$-tuples $(x_{i-t+2}, x_{i-t+3}, \ldots, x_i)$, $x_j \in \{0, 1, \ldots, t - 1\}$, $i - t + 2 \leq j \leq i$, where $x_j$ represents the ranking of the charge level $c_j$ among the charge levels $c_0, c_1, \ldots, c_{i-2}$. Note, that not all the $t^{t-1}$ possible $(t - 1)$-tuples can be obtained for a given permutation defined by (Q.1). We call a pair defined by
the permutation of (Q.1) and the set of \((t-1)\)-tuples defined by (Q.2) a state. The state at index \(i\) (for \(c_{i-t+2}, c_{i-t+3}, \ldots, c_i\)) will be denoted by \(P_i\). Let’s denote by \(\pi_i\), \(t-2 \leq i < n\), the permutation defined by \(c_{i-t+2}, \ldots, c_{i-1}, c_i\).

**Lemma 19.** A maximum of \(\binom{2t}{t-1}\) possible \((t-1)\)-tuples can be obtained in (Q.2) for a given state.

**Proof.** A state \(P_i\) is first identified by the permutation \(\pi_i\) defined by \(c_{i-t+2}, c_{i-t+3}, \ldots, c_i\) (see (Q.1)). The highest rank among \(c_{i-t+2}, c_{i-t+3}, \ldots, c_i\) can be ranked in \(t\) different ways among \(c_0, c_1, \ldots, c_{t-2}\). If it has rank \(\ell\) then the next highest rank among \(c_{i-t+2}, c_{i-t+3}, \ldots, c_i\) can be ranked in \(\ell\) different ways among \(c_0, c_1, \ldots, c_{t-2}\). If it has rank \(m\) then the next highest rank among \(c_{i-t+2}, c_{i-t+3}, \ldots, c_i\) can be ranked in \(m\) different ways among \(c_0, c_1, \ldots, c_{t-2}\). We continue in the same manner, and hence the maximum number of possibilities in (Q.2) is exactly the number of possible \((t-1)\)-tuples \(b_1, b_2, \ldots, b_{t-1}\) over \(\{1, 2, \ldots, t\}\) such that \(b_{j+1} \leq b_j\), \(1 \leq j \leq t-2\). The number of such \((t-1)\)-tuples is \(\binom{2t}{t-1}\). \(\Box\)

Recall that if the ranking between the charge levels \(c_{i-t+1}, c_{i-t+2}, \ldots, c_i\) is known then \(g_{i-t+1}\) can be computed based on the ranking of the charge level \(c_i\) among the \(t-1\) preceding charge levels \(c_{i-t+1}, c_{i-t+2}, \ldots, c_{i-1}\). Recall also that the state \(P_i\) is defined by two properties (Q.1) and (Q.2). By (Q.1) we know the ranking between the charge levels \(c_{i-t+2}, c_{i-t+3}, \ldots, c_i\) and by (Q.2) we know the ranking of each one of these last \(t-1\) charge levels among the first \(t-1\) charge levels. The state \(P_i\) is now determined based on these two properties.

**Lemma 20.** If \(P_i\) and \(g_{i-t+2}\), for some \(2t-3 \leq i < n-1\), are given, then \(P_{i+1}\) is uniquely determined.

**Proof.** \(P_i\) is characterized by the permutation \(\pi_i\) in (Q.1) and the \((t-1)\)-tuples in (Q.2). The permutation \(\pi_i\) is defined by the sequence of charge levels \(c' = (c_{i-t+2}, c_{i-t+3}, \ldots, c_i)\) and \(g_{i-t+2}\) defines the ranking of \(c_{i+1}\) among the set of charge levels in \(c'\). Hence, the permutation defined by \(c'' = (c_{i-t+3}, c_{i-t+4}, \ldots, c_{i+1})\) is uniquely determined and property (Q.1) for \(P_{i+1}\) is well defined.

Let \(y = (y_0, y_1, \ldots, y_{t-2})\) be a possible \((t-1)\)-tuple in (Q.2) of \(P_{i+1}\), that is, \(y\) represents a possible ranking of each charge level in \(c''\) among
\((c_0, c_1, \ldots, c_{t-2})\), where \(y_j, 0 \leq j \leq t-2\), represents the ranking of \(c_{i-t+3+j}\) among the charge levels of the first \(t-1\) cells. Then, there exists a possible \((t-1)\)-tuple in (Q.2) of \(P_i\), \(x = (x_0, x_1, \ldots, x_{t-2})\), where \(x_j = y_{j-1}\), \(0 < j \leq t-2\), since \(x_j\) represents a ranking of \(c_{i-t+2+j}\) among the charge levels of the first \(t-1\) cells in one possible \((t-1)\)-tuple in (Q.2) of \(P_i\).

Thus, to complete the proof, it is sufficient to show, that \(\pi_i, \pi_{i+1}\), and \(x\), determine all possibilities for \(y_{t-2}\). Recall, that \(y_{t-2}\) relates to the ranking possibilities of \(c_{i+1}\) among the first \(t-1\) charge levels.

The permutation \(\pi_{i+1}\) determines the ranking of \(c_{i+1}\) among \(c' = (c_{i-t+3}, c_{i-t+4}, \ldots, c_{i+1})\). Denote by \(c_j\) and \(c_{j2}\) the two charge levels in \(c'\) which are adjacent to \(c_{i+1}\) in their size, where \(c_{j1} < c_i < c_{j2}\), \(\text{(that is,}\ j_1 - (i + 1) + t - 1 \text{and } j_2 - (i + 1) + t - 1 \text{are adjacent to } t - 1 \text{in } \pi_{i+1}.)\)

Note that if \(t - 1\) is the first or the last symbol in \(\pi_{i+1}\) then only one of \(j_1, j_2\) exists. Then, \(x_{j_1-i+t-2}\) and \(x_{j_2-i+t-2}\) represent a possible ranking of \(c_{j1}\) and \(c_{j2}\) among the charge levels of the first \(t - 1\) cells. These possible rankings of \(c_{j1}\) and \(c_{j2}\), with the only constraint on \(c_{i+1}\) to be between \(c_{j1}\) and \(c_{j2}\), determine the possible rankings of \(c_{i+1}\) among the first \(t - 1\) cells, and therefore determine the possible values for \(y_{t-2}\).

\[\textbf{Corollary 2.} \text{ If } P_i = P_j \text{ for some } 2t-3 \leq i < j < n-1 \text{ and } g_{i-t+2} = g_{j-t+2} \text{ then } P_{i+1} = P_{j+1}.\]

A state which has all \(\binom{2t}{t-1}\) possible \((t-1)\)-tuples in property (Q.2) will be called a \textit{complete state}. In other words, \(P_i\) is a complete state if each one of the ranks \(c_{i-t+2}, c_{i-t+3}, \ldots, c_i\) is independent of the ranks \(c_0, c_1, \ldots, c_{t-2}\).

An immediate consequence is that

\[\textbf{Lemma 21.} \text{ If } P_i \text{ is a complete state then also } P_{i+1} \text{ is a complete state.}\]

\[\textbf{Lemma 22.} \text{ In the } (1, t, n)\text{-LRM there are } (t-1)! \text{ complete states.}\]

\[\text{Proof.} \text{ Each permutation on the ranks } c_{i-t+2}, c_{i-t+3}, \ldots, c_t \text{ induces } \binom{2t}{t-1} \text{ possible rankings among } c_0, c_1, \ldots, c_{t-2} \text{ as explained in the proof of Lemma 19.}\]

We are only interested in complete states since non-complete states might lead to a relatively small number of legal codewords. The non-complete states and their related codewords will be omitted in the computations of the number of legal codewords which follows.
Given \( \pi_{t-2} \), the permutation defined by the first \( t - 1 \) charge levels, and \( g' = (g_0, g_1, \ldots, g_{n-t}) \), the sub-base-word \((\alpha_0, \alpha_1, \ldots, \alpha_{n-t})\), of a realizable base-word which corresponds to \( \pi_{t-2} \) and \( g' \), is determined unambiguously. But, there are a few possible assignments for \( \alpha_{n-t+1}, \alpha_{n-t+2}, \ldots, \alpha_{n-1} \), which correspond to possible assignments for \( g_{n-t+1}, g_{n-t+2}, \ldots, g_{n-1} \). These assignments are determined by the state \( P_{n-1} \) and the permutation \( \pi_{t-2} \). Each assignment provides a distinct realizable base-word which is represented by the state \( P_{n-1} \) and the permutation \( \pi_{t-2} \).

Recall, that only complete states will be considered in the computations. We generate a table \( \mathcal{G} \) to with \((t-1)! \) rows indexed by the number of complete states and \((t - 1)! \) columns indexed by the possible assignments of \( \pi_{t-2} \). In entry \( \mathcal{G}(i,j), 1 \leq i, j \leq (t-1)! \), we have the number of realizable base-words that can be obtained from \( P_{n-1} \) which is the \( i \)-th complete state when \( \pi_{t-2} \) is the \( j \)-th permutation of \( S_{t-1} \).

**Lemma 23.** The sum of elements in the \( i \)-th row of \( \mathcal{G} \) is \( t^{t-1} \).

**Proof.** Let \( P_{n-1} \) be the \( i \)-th complete state. If there are no constraints than clearly the possible assignments for \( g_{n-t+1}, g_{n-t+2}, \ldots, g_{n-1} \) is at most \( t^{t-1} \) since there are \( t \) distinct assignments for each \( g_j \). Given the permutation \( \pi_{n-1} \), any assignment to \( g_{n-t+1}, g_{n-t+2}, \ldots, g_{n-1} \) yields a unique permutation \( \pi \) of the first \( t - 1 \) charge levels. This assignment is feasible since \( c_{n-t+1}, c_{n-t+2}, \ldots, c_{n-1} \) and \( c_0, c_1, \ldots, c_{t-2} \) are independent when \( P_{n-1} \) is a complete state. For this assignment we have \( \pi_{t-2} = \pi \) (assume now that \( \pi \) is the \( j \)-th permutation of \( S_{t-1} \)) and a contribution of one to \( \mathcal{G}(i,j) \) and to the \( i \)-th (which corresponds to \( \pi_{n-1} \)) row of \( \mathcal{G} \). Thus, we have total contributions of \( t^{t-1} \) to the \( i \)-th row of \( \mathcal{G} \) and the lemma follows. \( \square \)

**Theorem 15.** If \( M_t \) is the number of legal codewords in the \((1, t, n)\)-LRM scheme then \( \lim_{n \to \infty} \frac{M_t}{n^{t-1}} = 1 \).

**Proof.** Consider a prefix of a codeword \( g' = (g_0, g_1, \ldots, g_{n-t}) \) which contains the subsequence of length \( 2(t-1) \), \( \beta = (g_{i-2t+3}, g_{i-2t+4}, \ldots, g_{i-1}, g_i) = (t-1 \text{ times}, t-1 \text{ times}) \). We will prove now that the charge levels \( c_{i+1}, c_{i+2}, \ldots, c_{i+t-1} \) are independent of the charge levels \( c_0, c_1, \ldots, c_{t-2} \).
This implies that $P_{i+t-1}$ is one of the $(t-1)!$ complete state in the $(1, t, n)$-LRM scheme. For this, we have to prove that each one of the charge levels $c_{i+1}, c_{i+2}, \ldots, c_{i+t-1}$ can be lower than $c_0, c_1, \ldots, c_{t-2}$, between them $(t-2)$ distinct possible options), or higher than all of them.

The subsequence $\beta$ starts with $g_{i-2t+3} = t - 1$ which imposes that $c_{i+t+2}$ is higher than the $t - 1$ charge levels before it, and implies that $c_{i+t+2}$ might be higher than the first $t - 1$ charge levels. The sub-codeword $(g_{i-2t+4}, \ldots, g_{i-t+1}) = (t-1, t-1, \ldots, t-1)$ imposes that $c_{i-t+2} < c_{i-t+3} < \cdots < c_i$. Furthermore, $g_{i-t+2} = 0$ implies $c_{i+1} < c_{i-t+2}$, and $(g_{i-t+3}, \ldots, g_t) = (0, 0, \ldots, 0)$ imposes the constraint $c_{i+1} > c_{i+2} > c_{i+3} > \cdots > c_{i+t-1}$. Hence, we have that $c_{i+t-1} < c_{i+t-2} < \cdots < c_{i+1} < c_{i-t+2} < c_{i-t+3} < \cdots < c_i$, where $c_{i+t+2}$ might be higher than the first $t - 1$ charge levels, and there does not exist any additional constraint on the possible values of the charge levels $c_{i+1}, c_{i+2}, \ldots, c_{i+t-1}$ regarding their ranking among the previous charge levels. This implies that there is no dependency between the charge levels $c_{i+1}, c_{i+2}, \ldots, c_{i+t-1}$, and the first $t - 1$ charge levels, i.e., $P_{i+t-1}$ is a complete state. Moreover, the permutation $\pi_{i+t-1}$ of the property (Q.1) for this state is $[1, 2, \ldots, t-1]$. As a consequence Lemma 21 implies that also $P_{n-1}$ is a complete state.

Let $B$ be the set of sequences of length $n - t + 1$, over the alphabet $\{0, 1, \ldots, t-1\}$, which include $\beta$ as a subsequence. We partition each sequence of length $2\ell(t-1)$ into $\ell$ consecutive disjoint segments of length $2(t-1)$ and one segment (the last) of length $n-t+1-2\ell(t-1)$. Let $A$ be a subset of $B$ which consists of the sequences which contain a segment equals to $\beta$. Let $A^c$ be the complimentary set of $A$, i.e. the set of all sequences of length $n-t+1$, over the alphabet $\{0, 1, \ldots, t-1\}$, which do not contain a segment equals to $\beta$. Now, we have

\[ |B| \geq |A| = t^{n-t+1} - |A^c| = t^{n-t+1} - (2^{t(t-1)} - 1)\ell t^{n-t+1-2\ell(t-1)}. \]

By Lemma 23 we have that $M_t \geq t^{t-1}|B|$ and therefore

\[ \lim_{n \to \infty} \frac{M_t}{t^n} \geq \lim_{n \to \infty} \frac{t^{n-t+1} - (2^{t(t-1)} - 1)\ell t^{n-t+1-2\ell(t-1)}}{t^n} = 1. \]

\[ \square \]

108
5.A.4 Conclusions and Open Problems

In this paper, encoding, decoding, and enumeration of the \((1, t, n)\)-LRM scheme are studied. A complete solution is given for the \((1, 3, n)\)-LRM scheme. A simple encoding for the \((1, t, n)\)-LRM scheme for any \(t \geq 3\) is presented. For the \((1, 3, n)\)-LRM scheme a related decoding was presented. We also proved that if \(M_t\) is the number of legal codewords in the \((1, t, n)\)-LRM scheme then \(\lim_{n \to \infty} \frac{M_t}{t^n} = 1\). We conclude with two problems for future research raised in our discussion.

1. Find an efficient algorithm to determine if a given codeword in the \((1, t, n)\)-LRM scheme, for \(t \geq 4\), is legal or not.

2. Prove that the encoding algorithm for the \((1, t, n)\)-LRM scheme, \(t \geq 4\), induces a bijection between the realizable base-words and the legal codewords.

3. Find an efficient decoding algorithm for the \((1, t, n)\)-LRM scheme, \(t \geq 4\).

Acknowledgment

The authors would like to thank Ronny Roth for helpful discussions.
Bibliography


5.B Conference Version

abstract

Local rank modulation scheme was suggested recently for representing information in flash memories in order to overcome drawbacks of rank modulation. For $0 < s \leq t \leq n$ with $s$ divides $n$, an $(s, t, n)$-LRM scheme is a local rank modulation scheme where the $n$ cells are locally viewed cyclically through a sliding window of size $t$ resulting in a sequence of small permutations which requires less comparisons and less distinct values. The gap between two such windows equals to $s$. In this work, encoding, decoding, and asymptotic enumeration of the $(1, 3, n)$-LRM scheme is studied. The techniques which are suggested have some generalizations for $(1, t, n)$-LRM, $t > 3$, but the proofs will become more complicated. The enumeration problem is presented also as a purely combinatorial problem. Finally, we prove the conjecture that the size of a constant weight $(1, 2, n)$-LRM Gray code with weight two is at most $2^n$.

5.B.1 Introduction

Flash memory is a non-volatile technology that is both electrically programmable and electrically erasable. It incorporates a set of cells maintained at a set of levels of charge to encode information. While raising the charge level of a cell is an easy operation, reducing the charge level requires the erasure of the whole block to which the cell belongs. For this reason charge is injected into the cell over several iterations. Such programming is slow and can cause errors since cells may be injected with extra unwanted charge. Other common errors in flash memory cells are due to charge leakage and reading disturbance that may cause charge to move from one cell to its adjacent cells. In order to overcome these problems, the novel framework of rank modulation was introduced in [3]. In this setup, the information is carried by the relative ranking of the cells’ charge levels and not by the absolute values of the charge levels. Denote the charge level in the $i$th cell by $c_i$, $0 \leq i < n$, and $c = (c_0, c_2, \ldots, c_{n-1})$ is the sequence of the charges in $n$ cells. A codeword in this scheme is the permutation defined by the order of the charge levels, from the highest one to the lowest one, e.g. if $n = 5$ and
Then the permutation, i.e., the codeword in the rank modulation scheme, is \([5, 4, 2, 1, 3]\). This allows for more efficient programming of cells, and coding by the ranking of the cells’ charge levels is more robust to charge leakage than coding by their actual values. The push-to-the-top operation is a basic minimal cost operation in the rank modulation scheme by which a single cell has its charge level increased such that it will be the highest of the set.

A drawback of the rank modulation scheme is the need for a large number of comparisons when reading the induced permutation. Furthermore, distinct \(n\) charge levels are required for a group of \(n\) cells. The local rank modulation scheme was suggested in order to overcome these problems. In this scheme, the \(n\) cells are locally viewed through a sliding window, resulting in a sequence of permutations for a much smaller number of cells which requires less comparisons and less distinct values. For \(0 < s \leq t \leq n\), where \(s\) divides \(n\), the \((s, t, n)\)-LRM scheme, defined in [2, 5], is a local rank modulation scheme over \(n\) physical cells, where \(t\) is the size of each sliding window and \(s\) is the gap between two such windows. In this scheme the permutations are over \(\{1, 2, \ldots, t\}\), i.e., form \(S_t\), and the push-to-the-top operation merely raises the charge level of the selected cell above those cells which are comparable with it. We say a sequence with \(\frac{t!}{s}\) permutations from \(S_t\) is an \((s, t, n)\)-LRM scheme realizable if it can be demodulated to a sequence of charges in \(n\) cells under the \((s, t, n)\)-LRM scheme. Except for the degenerate case where \(s = t = n\), not every sequence is realizable.

The \((1, 2, n)\)-LRM scheme was defined in [2] in order to get the simplest hardware implementation. The demodulated sequences of permutations in this scheme contain all the binary words except two, the all-ones and all-zeros sequences. Therefore, the number of codewords in this scheme is \(2^n - 2\).

In this paper we focus on the \((1, t, n)\)-LRM schemes for \(t \geq 3\), and suggest a demodulation method for these schemes. The \((1, t, n)\)-LRM scheme is a local rank modulation scheme over \(n\) physical cells, where the size of each sliding window is \(t\), and each cell starts a new window. Since the size of a sliding window is \(t\), demodulated sequences of permutations in this scheme contain \(t!\) permutations. Therefore, we need \(t!\) symbols to present the demodulated sequences of permutations.

Let \(s = (s_1, s_2, \ldots, s_{t!})\) be an order of the \(t!\) permutations from \(S_t\), and \(\Sigma = \{1, 2, \ldots, t!\}\) be an alphabet where \(i\) represents the permutation \(s_i\). A
sequence $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$ over the alphabet $\Sigma$ is called a base-word in the $(1, t, n)$-LRM scheme, and it is realizable, if there exists a sequence of charges $c = (c_0, c_1, \ldots, c_{n-1})$, such that for each $i$, $0 \leq i \leq n - 1$, $\alpha_i$ represent the permutation induced by $c_i, c_{i+1}, \ldots, c_{i+t-1}$, where indices are taken modulo $n$. The indices in the base-words and codewords are also taken modulo $n$ as in the charge levels.

In this paper we produce a mapping method, in which each $\alpha$, a base-word over the alphabet of size $t!$, is mapped to a codeword $g = (g_0, g_1, \ldots, g_{n-1})$ over an alphabet of size $t$. A codeword is called legal if there exists a realizable base-word which is mapped to it. We have to make sure that two distinct realizable base-words are mapped into two distinct legal codewords.

Let $M_t$ be the number of legal codewords in the $(1, t, n)$-LRM scheme. Clearly, $M_t \leq t^n$, but this upper bound is not tight since there exist illegal codewords. We conjecture that $\lim_{n \to \infty} \frac{M_t}{t^n} = 1$ and prove this conjecture for $t = 3$ and $t = 4$.

The rest of this paper is organized as follows. The encoding, decoding and asymptotic enumeration of $(1, 3, n)$-LRM scheme is presented in Section 5.B.2. Generalizations, especially for the enumeration technique for the $(1, t, n)$-LRM scheme, $t > 3$, is given in Section 5.B.3. The generalization of the asymptotic enumeration problem is presented as a combinatorial problem. The solution for the $(1, 4, n)$-LRM scheme is also given. In Section 5.B.4 we discuss constant weight $(1, 2, n)$-LRM Gray codes with weight two. We claim that conjecture from [2] that the size of such code is at most $2n$ is true. In Section 5.B.5 conclusion and problems for future research are presented.

5.B.2 The $(1, 3, n)$-LRM scheme

In the $(1, 3, n)$-LRM scheme the size of each sliding window is 3. Therefore, an alphabet of size $3!$ is required to present the demodulated sequences of permutations.

$$s_1 = [1, 2, 3] \quad s_2 = [1, 3, 2]$$
$$s_3 = [2, 1, 3] \quad s_4 = [3, 1, 2]
$$
$$s_5 = [2, 3, 1] \quad s_6 = [3, 2, 1]$$
The alphabet of the base-words is \( \Sigma = \{1, 2, \ldots, 6\} \), where the symbol \( \ell \) represents the permutation \( s_\ell \). Let \( \alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) \) be a base-word. Note that the last two cells which determine \( \alpha_i \) (\( 0 \leq i \leq n - 1 \)) are the first two cells which determine \( \alpha_{i+1} \), i.e., the permutation related to \( \alpha_{i+1} \) is obtained from \( \alpha_i \) by the following way. The symbol 1 in the permutation related to \( \alpha_i \) is omitted, the symbols 2, 3 in the permutation are replaced with 1, 2, respectively, and a new symbol 3 is inserted before 1, 2, between them, or after both of them. Therefore, given \( \alpha_i \), there are exactly 3 options for \( \alpha_{i+1} \).

Let \( \Sigma^1 = \{1, 3, 5\} \) and \( \Sigma^2 = \{2, 4, 6\} \) be a partition of \( \Sigma \) into the even and the odd symbols, respectively. Note that for each \( \Sigma^i, i \in \{1, 2\} \), the permutations related to the symbols in \( \Sigma^i \) agree on the order of cells 2 and 3. Therefore, they also agree on the 3 possibilities of their succeeding permutation. Denote the set of symbols of these succeeding permutations by \( \tilde{\Sigma}^i \). Thus, we have \( \tilde{\Sigma}^1 = \{1, 2, 4\} \) and \( \tilde{\Sigma}^2 = \{3, 5, 6\} \).

The base-word \( \alpha \) is mapped to a codeword \( g = (g_0, g_1, \ldots, g_{n-1}) \) over the alphabet \( \{0, 1, 2\} \). The relations between \( \alpha_{i-1}, \alpha_i, \) and \( g_i \), where \( 0 \leq i \leq n - 1 \), are presented in Table 5.4. This table induces a mapping from the realizable base-words to the codewords. As mentioned before, given \( \alpha_{i-1} \), there are three options for \( \alpha_i \). In all these options the sub-permutation of \( \{1, 2\} \) is the same, and the difference is the index of symbol 3 in the permutation related to \( \alpha_i \). Thus, \( g_i \) represents the index of symbol 3 in this permutation and it equal to the number of symbols which are to the right of the symbol 3 in the permutation related to \( \alpha_i \). In other words, \( g_i \) represents the relation between \( c_{i+2} \), the charge level in cell \( i + 2 \), and the charge levels in two cells which proceed it, i.e., \( c_i \) and \( c_{i+1} \).

| \( \alpha_{i-1} \in \Sigma^1 \) | \( \alpha_i = 1 \) | \( \alpha_i = 2 \) | \( \alpha_i = 4 \) |
| \( \alpha_{i-1} \in \Sigma^2 \) | \( \alpha_i = 3 \) | \( \alpha_i = 5 \) | \( \alpha_i = 6 \) |
| \( g_i = 0 \) | \( g_i = 1 \) | \( g_i = 2 \) |

Table 5.4: The encoding key of the \((1, 3, n)\)-LRM scheme

Note that there might exist non-realizable base-words which are mapped to codewords by this method. A base-word \( \alpha \), which can be mapped to a codeword in this method, must satisfy only the dependencies between \( \alpha_i \) and \( \alpha_{i+1} \) (\( 0 \leq i \leq n - 1 \)), but it still can be non-realizable. The \( n \) cells are viewed
cyclically, i.e., the charge of the last cell, $c_{n-1}$ is compared with the charge in the first two cells, $c_0$ and $c_1$, and the same works for the three charge levels $c_{n-2}$, $c_{n-1}$, and $c_0$. Therefore, there might exists a non-realizable dependency between the charge levels in the last two cells and the charge levels in the first two cells. Such a non-realizable base-word will be called a *cyclic non-realizable* base-word. For example, the following base-words are cyclic non-realizable.

- $1^n$ - the charge levels are always decreased.
- $6^n$ - the charge levels are always increased.
- $(2,5)^n/2$, where $n$ is even - the charge level of each cell is between the charge levels of the two cells which proceed it.

Recall, that a codeword is called legal if there exists a realizable base-word which is mapped to it.

**Theorem 16.** Table 5.4 provides an one-to-one mapping between the realizable base-words and the legal codewords.

**Proof.** Obviously, each base-word is mapped to exactly one codeword. Now, we prove that the other direction is also true. Clearly, $1^n$ is an illegal codeword as the charge level of each cell should be between the charge levels of the two cells which proceed it. Thus, given a legal codeword $g = (g_0, g_1, \ldots, g_{n-1})$, there exists $0 \leq i \leq n-1$, such that $g_i \in \{0,2\}$. If $g_i = 0$ then we have $\alpha_i \in \{1,3\}$, i.e., $\alpha_i$ is odd. Thus, $\alpha_{i+1}$ is determined by an entry in the first row in Table 5.4, where the column is chosen by the value of $g_{i+1}$. If $g_i = 2$ then we have $\alpha_i \in \{4,6\}$, i.e., $\alpha_i$ is even. Thus, $\alpha_{i+1}$ is determined by an entry in the second row in Table 5.4, where the column is chosen by the value of $g_{i+1}$. Now, it is easy to determine $\alpha_{i+2}, \alpha_{i+3}, \ldots, \alpha_{i+n-1}, \alpha_{i+n} = \alpha_i$ one after one in this cyclic order. Note that if $\alpha_i$ is not equal to an optional initial value (from the set $\{1,3\}$ if $g_i = 0$ and from $\{4,6\}$ if $g_i = 2$) then we can conclude that $g$ is illegal. 

Decoding a given codeword to a base-word doesn’t guarantee that the codeword is legal, because the accepted base-word may be cyclic non-realizable. For example, the cyclic non-realizable base-word $\alpha = 1^n$ is mapped to the illegal codeword $g = 0^n$. Given such a codeword, it would be interesting to decide efficiently if it is a legal codeword or not.
Next, the main theorem for the \((1, 3, n)\)-LRM scheme is given.

**Theorem 17.** If \(M_3\) is the number of legal codewords in the \((1, 3, n)\)-LRM scheme then \(\lim_{n \to \infty} \frac{M_3}{n} = 1\).

**Proof.** Note that \(g_i\) is determined by \(c_i, c_{i+1},\) and \(c_{i+2}\). \(g_i = 0\) if \(c_{i+2}\) is lower than \(c_i\) and \(c_{i+1}\); \(g_i = 1\) if \(c_{i+2}\) is between \(c_i\) and \(c_{i+1}\); and \(g_i = 2\) if \(c_{i+2}\) is higher than \(c_i\) and \(c_{i+1}\).

Given a sub-codeword \((g_0, g_1, \ldots, g_{i-2})\) obtained by the charge levels \(c_0, c_1, \ldots, c_i\), the relation between the charge levels in the \(i\)th cell, \(c_i\), and the first two cells, \(c_0\) and \(c_1\), might have a few options. These options will be denoted by 0, 1 and 2, where 0 represents that \(c_i\) is lower than \(c_0\) and \(c_1\), 1 represents that \(c_i\) is between them, and 2 represents that \(c_i\) is higher than both of them. For each \(i, 3 \leq i < n\), we provide two properties regarding the charge levels \(c_{i-1}\) and \(c_i\):

(Q.1) the permutation induced by \(c_{i-1}\) and \(c_i\) ([1, 2] or [2, 1]);

(Q.2) the set of all possible pairs of the relations between the charge levels \(c_{i-1}\) and \(c_i\) and the charge levels \(c_0\) and \(c_1\).

The elements of the set defined in (Q.2) will be denoted by pairs \((x, y)\), \(x, y \in \{0, 1, 2\}\), where \(x\) represents the relation between \(c_{i-1}\) and the first two cells, and \(y\) represents the relation between \(c_i\) and the first two cells. Note, that not all the nine pairs \((x, y)\) can be obtained for a given permutation defined by (Q.1).

We call each set of properties defined by (Q.1) and (Q.2) a state, and the state at index \(i\) will be denoted by \(P_i\). Given a sub-codeword \(g' = (g_0, g_1, \ldots, g_{n-4}, g_{n-3})\), the states in the sequence \((P_3, P_4, \ldots, P_{n-1})\) are determined one after one in this order. It is easy to verify that if \(P_i = P_j\) for some \(3 \leq i < j < n - 1\) then \(P_{i+1} = P_{j+1}\). A state which has the all possibilities in the second property will be called a complete state. It is easy to verify that in the \((1, 3, n)\)-LRM there are two complete states:

1) state 1 : [1, 2], \{(0, 0), (1, 1), (1, 0), (2, 2), (2, 1), (2, 0)\}.

2) state 2 : [2, 1], \{(0, 0), (0, 1), (0, 2), (1, 1), (1, 2), (2, 2)\}.

We are only interested in complete states because states which are not complete will lead to a relatively small number of legal codewords. These codewords will be omitted in our computations. Given \(g_{i-1}\), the succeeding
state $P_{i+1}$ of a state $P_i$ which is a complete state, is given in Table 5.5. It is clear that from the table that if $P_i$ is a complete state then also $P_{i+1}$ is a complete state.

<table>
<thead>
<tr>
<th>$g_{i-1}$</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>state 1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>state 2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 5.5: Succeeding states in the (1, 3, n)-LRM scheme

Denote by $\pi$ the permutation defined by the charge levels in the first two cells. Given $\pi$ and $g' = (g_0, g_1, \ldots, g_{n-4}, g_{n-3})$, the sub-base-word $(\alpha_0, \alpha_1, \ldots, \alpha_{n-3})$ of a realizable base-word which corresponds to $\pi$ and $g'$ is determined unambiguously. But, $\alpha_{n-2}, \alpha_{n-1}$ might have a few options. These options are determined by the state $P_{n-1}$ and the permutation $\pi$. Each option provides a distinct base-word which is represented by the state $P_{n-1}$ and the permutation $\pi$. For example the permutation $\pi = [2, 1]$ and the sub-codeword $g' = 2^{n-2}$ imply that the charge levels are always increased, where $c_0$ is the lowest, and $c_{n-1}$ is the highest. Therefore, the only base-word it represents is $(6, 6, \ldots, 6, 3, 2)$, where $P_{n-1} = ([2, 1], \{(2, 2)\})$.

Recall, that we are only interested in complete states in our computations. In Table 5.6 we enumerate the number of base-words represented by $\pi$ and $P_{n-1}$ for state 1 or state 2. Given $g'$, $c_0$ is compared with $c_{n-2}$ and $c_{n-1}$ to obtain $g_{n-2}$; and $c_1$ is compared with $c_{n-1}$ and $c_0$ to obtain $g_{n-1}$. Note, that some different pairs $(x, y)$ of a given state result in the same pair $(g_{n-2}, g_{n-1})$ for a given $\pi$. Note also that in Table 5.6, the sum of values in each row and in each column equals to $9 = 3^2$.

<table>
<thead>
<tr>
<th>$P_{n-1}$</th>
<th>$\pi$</th>
<th>[1, 2]</th>
<th>[2, 1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>state 1</td>
<td></td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>state 2</td>
<td></td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 5.6: The number of base-words represented by the complete states in the (1, 3, n)-LRM scheme
If a sub-codeword $g' = (g_0, g_1, \ldots, g_{n-4}, g_{n-3})$ contains the sequence $(2, 0, 1, 1)$ as a subsequence at indices $(i-3, i-2, i-1, i)$, then this sequence ends with state 1, i.e. $P_{i+2}$ is state 1. The reason is that in this case $c_{i+1}$ and $c_{i+2}$ have no dependency on the charge levels of $c_{i-3}$ or $c_{i-2}$, i.e., each one of $c_{i+1}$ and $c_{i+2}$ can be lower, between, or higher than $c_{i-3}$ and $c_{i-2}$. Therefore, the relation between $c_{i+1}$ and $c_{i+2}$, and the first two cells, $c_0$ and $c_1$, has all the possibilities, i.e., $P_{i+2}$ is a complete state. It is easy to verify that $c_{i+2}$ is lower than $c_{i+1}$, and thus $P_{i+2}$ is state 1. By Table 5.5 we have that $P_{n-1}$ of this sequence must be a complete state (state 1 or state 2).

By using the well known Perron-Frobenius Theorem [1, 4], we can compute the behavior of the number of sequences of length $n - 2$ over the alphabet $\{0, 1, 2\}$ which don’t include $(2, 0, 1, 1)$ as a subsequence. First, an automata whose states accept all the sequences over $\{0, 1, 2\}$ which don’t contain the subsequence $(2, 0, 1, 1)$ is given.

\[
\begin{array}{ccccc}
q_1 & q_2 & q_3 & q_4 \\
0 & 2 & 2 & 1 \\
0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 \\
\end{array}
\]

The largest real eigenvalue of this matrix is $2.9615$. Therefore, by the Perron-Frobenius Theorem, we can conclude that the number of sub-codewords of length $n - 2$ which don’t contain $(2, 0, 1, 1)$ as a subsequence, tends to be $2.9615^{n-2}$, where $n$ tends to $\infty$. Let $M'$ be the number of sub-codewords of length $n - 2$ which contain $(2, 0, 1, 1)$ as a subsequence. By Table 5.6 we can

118
conclude that the number of legal codewords in the \((1, 3, n)\)-LRM scheme is

\[ M_3 \geq 9M' \]

and therefore

\[ \lim_{n \to \infty} \frac{M_3}{3^n} = 1. \]

\[ \square \]

5.B.3 The \((1, t, n)\)-LRM Scheme for \( t \geq 4 \)

Some of the results in Section 5.B.2 can be generalized to the \((1, t, n)\)-LRM Scheme, \( t \geq 4 \). In the \((1, t, n)\)-LRM scheme the size of each sliding window is \( t \). Therefore, to present the demodulated sequences of permutations the alphabet of the base-words has size \( t! \). Given \( t \) consecutive charge levels, \( c_i, c_{i+1}, \ldots, c_{i+t-1} \), the corresponding permutation related to \( \alpha_i \), from \( S_t \), is uniquely determined by the order of the \( t \) charge levels. The position of the symbol \( t \) in this permutation determines the value of \( g_i \), i.e., \( g_i = j, 0 \leq j \leq t-1 \), if \( t \) is in position \( t - j \) in the permutation. Therefore, a base-word uniquely determines the related codeword. To obtain the original base-word from the given codeword would be easy if for some \( i \), \( \alpha_i \) is given (in fact only the permutation related to \( t - 1 \) consecutive cells is required). If no such permutation is given then the task becomes more complicated.

Given a sub-codeword \((g_0, g_1, \ldots, g_{n-t+1})\), the relation between the charge levels in the \( i \)th cell, \( c_i \), and the first \( t - 1 \) cells, \( c_0, c_1, \ldots, c_{t-2} \), might have a few options. These options will be denoted by \( 0, 1, \ldots, t-1 \), where \( j, 0 \leq j \leq t-1 \), represents that \( c_i \) is in position \( t - j \) among the \( t \) charge levels \( c_0, c_1, \ldots, c_{t-2} \) and \( c_i \) (counting from the highest, the first position, to the lowest, the \( t \)th position). For each \( i, 2t-3 \leq i \leq n-1 \), we provide two properties regarding the charge levels \( c_{i-t+2}, c_{i-t+3}, \ldots, c_i \):

\begin{enumerate}
  \item [(Q.1)] the permutation induced by \( c_{i-t+2}, c_{i-t+3}, \ldots, c_i \);
  \item [(Q.2)] the set of all possible \((t-1)\)-tuples of the relations between the charge levels \( c_{i-t+2}, c_{i-t+3}, \ldots, c_i \) and the charge levels of the first \( t - 1 \) cells.
\end{enumerate}

The elements of the set defined in (Q.2) will be denoted by a \((t-1)\)-tuple \((x_1, x_2, \ldots, x_{t-1})\), \( x_j \in \{0, 1, \ldots, t-1\}, 1 \leq j \leq t-1 \), where \( x_j \) represents the relation between \( c_{i-t+j+1} \) and the first \( t - 1 \) cells. Note, that not all the \( t^{t-1} (t-1)\)-tuples can be obtained for a given permutation defined by (Q.1).

We call each set of properties defined by (Q.1) and (Q.2) a state, and the state at index \( i \) will be denoted by \( P_i \). Given a sub-codeword \( g' = (g_0, g_1, \ldots, g_{n-t-1}, g_{n-t}) \), the states in the sequence \((P_{2t-3}, P_{2t-2}, \ldots, P_{n-1})\)
are determined one after one in this order. It is easy to verify that if \( P_i = P_j \) for some \( 2t - 3 \leq i < j < n - 1 \) then \( P_{i+1} = P_{j+1} \). A state which has the all possibilities in the second property will be called a complete state. It is easy to verify that in the \((1, t, n)\)-LRM there are \((t - 1)!\) complete states (defined by the permutations of (Q.1)).

Given \( P_i \) which is a complete state and any value of \( g_{t-1} \), it is easily verified that the succeeding state \( P_{t+1} \) is also a complete state.

Denote by \( \pi \) the permutation defined by the charge levels in the first \( t - 1 \) cells. Given \( \pi \) and \( g' = (g_0, g_1, \ldots, g_{n-t-1}, g_{n-t}) \), the sub-base-word \((\alpha_0, \alpha_1, \ldots, \alpha_{n-t})\) of a realizable base-word which corresponds to \( \pi \) and \( g' \) is determined unambiguously. But, \( \alpha_{n-t+1}, \alpha_{n-t+2}, \ldots, \alpha_{n-1} \) might have a few options. These options are determined by the state \( P_{n-1} \) and the permutation \( \pi \). Each option provides a distinct base-word which is represented by the state \( P_{n-1} \) and the permutation \( \pi \).

We generate a table to enumerate the number of base-words represented by \( \pi \) and \( P_{n-1} \) for the \((t - 1)!\) complete states. Given \( g' \), \( c_0 \) is compared with \( c_{n-t+1}, c_{n-t+2}, \ldots, c_{n-1} \) to obtain \( g_{n-t+1} \); \( c_1 \) is compared with \( c_{n-t+2}, \ldots, c_{n-1}, c_0 \) to obtain \( g_{n-t+2} \), and so on until \( c_{t-2} \) is compared with \( c_{n-1}, c_0, \ldots, c_{t-3} \) to obtain \( g_{n-1} \). It can be proved that in this table the sum of values in each row and in each column is equal to \( t^{t-1} \).

The next step is to find a sequence \((a_1, a_2, \ldots, a_r)\), \( r \geq t \), with the following properties. If a sub-codeword \( g' = (g_0, g_1, \ldots, g_{n-t}) \) contains the sequence \((a_1, a_2, \ldots, a_r)\) as a subsequence at indices \((i-r+1, i-r+2, \ldots, i-1, i)\), then this sequence ends in a complete state, i.e. \( P_{t+t-1} \) is one of the \((t - 1)!\) complete states. The reason is that in this case \( c_{i+1}, c_{i+2}, \ldots, c_{i+t-1} \) have no dependency on the charge levels of \( c_{i-r+1}, c_{i-r+2}, \ldots, c_{i-t+1} \), i.e., each one of the charge levels \( c_{i+1}, c_{i+2}, \ldots, c_{i+t-1} \) can be lower than \( c_{i-r+1}, c_{i-r+2}, \ldots, c_{i-r+t-1} \), between them \((t - 2)\) options, or higher than all of them. Therefore, the relations between \( c_{i+1}, c_{i+2}, \ldots, c_{i+t-1} \), and the charge levels in the first \( t - 1 \) cells, \( c_0, c_1, \ldots, c_{t-2} \), have all the possibilities, i.e., \( P_{t+t-1} \) is a complete state. This implies that also \( P_{n-1} \) is a complete state.

Now, the Perron-Frobenius Theorem can be used to compute the number of sequences, of length \( n - t + 1 \) over alphabet \( \{0, 1, \ldots, t - 1\} \), which don’t include \((a_1, a_2, \ldots, a_r)\) as a subsequence. It is required that this number will tend to \( \beta^{n-t+1} \) where \( n \) tends to \( \infty \) and \( \beta \) is a constant (related to the largest
eigenvalue of the transition matrix of the related automata) for which $\beta < t$. Now, it can be concluded with the generated table that if $M_t$ is the number of legal codewords in the $(1, t, n)$-LRM scheme then $\lim_{n \to \infty} \frac{M_t}{t^n} = 1$.

For $t = 4$ one required such subsequence is $(3, 3, 0, 1, 2, 1)$ for which $\beta = 3.99902$, and hence

**Theorem 18.** $\lim_{n \to \infty} \frac{M_4}{t^n} = 1$.

The problem of finding the value of $\lim_{n \to \infty} \frac{M_t}{t^n}$ can be formulated as a purely combinatorial problem. Let

$$\mathcal{A}_t \overset{\text{def}}{=} \{(a_0, \ldots, a_{t^n-1}) : a_j \in \mathbb{Z}, 0 \leq j \leq t^n - 1, \quad |\{a_j, a_{j+1}, \ldots, a_{j+t-1}\}| = t\},$$

$\Pi(b_1, b_2, \ldots, b_t)$ be the permutation from $S_t$ defined by the subsequence $(b_1, b_2, \ldots, b_t)$, and

$$\mathcal{S}_t^n \overset{\text{def}}{=} \{(\pi_0, \ldots, \pi_{t^n-1}) : \pi_j \in S_t, 0 \leq j \leq t^n - 1, \quad \pi_j = \Pi(a_j, a_{j+1}, \ldots, a_{j+t-1}), \quad (a_0, \ldots, a_{t^n-1}) \in \mathcal{A}_t^n\},$$

where the indices are taken modulo $t^n$.

What is the value of $\lim_{n \to \infty} \frac{|\mathcal{S}_t^n|}{t^n}$? We conjecture that the value is 1 and proved this value for $t = 3$ and $t = 4$.

### 5.B.4 Constant Weight $(1, 2, n)$-LRM Gray Codes

One important topic related to rank modulation is an order of the codewords in such a way that each codeword will define an alphabet letter. This implies that $n$ consecutive cells define an alphabet letter and any change in the charge levels of some cells relates to a change in the alphabet letter. The most effective ordering is a Gray code ordering, i.e., a codeword is obtained by a minimal change in the codeword which proceed it. This should be a consequence of a small change in the related charge levels. In the rank modulation scheme this change in the charge levels is obtained by the push-to-the-top operation in a window of length $t$. We will concentrate only in the case where $t = 2$ since in this case the windows have length two and hence the permutations are from $S_2$, i.e., we can use binary codewords. In this
respect, we will be interested also in the case where all the codewords have the same weight. We define an \((1, 2, n; w)\)-LRMGC to be an \((1, 2, n)\)-LRM Gray code, where the codewords are ordered in an order which defines a Gray code and each codeword has weight \(w\). If all the codewords have the same weight then we can bound the difference between the charge levels of a cell on which the push-to-the-top operation is performed [2]. Two codewords are adjacent only if they differ in two positions, where 01 can be changed to 10. In this respect, the last codeword and the first codeword are also considered to be adjacent. It was conjectured in [2] that an \((1, 2, n; 2)\)-LRMGC has at most \(2^n\) codewords and such a code with \(2^n\) codewords was constructed. We have been able to prove this conjecture by a careful analysis of the path mentioned in [2]. The proof of the following theorem will be presented in the full version of this paper.

**Theorem 19.** An \((1, 2, n; 2)\)-LRMGC has at most \(2^n\) codewords.

### 5.B.5 Conclusions and Open Problems

In this paper, encoding, decoding, and enumeration of the \((1, t, n)\)-LRM scheme are studied. A complete solution is given for the \((1, 3, n)\)-LRM scheme. Encoding for the \((1, t, n)\)-LRM scheme for each \(t \geq 3\) is presented. For \((1, 3, n)\) a related decoding was presented. We also proved for \(t \in \{3, 4\}\) that if \(M_t\) is the number of legal codewords in the \((1, t, n)\)-LRM scheme then

\[
\lim_{n \to \infty} \frac{M_t}{2^n} = 1.
\]

We conclude with some problems for future research raised in our discussion.

- Find an efficient algorithm to determine if a given codeword in the \((1, t, n)\)-LRM scheme, for \(t \geq 3\), is legal or not.

- Find an efficient decoding algorithm for the \((1, t, n)\)-LRM scheme, \(t \geq 4\).

- Prove that if \(t > 4\) then

\[
\lim_{n \to \infty} \frac{M_t}{2^n} = 1.
\]

The key is to find the appropriate subsequences.

- For \(w > 2\), find optimal \((1, 2, n; w)\)-LRMGC.
Acknowledgment

This work was supported in part by the U.S.-Israel Binational Science Foundation, Jerusalem, Israel, under Grant No. 2012016.
Bibliography


Chapter 6

Rennconstruction of Sequences over Non-Identical Channels

Michal Horovitz and Eitan Yaakobi

A Note

The long version paper [21] which is presented in Subchapter 6.A, includes new results as well as some corrections and more accurate details for the results which have already published in the short version [19], Subchapter 6.B. For example, Theorem 39 in [19] is fixed in the respective claim, Proposition 28 in [21].

6.A Unpublished Full Version

Abstract

Motivated by the error behavior in the DNA storage channel, in this work we extend the previously studied sequence reconstruction problem by Levenshtein. The reconstruction problem studies the model in which the information is read through multiple noisy channels, and the decoder, which receives all channel estimations, is required to decode the information. For
the combinatorial setup, the assumption is that all the channels cause at most some $t$ errors. However, since the channels do not necessarily have the same behavior, we generalize this model and assume that the channels are not identical and thus may cause a different maximum number of errors. For example, we assume that there are $N$ channels, which cause at most $t_1$ or $t_2$ errors, where $t_1 < t_2$, and the number of channels with at most $t_1$ errors is at least $\lceil pN \rceil$, for some fixed $0 < p < 1$. If the information codeword belongs to a code with minimum distance $d$, the problem is then, to find the minimum number of channels that guarantees successful decoding in the worst case. A different problem we study in this work is where the number of channels is fixed, and the question is finding the minimum distance $d$ that provides exact reconstruction.

6.A.1 Introduction

The sequence reconstruction problem was first proposed and studied by Levenshtein in [8, 9]. In this model, a codeword is transmitted over multiple channels and a decoder, which receives all channel outputs, decodes the transmitted word. The assumption is that all channels are the same and are uncorrelated, with the only exception that all channel outputs are different from each other. This model was originally motivated by chemical and biological processes where the information is replicated and can be read from different noisy sources. However, it was also shown to be relevant in storage technologies, where the stored information has multiple copies or where a single copy is read by several different read heads. Specifically, the applicability of this model is most relevant to DNA storage [1, 2, 16, 17, 18]. Both for in vitro and in vivo storage systems, the information has a large number of copies stored in DNA strands and the goal is to read these strands and reconstruct the data, while every estimation of the data is erroneous.

The reconstruction model studied by Levenshtein and later by others was combinatorial. Suppose all words belong to some space $V$ with a distance function $\rho$. It is assumed that the information codeword $x$ belongs to a code with minimum distance $d$ and the number of errors in every channel is at most $t$. Then, the goal is to find the minimum number of channels that guarantees unique decoding in the worst case. Clearly, if $t < \lfloor (d - 1)/2 \rfloor$, then a single channel is sufficient. Otherwise, it was shown in [8] that this
number has to be greater than the largest intersection of two balls with radius \( t \) and minimum distance \( d \) between their centers, that is, greater than
\[
\max\{|B_t(x) \cap B_t(z)| : x, z \in V, \rho(x, z) \geq d\},
\]
where \( B_t(x) = \{y \in V : \rho(x, y) \leq t\} \). Later, this combinatorial problem was studied for several channels. In [8], Levenshtein studied the cases of substitution errors, the Johnson graphs, and several more general metric distances. More results for other general error graphs were given in [10, 11], and in [5, 6, 7], it was studied for permutations. The case of permutations with the Kendall’s \( \tau \) distance was investigated in [15] as well as the Grassmann graph case. Levenshtein’s results for deletions and insertions in [9] were extended in [13] for insertions and in [3] for deletions. In [14], the connection between the reconstruction problem and associative memories was studied, and in [4] it was analyzed for the purpose of asymptotically improving the Gilbert-Varshamov bound.

The motivation for the paradigm studied in this paper comes from the error behavior in DNA storage. We generalize Levenshtein’s model and assume a combinatorial model where the channels are not identical. When reading the data stored in DNA strands, it may happen that some estimations are more noisy than the others [16]. In the reconstruction model this is translated to channels that cause a different maximum number of errors. For example, it is known that for substitution errors, if the transmitted word belongs to a code with minimum Hamming distance 3 and there are at most 2 errors in every channel, then 7 channels are necessary and sufficient for successful decoding. However, if at most 2 channels cause two errors (and the rest 1 error), then we show that 5 channels are necessary and sufficient for successful decoding. In [12], a similar problem was studied for the setup in which every channel can cause a different number of insertions.

In this paper, we mainly focus on studying the following two problems. In the first problem we assume that the input set, i.e. the code and its minimum distance, is given, and we seek to find the minimum number of channels required for exact reconstruction. In the second problem, the number of channels is given, and then we study the minimum distance of a code which is required for exact reconstruction. In these two problems, we consider three cases which depend upon whether the information about the number
of errors in each channel is given, only the distribution of the number of errors in the channels is given, or only the average number of errors is given.

Formally, we define the first model as follows. Let $\ell$ be the number of possible types of channels. For $T = (t_1, \ldots, t_\ell)$ and $P = (p_1, \ldots, p_\ell)$, where $t_1 < \cdots < t_\ell \in \mathbb{N}$ and $0 < p_1 < \cdots < p_{\ell-1} < p_\ell = 1$, we say that a system with $N$ channels is a $(T, P)$-channel system if for all $i$, $1 \leq i \leq \ell$, $\lceil p_i N \rceil$ of the channels cause at most $t_i$ errors. For example, Levenshtein’s model is a special case with $\ell = 1$ and $p_1 = 1$. We study the minimum number of channels required for a $(T, P)$-channel system for successful decoding when the information is a codeword which belongs to a code with minimum distance $d$. The first two cases we consider here take into account whether the decoder may or may not know the type of each channel. We also study the case where only the average number of errors is known. We solve the general cases, and focus on substitution and transposition errors for $\ell = 2$.

In the second problem the number of channels $N$ is fixed, and the goal is to find the minimum distance $d$ of a code which guarantees exact reconstruction. As in the first problem, we consider here also three parallel questions according to the knowledge about the error behavior of the channels. That is, whether the decoder knows the number of errors in each channel, the distribution of the number of errors, or only the average number of errors.

The rest of the paper is organized as follows. In Section 6.A.2, we formally define the models and the main problems studied in the paper. In Section 6.A.3 we solve the first problem, i.e., we find the minimum number of channels required for exact reconstruction, where the input set is given, and in Section 6.A.4 we demonstrate this solution for substitution and transposition errors. Then, in Section 6.A.5 we solve the second question, where the number of channels is given, and the goal is to find the minimum distance of the code required for exact reconstruction. Later on, in Section 6.A.6, we consider two special systems. Finally, we conclude in Section 6.A.7.

6.A.2 Definitions and Problems Setup

For a positive integer $h$, we denote by $[h]$ the set $\{1, 2, \ldots, h\}$. Let $V$ be a finite set with a distance function $\rho : V \times V \rightarrow \mathbb{N}$, when $\mathbb{N}$ is the set of all non-negative integer numbers. For $x \in V$, the ball of radius $t$ centered at $x$ is the set $B_t(x) = \{y \in V : \rho(x, y) \leq t\}$. A combinatorial channel $C$ is
called a $t$-error channel, if for any input $x \in V$ the output of $C$ is in $B_t(x)$. Note that for $t < t'$, a $t$-error channel is also a $t'$-error channel.

A **channel system** is a system consisting of some $N$ combinatorial channels $C_1, C_2, \ldots, C_N$. The **size** of the channel system is the number of channels $N$ comprised in it. We say that a word $x \in V$ is transmitted over the channel system if $x$ is transmitted over $C_i$ for all $i \in [N]$, and $y_i$ is the output of the $i$th channel. The sequence $(y_1, \ldots, y_N)$ is called the **outputs sequence** of the system. The receiver applies a decoding function $D(y_1, \ldots, y_N)$ in order to reconstruct the transmitted word $x$, and **exact reconstruction** happens when $x = D(y_1, \ldots, y_N)$. In this paper we only refer to the exact reconstruction problem and we assume that all channel outputs are different from each other. We say that a channel system **supports exact reconstruction for** $U$, $U \subseteq V$, if there exists a decoding function $D$ such that for each $x \in U$, $x = D(y_1, \ldots, y_N)$ where $(y_1, \ldots, y_N)$ is a possible outputs sequence when $x$ is transmitted over the system.

Let $T = (t_1, \ldots, t_\ell)$ and $P = (p_1, \ldots, p_\ell)$ be such that $t_1 < t_2 < \cdots < t_\ell \in \mathbb{N}$ and $0 < p_1 < p_2 < \cdots < p_{\ell-1} < p_\ell = 1$. A channel system with $N$ combinatorial channels is called a $(T, P)$-**channel system** if for each $i \in [\ell]$, $\lceil p_i N \rceil$ of the channels are $t_i$-error channels.

We consider two models, which depend upon whether the behavior of each specific channel is known or unknown to the decoder. In the first channel system, called the **sequenced-channel system**, the decoder knows the maximum number of errors in every channel. In this model we assume, without loss of generality, that for each $i \in [\ell]$ the first $\lceil p_i N \rceil$ channels are $t_i$-error. However, in the second channel system, called the **non-sequenced-channel system**, only the distribution of the errors in the channels is known to the decoder, but the number of errors in each individual channel is unknown. For example, in the non-sequenced model, the decoder knows that half of the channels are $t_1$-error channels, and the rest are $t_2$-error channels, but it does not know for each specific channel, if it is a $t_1$-error or $t_2$-error channel. But, in the sequenced model, the decoder knows also that the first half of the channels are $t_1$-error. We will also consider a channel system where only $t$, the average number of errors, is known, when $t$ is not necessarily an integer number. Such a system will be called a $t$-**channel system**.

For $U \subseteq V$, we denote by $N_u(T, P, U, V)$ the minimum size of a $(T, P)$-**non-sequenced-channel system** such that every $x \in U$ has exact recon-
struction. Similarly, \( N^k(T, P, U, V) \) and \( N^a(t, U, V) \) are defined for the sequenced- and \( t \)-channel systems, respectively. Note that \( N^k(T, P, U, V) \leq N^u(T, P, U, V) \), and clearly the values of \( N^k(T, P, U, V) \), \( N^u(T, P, U, V) \), and \( N(t, U, V) \) depend also on the distance function \( \rho \). Yet, the distance function will be clear from the context, so to simplify, we omit \( \rho \) in these notations.

In the rest of the paper, whenever we write \( g \), we refer to \( g \in \{k, u\} \). The first problem we study in this paper is formulated as follows.

**Problem 1.** Let \( V \) be a finite set with distance function \( \rho : V \times V \rightarrow \mathbb{N} \), \( T = (t_1, \ldots, t_\ell) \), \( t \geq 0 \), and \( P = (p_1, \ldots, p_\ell) \). For all \( U \subseteq V \), find the values of \( N^k(T, P, U, V) \), \( N^u(T, P, U, V) \), and \( N^a(t, U, V) \).

Problem 1 is a generalized version for the setup presented and solved by Levenshtein for \( \ell = 1 \) [8]. The solution for a general \( \ell \), which will be described in Subsection 6.A.3.2, extends the result by Levenshtein, where in this case \( T = (t) \), \( P = (1) \) and \( N^k(T, P, U, V) = N^u(T, P, U, V) \).

Problem 1 is studied in Section 6.A.3. In Subsection 6.A.3.1, we find the values \( N^k(T, P, U, V) \) and \( N^u(T, P, U, V) \) for \( \ell = 2 \), that is, the channel system is given by \( T = (t_1, t_2) \) and \( P = (p, 1) \). Then, in Subsection 6.A.3.2, we generalize this combinatorial solution for all \( \ell \), and in Subsection 6.A.3.3 we present the solution for \( N^a(t, U, V) \). In Section 6.A.4, we apply the solution for \( \ell = 2 \) for two types of errors; for substitutions in Subsection 6.A.4.1 and for transpositions in Subsection 6.A.4.2, both over the binary alphabet.

In Section 6.A.6 we slightly modify the model of a \((T, P)\)-channel system and study some special cases of \( \ell = 2 \) when there exists a fixed number of \( t_1 \)-error channels and the rest of the channels are \( t_2 \)-error channels, and vice versa. These problems are solved for both cases, the sequenced- and non-sequenced-channel systems.

In the sequel, when writing \( x \) and \( z \), we assume that \( x \neq z \), and we use a bold notation \((x, z)\) when \( x \) and \( z \) are vectors. It is straightforward to verify that

\[
N^g(T, P, U, V) = \max\{N^g(T, P, \{x, z\}, V) : x, z \in U\}, \quad \text{and}
\]
\[
N^a(t, U, V) = \max\{N^a(t, \{x, z\}, V) : x, z \in U\}.
\]

We also define \( N^g(T, P, d, V) \) as the maximum value of \( N^g(T, P, U, V) \), for all \( U \subseteq V \) of minimum distance at least \( d \), that is, \( d(U) \geq d \), where \( d(U) = \)

130
min{ρ(x, z) : x, z ∈ U}. As before, we get that

\[
N^g(T, P, d, V) = \max\{N^g(T, P, \{x, z\}, V) : x, z ∈ V, ρ(x, z) ≥ d\},
\]

and \(N^a(t, d, V)\) is defined similarly. Therefore, we focus on finding the values of \(N^g(T, P, \{x, z\}, V)\) and \(N^a(t, \{x, z\}, V)\) for all \(x, z ∈ U\). For \(V = \{0, 1\}^n\), we denote the values of \(N^g(T, P, d, V)\) and \(N^a(t, d, V)\) by \(N^g(T, P, d, n)\) and \(N^a(t, d, n)\), respectively.

The second problem we study in this paper is formulated as follows. Assume that \(N\), the size of the channel system, is given. We define three models of channel systems, each one is of size \(N\). Denote by \(t_i\) the number of maximum errors in the \(i\)-th channel, and by \(t\) the average number of errors, i.e., \(t = \frac{\sum_{i=1}^{N} t_i}{N}\). Without loss of generality \(t_1 ≤ t_2 ≤ ⋮ ≤ t_N\). We define the three models as follows,

1. a \((T, N)\)-sequenced-channel system - where the \(N\)-tuple \(T = (t_1, t_2, \ldots, t_N)\) is given,

2. a \((T, N)\)-non-sequenced-channel system - where the multiset \(T = \{t_1, t_2, \ldots, t_N\}\) is given, and

3. a \((t, N)\)-channel system - where \(t\), the average number of errors is given.

Then, we study the minimum distance \(d\) required for exact reconstruction in each one of these three models. We denote by \(D^k(T, N, V)\), \(D^a(T, N, V)\), and \(D^a(t, N, V)\), the minimum distance of the codes required for exact reconstruction for \((T, N)\)-sequenced-, \((T, N)\)-non-sequenced-, and \((t, N)\)-channel systems, respectively. Note that \(D^k(T, N, V) ≤ D^a(T, N, V) ≤ D^a(t, N, V)\), where we denote these values by \(∞\) if there does not exist such a minimum distance. In the sequel, we use the notation \(T = (t_1, \ldots, t_N)\) to denote the multiset \(\{t_1, \ldots, t_N\}\) or the \(N\)-tuple \((t_1, \ldots, t_N)\), where the meaning will be clear from the context, according to the model. Note that the parameter \(N\) is redundant in the non-sequenced and in the sequenced models, nevertheless, it is left for clarification. The second problem is defined as follows.
Problem 2. Let $V$ be a finite set with distance function $\rho : V \times V \to \mathbb{N}$. For all $N$, $T = (t_1, \ldots, t_N)$, and $t \geq 0$, find the values of $D^k(T, N, V)$, $D^u(T, N, V)$, and $D^a(t, N, V)$.

In Section 6.A.5 we solve Problem 2, using reduction to Problem 1. We also provide some examples for substitution errors.

6.A.3 Problem 1 - Minimum Size of a Channel-System

In this section we consider Problem 1 for every $U \subseteq V$. In Subsection 6.A.3.1 we find the minimum number of channels required for exact reconstruction in a $(T, P)$-channel system where $\ell = 2$, that is, the values of $N^u(T, P, U, V)$ and $N^k(T, P, U, V)$. Then, in Subsection 6.A.3.2, we extend this analysis for arbitrary $\ell$. Finally, in Subsection 6.A.3.3 we solve the problem of $N^a(t, U, V)$, where only the average number of errors is known. In the rest of this section, and unless stated otherwise, we assume that $U = \{x, z\} \subseteq V$.

6.A.3.1 The Case $\ell = 2$

In this subsection we study Problem 1 for $\ell = 2$. This result extends the case studied by Levenshtein when all the channels are identical [8]. For $\ell = 2$ we have, $T = (t_1, t_2)$, and $P = (p, 1)$, where $t_1 < t_2 \in \mathbb{N}$ and $0 < p < 1$. Recall that a $(T, P)$-channel system of size $N$ is a set of $N$ combinatorial channels, where $\lceil pN \rceil$ of the channels are $t_1$-error channels and the others are $t_2$-error channels.

For the rest of this subsection we assume that $T = (t_1, t_2)$ and $P = (p, 1)$.

We define

$$I(x, z, t_1, t_2) = B_{t_1}(x) \cap B_{t_2}(z), \quad I(x, z, t_1) = B_{t_1}(x) \cap B_{t_1}(z),$$

and

$$N(x, z, t_1, t_2) = |I(x, z, t_1, t_2)|, \quad N(x, z, t_1) = |I(x, z, t_1)|.$$ 

Note that the values of $B_{t_1}(x)$, $I(x, z, t_1, t_2)$, and $N(x, z, t_1, t_2)$ depend also on $V$ which is omitted to simplify the notations and it will be clear from the context.
We start with two simple propositions that will be used in the following theorem.

**Proposition 20.** For $r_1, r_2 \in \mathbb{N}$ and $0 < p \leq 1$ the followings hold.

(a) If $r_1 > \left\lceil \frac{r_2}{p} \right\rceil$, then $\lceil p \cdot r_1 \rceil > r_2$.

(b) If $r_1 \leq \left\lfloor \frac{r_2}{p} \right\rfloor$, then $\lceil p \cdot r_1 \rceil \leq r_2$.

**Proof.**

(a) For an integer $r_1$, the equality $r_1 > \left\lceil \frac{r_2}{p} \right\rceil$ is equivalent to $r_1 \geq \left\lfloor \frac{r_2}{p} \right\rfloor + 1$. Thus, we have

$$\lceil p \cdot r_1 \rceil \geq p \cdot r_1 \geq p \cdot \left\lfloor \frac{r_2}{p} \right\rfloor + 1 > p \cdot \frac{r_2}{p} = r_2.$$

(b) If $\lceil p \cdot r_1 \rceil = p \cdot r_1$, then

$$\lceil p \cdot r_1 \rceil = p \cdot r_1 \leq p \cdot \left\lfloor \frac{r_2}{p} \right\rfloor \leq p \cdot \frac{r_2}{p} = b.$$

Otherwise, $\lceil p \cdot r_1 \rceil = \lceil p \cdot r_1 \rceil + 1$, and

$$\lceil p \cdot r_1 \rceil = p \cdot r_1 + 1 < p \cdot \left\lfloor \frac{r_2}{p} \right\rfloor + 1 \leq p \cdot \left( \frac{r_2}{p} \right) + 1 = r_2 + 1 \quad \square$$

The following theorem solves Problem 1 for the sequenced model.

**Theorem 21.** $N^k(T, P, U, V) = N + 1$, where

$$N = \min \{ \lceil N(x, z, t_1)/p \rceil, N(x, z, t_2) \}.$$

**Proof.** The above conditions are symmetric for $x$ and $z$. Thus, without loss of generality, let $x$ be the transmitted word.

For the first direction we prove that a $(T, P)$-sequenced-channel system of size $J$, where $J > \min \{ \lceil N(x, z, t_1)/p \rceil, N(x, z, t_2) \}$, supports exact reconstruction for $U = \{x, z\}$. If $J > \min \{ \lceil N(x, z, t_1)/p \rceil, N(x, z, t_2) \}$ then at least one of the following conditions holds:

1. $J > \lceil N(x, z, t_1)/p \rceil$ which implies $\lceil pJ \rceil > N(x, z, t_1)$ by Proposition 20((a)), or
2. $J > N(x, z, t_2)$.
Assume the word $x$ was transmitted over the channel system. Therefore, at least $\lceil pJ \rceil$ outputs are in $B_{t_1}(x)$ and all the $J$ outputs are in $B_{t_2}(x)$. Thus, if the first condition holds, since at least $\lceil pJ \rceil$ of the outputs are in $B_{t_1}(x)$, then there are no $\lceil pJ \rceil$ outputs in $B_{t_1}(z)$, and if the second condition holds, since all $J$ outputs are in $B_{t_2}(x)$, not all the outputs are in $B_{t_2}(z)$. Hence, for both cases, the word $z$ will not be decoded.

For the second direction we prove that if the following two conditions hold simultaneously:

1. $J \leq \lfloor N(x, z, t_1)/p \rfloor$ which implies $\lceil pJ \rceil \leq N(x, z, t_1)$ by Proposition 20((b)), and
2. $J \leq N(z, x, t_2),$

then a channel system of size $J$ does not support exact reconstruction for $U = \{x, z\}$. We present a sequence of $J$ outputs which can be an outputs sequence when transmitting either $x$ or $z$. We chose the first $m = \min\{J, N(x, z, t_1)\}$ outputs from $I(x, z, t_1)$, where $m \geq \lceil pJ \rceil$ by the first condition. The rest of the outputs are in $I(x, z, t_2)$, which is possible by the second condition. Thus, the first $\lceil pJ \rceil$ outputs are in $B_{t_1}(x)$, and all the $J$ outputs are in $B_{t_2}(x)$, and the same holds for $z$.

The rest of this subsection is dedicated for presenting the solution for the non-sequenced model. Note that if $x$ is transmitted over a $(T, P)$-non-sequenced-channel system with $N$ channels, then at least $\lceil pN \rceil$ of the outputs are in $B_{t_1}(x)$, and all the $N$ outputs are in $B_{t_2}(x)$. Thus, to support exact reconstruction for $x$ we require that for every $z \in U$, there are no $N$ outputs such that all the following three conditions hold simultaneously:

1. at least $\lceil pN \rceil$ of the outputs are in $B_{t_1}(x),$
2. at least $\lceil pN \rceil$ of the outputs are in $B_{t_1}(z),$
3. all $N$ outputs are in $I(x, z, t_2) = B_{t_2}(x) \cap B_{t_2}(z)$.

Thus, we conclude

**Lemma 24.** A $(T, P)$-non-sequenced-channel system of size $J$ supports exact reconstruction for $U$, if and only if at least one of the following four conditions holds:

134
\[(1) \quad \lceil pJ \rceil > N(x, z, t_1, t_2),\]
\[(2) \quad \lceil pJ \rceil > N(z, x, t_1, t_2),\]
\[(3) \quad J > N(x, z, t_2), \text{ or} \]
\[(4) \quad 2 \lceil pJ \rceil - N(x, z, t_1) > J.\]

**Proof.** The above conditions are symmetric for \(x\) and \(z\). Thus, without loss of generality, let \(x\) be the transmitted word.

For the first direction we prove that if at least one of the above conditions holds, then the system supports exact reconstruction. If Condition (1) or Condition (3) holds, since at least \(\lceil pJ \rceil\) of the outputs are in \(B_{t_1}(x)\) and all the \(J\) outputs are in \(B_{t_2}(x)\), then not all the outputs can be in \(B_{t_2}(z)\). If Condition (2) holds, there are no \(\lceil pJ \rceil\) outputs in \(B_{t_1}(z)\). Thus, if one of conditions (1), (2), or (3) holds, then \(z\) will not be decoded.

For Condition (4), assume that we have \(m\) outputs in \(I(x, z, t_1)\), \(m \leq N(x, z, t_1)\). In order for \(z\) to be a possible output for the decoder, we must have at least \(\lceil pJ \rceil - m\) outputs in \(I(z, x, t_1, t_2)\setminus I(x, z, t_1)\). Furthermore, since \(x\) was transmitted at least \(\lceil pJ \rceil - m\) outputs are in \(I(x, z, t_1, t_2)\setminus I(x, z, t_1)\). Thus, we must have that \(2 \lceil pJ \rceil - m \leq J\) in contradiction to Condition (4).

For the second direction we prove that if the following four conditions hold simultaneously:

\[(1) \quad \lceil pJ \rceil \leq N(x, z, t_1, t_2),\]
\[(2) \quad \lceil pJ \rceil \leq N(z, x, t_1, t_2),\]
\[(3) \quad J \leq N(x, z, t_2), \text{ and} \]
\[(4) \quad 2 \lceil pJ \rceil - N(x, z, t_1) \leq J,\]

then the channel system of size \(J\) does not support exact reconstruction for \(U = \{x, z\}\). For this part, we present a set of \(J\) outputs, which any order of them may be an outputs sequence when transmitting either \(x\) or \(z\). Let \(m = N(x, z, t_1)\). If \(m < \lceil pJ \rceil\) then \(m\) outputs are chosen from \(I(x, z, t_1)\), at least \(\lceil pJ \rceil - m\) outputs are in \(I(x, z, t_1, t_2)\setminus I(x, z, t_1)\) (by Conditions (1) and (4)), at least \(\lceil pJ \rceil - m\) in \(I(z, x, t_1, t_2)\setminus I(x, z, t_1)\) (by Conditions (2) and (4)), and all the outputs are in \(I(x, z, t_2)\) (by Condition (3)). Otherwise, \(m \geq \lceil pJ \rceil\), and then at least \(\lceil pJ \rceil\) outputs are in \(I(x, z, t_1)\), and
all the rest are in \( I(x, z, t_2) \) (by Condition ((3))). Thus, at least \([pJ]\) of the outputs are in \( B_{t_1}(x) \), and all the \( J \) outputs are in \( B_{t_2}(x) \), and the same holds for \( z \).

We are now ready to find the value of \( N^u(T, P, U, V) \). Let us define

\[
N'(x, z, t_1, p) = \min\{L : 2 \lceil pL \rceil - L > N(x, z, t_1), L \geq 1\} - 1,
\]

(6.1)

where here and in the rest of this paper \( \min\emptyset = \infty \). The following theorem establishes our result in calculating the value of \( N^u(T, P, U, V) \) by using Lemma 24 and the definition of \( N'(x, z, t_1, p) \).

**Theorem 22.** \( N^u(T, P, U, V) = N + 1 \), where

\[
N = \min\{ \lceil N(x, z, t_1, t_2)/p \rceil, N(x, z, t_2), \lceil N(z, x, t_1, t_2)/p \rceil, N'(x, z, t_1, p) \}.
\]

**Proof.** If a \((T, P)\)-channel system consists of \( J = N + 1 \) channels, then, by the definition of \( N \), at least one of the following conditions holds:

1. \( J > \lceil N(x, z, t_1, t_2)/p \rceil \) which implies by Proposition 20((a)) \([pJ]\) > \(N(x, z, t_1, t_2)\),
2. \( J > \lceil N(z, x, t_1, t_2)/p \rceil \) which implies by Proposition 20((a)) \([pJ]\) > \(N(z, x, t_1, t_2)\),
3. \( J > N(x, z, t_2) \), or
4. \( 2 \lceil pJ \rceil - N(x, z, t_1) > J \).

Thus, by Lemma 24, a channel system of size \( J \) supports exact reconstruction.

For the second direction we have to prove that if \( J \leq N \), then \( J \) channels are not sufficient for exact reconstruction where \( U = \{x, z\} \). The following four conditions hold simultaneously:

1. \([pJ]\) \(\leq N(x, z, t_1, t_2)\) which is derived by Proposition 20((b)) from \( J \leq \lceil N(x, z, t_1, t_2)/p \rceil \).
2. \([pJ]\) \(\leq N(z, x, t_1, t_2)\) which is derived by Proposition 20((b)) from \( J \leq \lceil N(z, x, t_1, t_2)/p \rceil \).
(3) $J \leq N(x, z, t_2)$, and

(4) $2 \left\lceil pJ \right\rceil - N(x, z, t_1) \leq J$.

Then, we apply again Lemma 24 to conclude that exact reconstruction is not supported.

\textbf{Remark 1.} We note that a $(T, P)$-non-sequenced-channel system of size $J$, where $J > N^u(T, P, \{x, z\}, V)$, may not support exact reconstruction for $U = \{x, z\}$. That could happen only if $J \leq \min\{[N(x, z, t_1, t_2)/p], [N(z, x, t_1, t_2)/p], N(x, z, t_2)\}$. The reason for this undesirable phenomena is that $N^u(T, P, U, V)$ may be determined by $N'(x, z, t_1, p) + 1$, that is, $N^u(T, P, U, V) = N'(x, z, t_1, p) + 1$ which means that $2 \left\lceil pN^u(T, P, U, V) \right\rceil - N^u(T, P, U, V) > N(x, z, t_1)$. However, $2 \left\lceil pJ \right\rceil - J \leq N(x, z, t_1)$. For example, assume $V = \{0, 1\}^n$, $\rho$ is the Hamming distance function, $d = 3$, $T = (1, 4)$, and $P = (1/3, 1)$. Thus, we have $N^u(T, P, d, n) = 1$, but 2 channels (and even more) are not sufficient. This phenomena happens also for general $\ell$ in the non-sequenced model.

In order to complete the study of the value $N^u(T, P, U, V)$ we are only left with studying the value of $N'(x, z, t_1, p)$.

\textbf{Proposition 23.} For $0 < p \leq 1/2$:

$$N'(x, z, t_1, p) = \begin{cases} 0 & \text{if } N(x, z, t_1) = 0, \\ \infty & \text{otherwise.} \end{cases}$$

For $1/2 < p < 1$:

$$\left\lfloor \frac{N(x, z, t_1) - 2}{2p - 1} \right\rfloor \leq N'(x, z, t_1, p) \leq \left\lfloor \frac{N(x, z, t_1)}{2p - 1} \right\rfloor.$$ 

\textbf{Proof.} For $0 < p \leq 1/2$, $2 \left\lceil pL \right\rceil - L \leq 1$ for all $L$. Thus, in this case the value of $N'(x, z, t_1, p)$ is an immediate result by Equation (6.1).

For $1/2 < p < 1$, denote $a = N(x, z, t_1)$. For each $\delta$, $0 \leq \delta < 1$, let $A_\delta = \{J : [pJ] - pJ = \delta, J \geq 1\}$, and $J_\delta = \min\{L : 2 \left\lceil pL \right\rceil - L > a, L \geq 1, L \in A_\delta\}$. For $A_\delta \neq \emptyset$, let $L_\delta$ be an element in $A_\delta$, i.e., $[pL_\delta] - pL_\delta = \delta$. Thus, $2 \left\lceil pL_\delta \right\rceil - L_\delta > a$ if and only if $2(pL_\delta + \delta) - L_\delta > a$, which holds if and only if $L_\delta > \frac{a - 2\delta}{2p - 1}$, that is equivalent to $L_\delta \geq \left\lceil \frac{a - 2\delta}{2p - 1} \right\rceil + 1$. Thus, if $A_\delta \neq \emptyset$
then $J_\delta = \left\lceil \frac{a - 2\delta}{2p - 1} \right\rceil + 1$. We can conclude that $N'(x, z, t_1, p) = \min\{J_\delta - 1 : 0 \leq \delta < 1\}$. Therefore,

$$\left\lfloor \frac{N(x, z, t_1) - 2}{2p - 1} \right\rfloor \leq N'(x, z, t_1, p) \leq \left\lceil \frac{N(x, z, t_1)}{2p - 1} \right\rceil$$

$\square$

The following corollary is deduced immediately by Proposition 23 and Theorem 22.

**Corollary 3.** $N^u(T, P, U, V) = N + 1$ where $N$ is defined as follows. For $0 < p \leq 1/2$:

$$N = \begin{cases} 0 & \text{if } N(x, z, t_1) = 0 \\ \min\{ \lfloor N(x, z, t_1, t_2)/p \rfloor, N(x, z, t_2) \} & \text{otherwise.} \end{cases}$$

For $1/2 < p < 1$:

$$N = \min\{ \lfloor N(x, z, t_1, t_2)/p \rfloor, N(x, z, t_2), \lfloor N(z, x, t_1, t_2)/p \rfloor, N'(x, z, t_1, p) \}.$$

In Section 6.A.4 we show how to apply the result from Corollary 3 in order to explicitly solve Problem 1 with $\ell = 2$ for substitution and transposition errors over the binary alphabet.

**6.A.3.2 The General Case**

In this section, we extend the solution from Subsection 6.A.3.1. We provide a combinatorial translation for the general case of Problem 1, where $T = (t_1, \ldots, t_\ell)$ and $P = (p_1, \ldots, p_{\ell-1}, p_\ell)$, $t_1 < t_2 < \cdots < t_\ell \in \mathbb{N}$, and $0 < p_1 < p_2 < \cdots < p_{\ell-1} < p_\ell = 1$. Remember that a $(T, P)$-channel system of size $N$ consists of $N$ channels, where for each $i \in [\ell]$, $[p_iN]$ channels are $t_i$-error channels.

Theorem 24 and Theorem 25 generalize Theorem 21 and Theorem 22 for arbitrary $\ell$, respectively.
Theorem 24. $N^k(T, P, U, V) = N + 1$, where

$$N = \min\{\lfloor N(x, z, t_i)/p_i \rfloor : i \in [\ell] \}.$$  

Furthermore, for all $J \geq N^k(T, P, U, V)$, a $(T, P)$-sequenced-channel system of size $J$ supports exact reconstruction for $U$.

Proof. The above conditions are symmetric for $x$ and $z$. Thus, without loss of generality, let $x$ be the transmitted word. For the first direction we prove that a $(T, P)$-sequenced-channel system of size $J$, where $J > \min\{\lfloor N(x, z, t_i)/p_i \rfloor : i \in [\ell] \}$, supports exact reconstruction for $U = \{x, z\}$. If $J > \min\{\lfloor N(x, z, t_i)/p_i \rfloor : i \in [\ell] \}$ then there exists $i \in [\ell]$, such that $J > \lfloor N(x, z, t_i)/p_i \rfloor$, which implies by Proposition 20((a)) that $[p_i J] > N(x, z, t_i)$. The word $x$ was transmitted over the channel system. Therefore, at least $\lceil p_i J \rceil$ outputs are in $B_{t_i}(x)$. But, since $[p_i J] > N(x, z, t_i)$, there are no $\lceil p_i J \rceil$ outputs in $B_{t_i}(z)$. Hence, the word $z$ will not be decoded.

For the second direction we prove that if for all $i \in [\ell]$, $J \leq \lfloor N(x, z, t_i)/p_i \rfloor$ then a channel system of size $J$ does not support exact reconstruction for $U = \{x, z\}$.

By Proposition 20((b)), $J \leq \lfloor N(x, z, t_i)/p_i \rfloor$ implies $[p_i J] \leq N(x, z, t_i)$. Now, we present a sequence of $J$ outputs, which is an outputs sequence when transmitting either $x$ or $z$. We place the first $[p_i J]$ outputs in $I(x, z, t_1)$, then we add the next $([p_2 J] - [p_1 J])$ outputs in $I(x, z, t_2)$, then $([p_3 J] - [p_2 J])$ additional outputs in $I(x, z, t_3)$, and so on. This can be applied by the fact that $[p_i J] \leq N(x, z, t_i)$ for all $i \in [\ell]$. Thus, the first $[p_i J]$ of the outputs are in $B_{t_i}(x)$, for all $i \in [\ell]$, and the same holds for $z$. Hence, $z$ might be decoded when $x$ was transmitted.

Next, we consider the non-sequenced case. Recall that if $x$ is transmitted over a $(T, P)$-channel system of size $N$, then for all $i \in [\ell]$ at least $[p_i N]$ of the outputs are in $B_{t_i}(x)$ (if $i = \ell$ we have that all the $N$ outputs are in $B_{t_i}(x)$). Then, $x$ does not have exact reconstruction if there exists a different word $z$, where for all $i \in [\ell]$ at least $[p_i N]$ of the outputs are in $B_{t_i}(z)$.

Lemma 25. A non-sequenced-channel system of size $J$ does not support exact reconstruction for $U$ if and only if for all $i, j \in [\ell]$ the following inequality
holds

\[ N(x, z, t_i, t_j) \geq \lceil p_i J \rceil + \lceil p_j J \rceil - J. \]

Proof. For the first direction, we assume that the system does not support exact reconstruction. For all \( i \in [\ell] \), we denote by \( A_i, B_i \) the sets of outputs in \( B_{t_i}(x), B_{t_i}(z) \), respectively. Since the system does not support exact reconstruction for \( U = \{x, z\} \), we conclude that for all \( i, j \),

\[ |A_i| \geq \lceil p_i J \rceil, \quad |B_j| \geq \lceil p_j J \rceil, \quad \text{and} \quad J \geq |A_i \cup B_j|. \]

Thus, we have:

\[ J \geq |A_i \cup B_j| = |A_i| + |B_j| - |A_i \cap B_j| \geq \lceil p_i J \rceil + \lceil p_j J \rceil - N(x, z, t_i, t_j), \]

as required.

In the second direction we are given that for all \( i, j \in [\ell] \)

\[ N(x, z, t_i, t_j) \geq \lceil p_i J \rceil + \lceil p_j J \rceil - J, \]

and we prove that a channel system of size \( J \) does not support exact reconstruction for \( U = \{x, z\} \). For this part, we present a set of \( J \) outputs, which any order of them may be an outputs sequence when transmitting either \( x \) or \( z \). Let us assume that \( x \) is transmitted. The \( J \) outputs can be divided as follows. First, we place \( \lceil p_1 J \rceil \) outputs in \( I(x, z, t_1, t_\ell) \) by choosing the outputs according to their distance from \( z \), preferring the closest. Then, in the second step we do the same for \( t_2 \) to have additional \( (\lceil p_2 J \rceil - \lceil p_1 J \rceil) \) outputs in \( I(x, z, t_2, t_\ell) \). Then, in the third step we do the same for \( t_3 \), and so on. Thus, we have \( J \) outputs for \( x \) which were chosen by \( \ell \) steps.

We prove now, that this sequence of outputs, may be an outputs sequence when transmitting \( z \). We have to prove that for each \( i \in [\ell] \), at least \( \lceil p_i J \rceil \) outputs are in \( B_{t_i}(z) \). Let \( i \in [\ell] \). Then, for each \( j \in [\ell] \), we define

\[ a_{i,j} = \min\{\lceil p_j J \rceil - \lceil p_{j-1} J \rceil, N(x, z, t_j, t_i) - a_{i,j-1}\}, \]

where \( p_0 = 0, a_{i,0} = 0 \). By induction on \( j \), it is readily proved, that for all \( 0 \leq j \leq \ell \), at the \( j \)-th step of the algorithm, we choose exactly \( a_{i,j} \) additional words from \( B_{t_i}(z) \) (actually, from \( I(x, z, t_j, t_i) \)). Therefore, \( a_{i,j} \geq 0 \) for all \( i, j \), and the number of outputs in \( B_{t_i}(z) \) is \( a_i = \sum_{k=1}^{j} a_{i,k} \). Thus, we have to prove that \( a_i \geq \lceil p_i J \rceil \). If for all \( j \), \( a_{i,j} = \lceil p_j J \rceil - \lceil p_{j-1} J \rceil \), then \( a_i = J - \lceil p_0 J \rceil \geq \lceil p_i J \rceil \). Otherwise, let \( j_i \) be the largest index such that
\[ a_{i,j_i} = N(x, z, t_{j_i}, t_i) - a_{i,j_i-1}. \] Then,
\[ a_i \geq \sum_{k=j_i-1}^{\ell} a_{i,k} = J - \lceil p_j J \rceil + N(x, z, t_{j_i}, t_i) \geq \lceil p_i J \rceil. \]

In order to continue with the analysis to study the value of \( N^u(T, P, U, V) \), we define the following term:
\[
N'(x, z, t_{i}, t_{j}, p_i, p_j) = \min\{L : [p_i L] + [p_j L] - L > N(x, z, t_{i}, t_{j}), L \geq 1\} - 1.
\]

The following theorem establishes this result in calculating the value of \( N^u(T, P, U, V) \) by using Lemma 25 and \( N'(x, z, t_{i}, t_{j}, p_i, p_j) \).

**Theorem 25.** \( N^u(T, P, U, V) = N + 1 \), where
\[ N = \min\{N'(x, z, t_{i}, t_{j}, p_i, p_j) : i, j \in [\ell]\}, \]
that is,
\[
N = \min \{N'(x, z, t_{i}, t_{j}, p_i, p_j) : i, j \in [\ell] \}
\]
\[
\cup \{[N(z, x, t_{i}, t_{\ell}/p_i) : i \in [\ell - 1]\}
\]
\[
\cup \{N(x, z, t_{\ell})\}
\]
\[
\cup \{N'(x, z, t_{i}, t_{j}, p_i, p_j) : i, j \in [\ell - 1]\}.
\]

**Proof.** The proof where \( N = \min\{N'(x, z, t_{i}, t_{j}, p_i, p_j) : i, j \in [\ell]\} \) is essentially similar to the proof of Theorem 22 by using Lemma 25. Therefore we omit this part. For the other version of \( N \), note that by the definition of \( N'(x, z, t_{i}, t_{j}, p_i, p_j) \) in Equation (6.2), the followings hold. For all \( i \in [\ell - 1] \), we have that \( N'(x, z, t_{i}, t_{\ell}/p_i) = [N(x, z, t_{i}, t_{\ell}), p_i] \), \( N'(x, z, t_{i}, t_{\ell}/p_{\ell}) = [N(z, x, t_{\ell}), p_i] \), and \( N'(x, z, t_{i}, t_{\ell}/p_{\ell}) = N(z, x, t_{\ell}) \).

### 6.A.3.3 The t-Channel System

In this subsection we solve Problem 1 for the case where only \( t \), the average number of errors in all the channels, is known. That is, we find the value of \( N^u(t, U, V) \) for all \( t \geq 0 \) and \( U \subseteq V \).
Lemma 26. Let $U \subseteq V$, such that $d(U) = d$. Then,

$$N^a(t, U, V) = \min \{N : \lfloor tN \rfloor \leq \lfloor (dN - 1)/2 \rfloor \}. $$

Furthermore, a $t$-channel system of size $N$ supports exact reconstruction for $U$ if and only if $\lfloor tN \rfloor \leq \lfloor (dN - 1)/2 \rfloor$.

Proof. The outputs of the $N$ channels can be translated to one channel which causes at most $\lfloor tN \rfloor$ errors, and the transmitted word $x$ is translated to transmitting $x^N$. Thus, if $U \subseteq V$ is a code with minimum distance $d$, then $U^N$ is a code with minimum distance $dN$, consisting of $N$ repetitions of each word from $U$. Therefore, this code can correct at most $\lfloor (dN - 1)/2 \rfloor$ errors.

A $t$-channel system of size $N$ with input $U$ corresponds to one channel with the code $U^N$. Hence, a $t$-channel system of size $N$ supports exact reconstruction for $U$ if and only if $\lfloor tN \rfloor \leq \lfloor (dN - 1)/2 \rfloor$, and

$$N^a(t, U, V) = \min \{N : \lfloor tN \rfloor \leq \lfloor (dN - 1)/2 \rfloor \}. $$

Recall that $N^a(t, d, V)$ was defined as the maximum value of $N^a(t, U, V)$, for all $U \subseteq V$ such that $d(U) \geq d$. That is,

$$N^a(t, d, V) = \max \{N(t, U, V) : d(U) \geq d\}.$$  

The next proposition proves that $N^a(t, d, V) = N^a(t, U, V)$ for all $U = \{x, z\} \subseteq V$ such that $d = \rho(x, z)$, that is $N^a(t, U, V) \geq N^a(t, U', V)$ where $d(U) = d$ and $d(U') = d' > d$. This desirable property was defined by Levenshtein in [8] as the monotonicity by intersection.

Proposition 26. Let $U = \{x, z\} \subseteq V$ such that $\rho(x, z) = d$. Then,

$$N^a(t, U, V) = N^a(t, U, V).$$

Proof. The first direction $N^a(t, d, V) \geq N^a(t, U, V)$ is trivial by the definitions of $N^a(t, d, V)$ and $U$.

For the second direction, assume to the contrary that $N^a(t, d, V) < N^a(t, U, V)$. By Lemma 26 we conclude that $N^a(t, U, V)$ depends on the distance between $x$ and $z$, $d = \rho(x, z)$, and does not depend on the specific values of $x$ or $z$. Furthermore, a $t$-channel system of size $N$ supports exact reconstruction for $U$ if and only if $\lfloor tN \rfloor \leq \lfloor (dN - 1)/2 \rfloor$, and

$$N^a(t, U, V) = \min \{N : \lfloor tN \rfloor \leq \lfloor (dN - 1)/2 \rfloor \}. $$

□
Thus, we conclude that $N^a(t, d, V) < N^a(t, U, V)$ implies $\lfloor tN \rfloor \leq \lfloor (dN - 1)/2 \rfloor$, but, $\lfloor tN \rfloor > \lfloor (d'N - 1)/2 \rfloor$ for some $d' > d$. We get $\lfloor (d'N - 1)/2 \rfloor < \lfloor (dN - 1)/2 \rfloor$ for $d' > d$, which is a contradiction. Thus, $N^a(t, d, V) = N^a(t, U, V)$ for all $U = \{x, z\} \subseteq V$ of distance $d$.

Following, in this subsection, we search for an explicit solution for $N^a(t, d, V)$, which, by Proposition 26 equals to $N^a(t, U, V)$ for all $V$ of distance $d$. We prove some properties of this value and find the exact solution in almost all the cases.

Note that by applying Proposition 26, we can get the following result.

**Lemma 27.** For all $t \geq 0$ and a positive integer $d$, $N^a(t, d, V) = \infty$ or $N^a(t, d, V) = 1$.

*Proof.* Assume to the contrary that $1 < N = N^a(t, d, V)$ for $N \in \mathbb{N}$. Thus, by Lemma 26 we have $\lfloor t \rfloor > \lfloor (d - 1)/2 \rfloor$ and $\lfloor tN \rfloor \leq \lfloor (dN - 1)/2 \rfloor$. We prove by induction that if $\lfloor t \rfloor > \lfloor (d - 1)/2 \rfloor$, then for all $N' \in \mathbb{N}$, $\lfloor tN' \rfloor > \lfloor (dN' - 1)/2 \rfloor$. The basis of the induction is $N' = 1$. For the step, we assume that $\lfloor tN' \rfloor > \lfloor (dN' - 1)/2 \rfloor$, and we prove that $\lfloor t(N' + 1) \rfloor > \lfloor (d(N' + 1) - 1)/2 \rfloor$. The following inequality holds

$$\lfloor t(N' + 1) \rfloor \geq \lfloor tN' \rfloor + \lfloor t \rfloor > \lfloor (dN' - 1)/2 \rfloor + \lfloor (d - 1)/2 \rfloor + 1.$$

For even $d$ we can continue

$$\lfloor (dN' - 1)/2 \rfloor + \lfloor (d - 1)/2 \rfloor + 1 = (dN' - 2)/2 + (d - 2)/2 + 1 = (dN' + 1)/2 - 2 = \lfloor (dN' + 1)/2 \rfloor,$$

and, for odd $d$ we continue as follows:

$$\lfloor (dN' - 1)/2 \rfloor + \lfloor (d - 1)/2 \rfloor + 1 \geq (dN' - 2)/2 + (d - 1)/2 + 1 = (dN' + 1)/2 - 2 = \lfloor (dN' + 1)/2 \rfloor.$$

Thus, we conclude that $N^a(t, d, V) = \infty$ or $N^a(t, d, V) = 1$.

*Remark 2.* We note that a $t$-channel system of size $J$, where $J > N^a(t, U, V)$, may not support exact reconstruction for $U$ of distance $d$. 143
That could happen since \( \lfloor t \cdot N^a(t, U, V) \rfloor \leq \lfloor (d \cdot N^a(t, U, V) - 1)/2 \rfloor \) does not imply \( \lfloor tJ \rfloor \leq \lfloor (dJ - 1)/2 \rfloor \) for \( J > N^a(t, U, V) \), and by Lemma 26, a \( t \)-channel system of size \( J \) supports exact reconstruction for \( U \) if and only if \( \lfloor tJ \rfloor \leq \lfloor (dJ - 1)/2 \rfloor \). For example, if \( t = 7/4 \) and \( d = 3 \) then \( N^a(t, U, V) = 1 \), but a \( t \)-channel system of size 2 (or any even size) does not support exact reconstruction for \( U \) of distance \( d \).

As described in Remark 2, in the general case, a \( t \)-channel system of size \( J \) where \( J > N^a(t, U, V) \) may not support exact reconstruction for \( U \). Yet, in the rest of this subsection, we prove that in most of the cases, including for an integer \( t \), this undesirable phenomena does not hold. Furthermore, in this section we prove several properties on \( N^a(t, d, V) \), and we find an explicit solution for almost all the parameters \( t \) and \( d \). The proofs of all the claims in the rest of this subsection are presented in Appendix A.

**Lemma 28.** If a \( t \)-channel system of size \( J \), for some even \( J \), supports exact reconstruction for \( U \), then for all even positive integer, \( N \), a \( t \)-channel system of size \( N \) supports exact reconstruction for \( U \).

**Lemma 29.** Let \( N^a(t, d, V) = 1 \). If \( d \) is even or if a \( t \)-channel system of size 2 supports exact reconstruction for \( U \) of distance \( d \), then for all \( N \geq 1 \) a \( t \)-channel system of size \( N \) supports exact reconstruction for \( U \).

**Theorem 27.** Let \( t \geq 0 \), \( d \) be a positive integer, and \( U \subseteq V \) where \( d(U) = d \). Then

1. If \( d < \lfloor 2t \rfloor \) then \( N^a(t, d, V) = \infty \),
2. If \( d \geq \lfloor 2t \rfloor \) then \( N^a(t, d, V) = 1 \), and exact reconstruction is supported for \( U \) for any size of the system.
3. If \( d = \lfloor 2t \rfloor \) and \( d \) is even then \( N^a(t, d, V) = \infty \).
4. If \( d = \lfloor 2t \rfloor \) and \( d \) is odd then for all even \( N \), a \( t \)-channel system of size \( N \) does not support exact reconstruction.

Thus, for an integer \( t \), if \( d > 2t \) then \( N^a(t, d, V) = 1 \), and exact reconstruction is supported for \( U \) for any size of the system. Otherwise, \( N^a(t, d, V) = \infty \).
6.A.4 Problem 1 - Examples

In this section we apply the solution for Problem 1 with \( \ell = 2 \) for two types of errors, for substitution errors in Subsection 6.A.4.1, and for transposition errors in Subsection 6.A.4.2. In this section we use the notations which defined in Section 6.A.2, where in some of the notations we add the subscript \( H \) or \( J \), to denote the Hamming or the Johnson distance function for the case of substitution or transposition errors, respectively.

6.A.4.1 Substitution Errors

Let \( V_H = \{0, 1\}^n \) be the set of all length \( n \) words over the binary alphabet. The Hamming distance function \( \rho_H : V_H \times V_H \to \mathbb{N} \) is defined by \( \rho_H(x, z) = |\{i : x_i \neq z_i\}|. \)

Note, that for all \( x, z \in V_H \), \( N_{H}(x, z, t_1, t_2) \) and \( N_{H}(x, z, t) \) depend only on \( d = \rho_H(x, z) \). Thus, for \( x, z \in V_H \) such that \( d = \rho_H(x, z) \), we denote by \( N_{H}(d, n, t_1, t_2) \) and \( N_{H}(d, n, t) \) the values \( N_{H}(x, z, t_1, t_2) \) and \( N_{H}(x, z, t) \), respectively. The next proposition proves that for all \( d \geq 1 \), \( N_{g,H}(T, P, d, V) \geq N_{g,H}(T, P', U, n) \) where \( d(U) = d \) and \( d(U') = d' > d \). This desirable property is known as the monotonicity by intersection [8]. Recall that \( N_{g,H}(T, P, d, V) \) was defined as the maximum value of \( N_{g,H}(T, P, U, V) \), for all \( U \subseteq V \) such that \( d(U) \geq d \). That is,

\[
N_{g,H}(T, P, d, V) = \max\{N_{g,H}(T, P, U, V) : d(U) \geq d\}.
\]

Thus we prove in Proposition 28 that \( N_{g,H}(T, P, d, n) = N_{g,H}(T, P, \{x, z\}, V) \) for all \( U = \{x, z\} \subseteq V_H \) such that \( d = \rho_H(x, z) \).

**Proposition 28.** Let \( U = \{x, z\} \subseteq V_H \) such that \( \rho_H(x, z) = d \). Then, \( N_{g,H}(T, P, d, V) = N_{g,H}(T, P, U, V) \).

**Proof.** The first direction \( N_{g,H}(T, P, d, V) \geq N_{g,H}(T, P, U, V) \) is trivial by the definitions of \( N_{g,H}(T, P, d, V) \) and \( U \). For the second direction, we note that in the Hamming case, \( N_{g,H}(T, p, U, V) \) depends on the distance between \( x \) and \( z \), \( d = \rho(x, z) \), and does not depend on the specific values of \( x \) or \( z \). It can be readily verified that \( N_{H}(d, n, t_1) \), \( N_{H}(d, n, t_1, t_2) \), and \( N_{H}'(d, t_1, p) \) are non-increasing functions of \( d \). Therefore, the function \( N_{g,H}(T, P, U, V) \) for \( U = \{x, z\} \subseteq \{0, 1\}^n \) of distance \( d \) is defined as the minimum between
some of these non-increasing functions. Thus, for all \( U' \) of distance \( d' \), where \( d' > d \), \( N^g_H(T, P, U, n) \geq N^g_H(T, P, U', n) \), Hence \( N^g_H(T, P, U, n) = N^g_H(T, P, d, n) \).

According to Proposition 28, in order to calculate the value of \( N^g_H(T, P, d, n) \) it is enough to find the value of \( N^g_H(T, P, \{x, z\}, V_H) \), where \( \rho_H(x, z) = d \) and \( x, z \in V_H \). Therefore, according to Theorem 21 and Theorem 22, for \( T = (t_1, t_2) \) and \( P = (p, 1) \), we conclude that

\[
N^k_H(T, P, d, n) = \min\{\lfloor N_H(d, n, t_1)/p \rfloor, N_H(d, n, t_2)\} + 1, \tag{6.3}
\]

and

\[
N^n_H(T, P, d, n) = \min\{\lfloor N_H(d, n, t_1)/p \rfloor, N_H(d, n, t_2), N'_H(d, t_1, p)\} + 1,
\]

where

\[
N'_H(d, t_1, p) = \min\{L : 2 \lfloor pL \rfloor - L > N_H(d, n, t_1), L \geq 1\} - 1.
\]

In the sequel, we find the values of \( N^k_H(T, P, d, n) \) and \( N^n_H(T, P, d, n) \). We start by computing the values of \( N_H(d, n, t) \) and \( N_H(d, n, t_1, t_2) \).

The following lemma was shown in [8], where we use the equality \( t - \left\lfloor \frac{d}{2} \right\rfloor = \left\lfloor t - \frac{d}{2} \right\rfloor \).

**Lemma 30.** For \( t, d \geq 1 \),

\[
N_H(d, n, t) = \sum_{i=0}^{\lfloor t - \frac{d}{2} \rfloor} \left( \begin{array}{c} n - d \\ i \end{array} \right) \cdot \sum_{k=d-t+i}^{t-i} \left( \begin{array}{c} d \\ k \end{array} \right),
\]

where \( \binom{a}{b} = 0 \) if \( a < b \) or \( b < 0 \).

We conclude that in the sequenced model the solution is simply derived by the expression in (6.3) while we use the result from Lemma 30. In particular, for a fixed \( p \) and \( n \) sufficiently large, we get that \( N^k_H(T, P, d, n) = \lfloor N_H(d, n, t_1)/p \rfloor \). Furthermore, at the end of this subsection, we provide some examples.

For the non-sequenced model, we compute the value of \( N_H(d, n, t_1, t_2) \) for all \( t_1, t_2 \). This value is presented in the following lemma, which generalizes
Lemma 30, and is proved by similar combinatorial computation. The value $N_H(d, n, t)$, which is presented in Lemma 30, can be obtained from Lemma 31, by substituting $t = t_1 = t_2$. For the completeness of the results in the paper we prove the following lemma in Appendix B.

**Lemma 31.** For $1 \leq t_1 \leq t_2$ and $d \geq 1$,

$$N_H(d, n, t_1, t_2) = \sum_{i=0}^{\lfloor \frac{t_1 + t_2 - d}{2} \rfloor} \binom{n - d}{i} \cdot \sum_{k=d-t_2+i}^{t_1-i} \binom{d}{k}.$$  

The following two lemmas compare between the three components which determine the value of $N_H(T, P, d, n)$, for $d \geq 1$, $t_1 < t_2 \in \mathbb{N}$, and fixed $0 < p < 1$. Lemma 32 compares between $[N_H(d, n, t_1, t_2)/p]$ and $N_H(d, n, t_2)$.

**Lemma 32.** For any fixed $p$ and $n$ sufficiently large the following holds. If $d$ is odd, $p \leq 1/2$, and $t_2 = t_1 + 1$, then

$$N_H(d, n, t_2) < \lfloor N_H(d, n, t_1, t_2)/p \rfloor.$$  

Otherwise,

$$N_H(d, n, t_2) \geq \lfloor N_H(d, n, t_1, t_2)/p \rfloor.$$  

**Proof.** Note that

$$N_H(d, n, t_2) = \Theta(n^{\lfloor \frac{2t_2-d}{2} \rfloor})$$  

and

$$N_H(d, n, t_1, t_2) = \Theta(n^{\lfloor \frac{t_1 + t_2 - d}{2} \rfloor}).$$  

Thus, we compare between the powers $\lfloor \frac{2t_2-d}{2} \rfloor$ and $\lfloor \frac{t_1 + t_2 - d}{2} \rfloor$. If $t_2 = t_1 + 1$ and $d$ is odd then $\lfloor \frac{2t_2-d}{2} \rfloor = \lfloor \frac{t_1 + t_2 - d}{2} \rfloor$. In all other cases, $\lfloor \frac{2t_2-d}{2} \rfloor < \lfloor \frac{t_1 + t_2 - d}{2} \rfloor$, and hence $N_H(d, n, t_2) > \lfloor N_H(d, n, t_1, t_2)/p \rfloor$.

For the case of $t_2 = t_1 + 1$ and odd $d$, we compare the coefficients of the dominant powers. Denote $d = 2m + 1$.

$$N_H(d, n, t_2) = \left(\binom{d}{m} + \binom{d}{m+1}\right) \cdot \binom{n-d}{t_1-m} + \sum_{k=m-1}^{m+2} \binom{d}{k} \cdot \binom{n-d}{t_1-m} + \Theta(n^{t_1-m-2}),$$

$$N_H(d, n, t_1, t_2) = \binom{d}{m} \cdot \binom{n-d}{t_1-m} + \sum_{k=m-1}^{m+1} \binom{d}{k} \cdot \binom{n-d}{t_1-m} + \Theta(n^{t_1-m-2}).$$
Thus, the coefficient of the dominant powers in $N(d, t_2)$ is twice the coefficient of the corresponding term in $N(d, t_1, t_2)$. But, $N_H(d, n, t_1, t_2)$ is multiplied by $1/p$. Thus, for $p > 1/2$ we have

$$[N_H(d, n, t_1, t_2)/p] \leq N_H(d, n, t_2),$$

and for $p < 1/2$,

$$[N_H(d, n, t_1, t_2)/p] > N_H(d, n, t_2).$$

For $p = 1/2$, we compare the coefficient of the second dominant powers in these two terms and get that $\sum_{k=m-1}^{m+2} \binom{d}{k} < 2 \cdot \sum_{k=m-1}^{m+1} \binom{d}{k}$. Thus, we conclude that for this case $[N_H(d, n, t_1, t_2)/p] > N_H(d, n, t_2)$.

The following lemma compares between the values of $N'_H(d, t_1, p)$ and $\min\{[N_H(d, n, t_1, t_2)/p], N(d, t_2)\}$. Recall that according to Proposition 23, for $0 < p \leq 1/2$, $N'_H(d, t_1, p) \in \{0, \infty\}$, and by Lemma 32 if $1/2 < p < 1$ then $N_H(d, n, t_1, t_2)/p \leq N_H(d, n, t_2)$. Thus, in Lemma 33 we compare only between $[N_H(d, n, t_1, t_2)/p]$ and $\left\lfloor \frac{N_H(d, n, t_1)}{2p-1} \right\rfloor$ for $1/2 < p < 1$.

**Lemma 33.** For any fixed $p$ and $n$ sufficiently large the following holds. If $d$ is even, $t_2 = t_1 + 1$, and $(1/2 < p \leq 2/3) \text{ or } (2/3 < p < 3/4 \text{ and } d < \frac{2 - 2p}{3p - 2})$, then

$$\left\lfloor \frac{N_H(d, n, t_1)}{2p-1} \right\rfloor > [N_H(d, n, t_1, t_2)/p].$$

Otherwise,

$$\left\lfloor \frac{N_H(d, n, t_1)}{2p-1} \right\rfloor \leq [N_H(d, n, t_1, t_2)/p].$$

**Proof.** Note that $N_H(d, n, t_1) = \Theta(n^{\frac{2t_1-d}{2}})$ and $N_H(d, n, t_1, t_2) = \Theta(n^{\frac{t_1+t_2-d}{2}})$. Thus, we compare the powers $\left\lfloor \frac{2t_1-d}{2} \right\rfloor$ and $\left\lfloor \frac{t_1+t_2-d}{2} \right\rfloor$. If $t_2 = t_1 + 1$ and $d$ is even then $\frac{2t_1-d}{2} = \left\lfloor \frac{t_1+t_2-d}{2} \right\rfloor$. In all other cases, $\frac{2t_1-d}{2} < \left\lfloor \frac{t_1+t_2-d}{2} \right\rfloor$, and hence, $N_H(d, n, t_1) < [N_H(d, n, t_1, t_2)/p]$.

For the case of $t_2 = t_1 + 1$ and even $d$, we compare the coefficients of the
dominant powers.

\[ N_H(d, n, t_1) = \binom{d}{d/2} \cdot \binom{n-d}{t_1-d/2} + \Theta(n^{t_1-d/2-1}), \]
\[ N_H(d, n, t_1, t_2) = \left(\binom{d}{d/2-1} + \binom{d}{d/2}\right) \cdot \binom{n-d}{t_1-d/2} + \Theta(n^{t_1-d/2-1}). \]

Thus, the coefficient of the dominant term in \( \left\lfloor \frac{N_H(d, n, t_1)}{2p-1} \right\rfloor \) is

\[ \frac{1}{2p-1} \binom{d}{d/2}, \]

while the corresponding coefficient in \( \left\lfloor \frac{N_H(d, n, t_1, t_2)}{p} \right\rfloor \) is

\[ \frac{1}{p} \left( \binom{d}{d/2} + \binom{d}{d/2-1} \right) = \frac{2d+2}{(d+2)p} \binom{d}{d/2}. \]

The inequality

\[ \frac{2d+2}{(d+2)p} < \frac{1}{2p-1} \]

holds if and only if

\( (p \leq 2/3) \) or \( (2/3 < p < 3/4 \) and \( d < \frac{2-2p}{3p-2} \). \)

Therefore, we conclude that

\[ \left\lfloor \frac{N_H(d, n, t_1, t_2)}{p} \right\rfloor < \left\lfloor \frac{N_H(d, n, t_1)}{2p-1} \right\rfloor \]

if and only if \( d \) is even, \( t_2 = t_1 + 1 \), and \( (1/2 < p \leq 2/3) \) or \( (2/3 < p < 3/4 \) and \( d < \frac{2-2p}{3p-2} \)).

According to Corollary 3, Lemma 32, and Lemma 33, we can now summarize the results for the case of binary substitution errors.

**Corollary 4.** For any fixed \( p \) and \( n \) sufficiently large the following holds.

- For \( 0 < p \leq 1/2 \):

\[ N_H^w(T, P, d, n) = \begin{cases} 
1 & \text{if } d > 2t_1, \\
\Theta(n^{t_1+t_2-d/2}) & \text{otherwise.}
\end{cases} \]
• For $1/2 < p < 1$:

$$N_H(T, P, d, n) = \Theta(n^{\left\lfloor \frac{2t_1 - d}{2} \right\rfloor}).$$

More specifically,

• For $0 < p \leq 1/2$:

$$N_H(T, P, d, n) = \begin{cases} 
1 & \text{if } d > 2t_1, \\
N_H(d, n, t_2) + 1 & \text{otherwise,} \\
\left\lceil \frac{N_H(d, n, t_2)}{p} \right\rceil + 1 & \text{if } d \text{ is odd} \\
& \text{and } t_2 = t_1 + 1, \\
\left\lceil \frac{N_H(d, n, t_1, t_2)}{p} \right\rceil + 1 & \text{otherwise.}
\end{cases}$$

• For $1/2 < p < 1$:

$$N_H(T, P, d, n) = \begin{cases} 
\left\lceil \frac{N_H(d, n, t_1, t_2)}{p} \right\rceil + 1 & \text{if } d \text{ is even, } t_2 = t_1 + 1, \\
& \text{and } \left( \frac{1}{2} < p \leq \frac{2}{3} \right) \lor \\
& \left( \frac{2}{3} < p < \frac{3}{4} \land d < \frac{2-2p}{3p-2} \right), \\
N_H(d, t_1, p) + 1 & \text{otherwise.}
\end{cases}$$

To understand the results for substitution errors better, we demonstrate some of them. Let $L_1 = N_H(d, n, t_1) + 1$ and $L_2 = N_H(d, n, t_2) + 1$ be the solutions for the cases where all the channels are identical and cause at most $t_1$ and $t_2$ errors, respectively. In addition, for $T = (t_1, t_2)$ and $P = (p, 1)$, we denote $L_u = N_H(T, Pd, n)$ and $L^k = N_H(T, P, d, n)$. That is, $L_u$ and $L^k$ are the solutions to Problem 1 with $\ell = 2$ for the non-sequenced and for the sequenced models, respectively. Clearly, $L_1 \leq L^k \leq L_u \leq L_2$.

The following examples compare between the four values, $L_1$, $L_2$, $L^k$, and $L_u$. These examples emphasize the benefit produced from knowing that a fraction $p$ of the channels are $t_1$-error, and not $t_2$-error, where $t_1 < t_2$. Additionally, these examples highlight the advantage of the sequenced model on the non-sequenced one.

• For fixed $p$, $0 < p \leq 1/2$, $d = 1$, and $T = (2, 4)$, we have $L_1 = L^k = \Theta(n)$ and $L_u = \Theta(n^2)$, while $L_2 = \Theta(n^3)$,
• For fixed $p$, $0 < p \leq 1/2$, $d = 1$, and $T = (2, 8)$, we have $L_1 = L^k = \Theta(n)$ and $L^u = \Theta(n^4)$, while $L_2 = \Theta(n^7)$,

• For fixed $p$, $1/2 < p \leq 2/3$, $d = 2$, and $T = (4, 5)$, we have $L_1 = L^k = L^u = \Theta(n^3)$, while $L_2 = \Theta(n^4)$.

For $T = (t_1, t_2)$, where $t_2 > t_1$, and a fixed $p$, $0 < p < 1$, the cases of $d = 2t_1 - 1$ or $d = 2t_1$ are interesting, since by Lemma 30, in both cases $N_H(d, n, t_1) = \binom{2t_1}{t_1}$, and thus $L_1$ is independent on $n$. For these parameters, we have $L_1 = \binom{2t_1}{t_1} + 1$, $L^k = \left\lceil \frac{2t_1}{t_1} / p \right\rceil + 1$, $L^u = \Theta(1)$ for $p > 1/2$, $L^u = \Theta(n^{\frac{t_1 + t_2 - d}{2}})$ for $p \leq 1/2$, and $L_2 = \Theta(n^{\frac{2t_2 - d}{2}})$.

### 6.A.4.2 Transposition Errors

For $x \in \{0, 1\}^n$, a transposition error transposes the symbols $x_i$ and $x_j$ for some $i, j \in [N]$, $i \neq j$. Note that transpositions do not change the Hamming weight of a word. Therefore we consider $V_J = J_w^n$, the set of all length $n$ words over the binary alphabet with Hamming weight $w$. The Johnson distance function $\rho_J : V_J \times V_J \to \mathbb{N}$ is defined by $\rho_J(x, z) = \frac{|\{i : x_i \neq z_i\}|}{2}$.

Note, that for all $x, z \in V_J$, $N_J(x, z, t_1, t_2)$ and $N_J(x, z, t)$ depend only on $d = \rho_J(x, z)$. Thus, for $x, z \in V_J$ such that $d = \rho_J(x, z)$, we denote by $N_J(d, n, t_1, t_2)$ and $N_J(d, n, t)$ the values $N_J(x, z, t_1, t_2)$ and $N_J(x, z, t)$, respectively. Note that all these values depend also on $w$ which is omitted to simplify the notations. The monotonicity by intersection property holds also for this case, as described in the following proposition. The proof is omitted since it essentially the same as the proof of Proposition 28 for substitution errors.

**Proposition 29.** Let $U = \{x, z\} \subseteq V_H$ such that $\rho_J(x, z) = d$. Then, $N^0_J(T, P, d, V) = N^0_J(T, P, U, V)$.

According to Proposition 29, in order to calculate the value of $N^0_J(T, P, d, n)$ it is enough to find the value of $N^0_J(T, P, \{x, z\}, V_J)$, for some $x, z$ such that $\rho_J(x, z) = d$. Therefore, according to Theorem 21 and Theorem 22, for $T = (t_1, t_2)$ and $P = (p, 1)$, we conclude that

$$N^k_J(T, P, d, n) = \min\{\lfloor N_J(d, n, t_1)/p \rfloor, N_J(d, n, t_2)\} + 1,$$

(6.5)
and

\[ N_j^T(T, P, d, n) = \min\{ \lceil N_j(d, n, t_1, t_2)/p \rceil, N_j(d, n, t_2) \} + N_j'(d, t_1, p), \]

where \( N_j'(d, t_1, p) = \min\{ L : 2 \left\lfloor pL \right\rfloor - L > N_j(d, n, t_1), L \geq 1 \} - 1. \)

The following lemma was shown in [8].

**Lemma 34.** For \( t, d \geq 1, \)

\[ N_j(d, n, t) = \sum_{i=0}^{t} \binom{n - w - d}{i} \cdot \sum_{a=0}^{t-i} \sum_{b=0}^{t-i} \binom{d}{a} \binom{d}{b} \binom{w - d}{a + b + i - d}, \]

where \( \binom{a}{b} = 0 \) if \( a < b \) or \( b < 0. \)

Thus, as in the substitution errors case, we conclude that the solution for the sequenced model is directly deduced from (6.5) together with the result from Lemma 34. We also note that for a fixed \( p \) and \( n \) sufficiently large, \( N_j^T(T, P, d, n) = \lceil N_j(d, n, t_1)/p \rceil. \) At the end of this subsection, we provide some examples for this value.

For the non-sequenced model, we compute the value of \( N_j(d, n, t_1, t_2) \) for all \( t_1, t_2. \) This value is presented in the following lemma, which generalizes Lemma 34, and is proved by similar combinatorial computation. The value \( N_j(d, n, t) \), which is presented in Lemma 34, can be obtained from Lemma 35, by substituting \( t = t_1 = t_2. \) For the completeness of the results in the paper we prove the following lemma in Appendix B.

**Lemma 35.** For \( t_1 \leq t_2: \)

\[ N_j(d, n, t_1, t_2) = \sum_{i=0}^{t_1} \binom{n - w - d}{i} \cdot \sum_{a=0}^{t_1-i} \sum_{b=0}^{t_1-i} \binom{d}{a} \binom{d}{b} \binom{w - d}{a + b + i - d}. \]

The following two lemmas compare between the three components which determine the value of \( N_j^u(T, P, d, n) \), for \( d \geq 1, t_1 < t_2 \in \mathbb{N}, \) and fixed \( 0 < p < 1. \) Lemma 36 compares between \( \lceil N_j(d, n, t_1, t_2)/p \rceil \) and \( N_j(d, n, t_2). \)
Lemma 36. For any fixed $p$ and $n$ sufficiently large

$$N_J(d, n, t_2) > \lfloor N_J(d, n, t_1, t_2)/p \rfloor.$$ 

Proof. Note that $N_J(d, n, t_2) = \Theta(n^{12})$ and $N_J(d, n, t_1, t_2) = \Theta(n^{11})$. Thus, for $t_2 > t_1$, $N_J(d, n, t_2) > \lfloor N_J(d, n, t_1, t_2)/p \rfloor$. □

The following lemma compares between the values of $N'_J(d, t_1, p)$ and $\min\{\lfloor N_J(d, n, t_1, t_2)/p \rfloor, N_J(d, n, t_2)\}$. Recall that according to Proposition 23, for $0 < p \leq 1/2$, $N'_J(d, t_1, p) \in \{0, \infty\}$, and by Lemma 36 if $1/2 < p < 1$ then $\lfloor N_J(d, n, t_1, t_2)/p \rfloor \leq N_J(d, n, t_2)$. Thus, in Lemma 37 we compare only between $\lfloor N_J(d, n, t_1, t_2)/p \rfloor$ and $\lfloor N_J(d, n, t_1, t_2)/p \rfloor$ for $1/2 < p < 1$.

Lemma 37. For any fixed $p$ and $n$ sufficiently large the following holds.

$$\lfloor N_J(d, n, t_1)/2p - 1 \rfloor < \lfloor N_J(d, n, t_1, t_2)/p \rfloor$$

if and only if

$$\frac{p}{2p-1} < \sum_{b=0}^{t_2-t_1} \binom{d}{b} \frac{1}{(t_1-d+b)^2 \cdot (t_1-d+1)^2}.$$ 

Proof. Note that

$$\frac{N_J(d, n, t_1, t_2)}{2p-1} = \frac{1}{2p-1} \cdot \binom{w-d}{t_1-d} \cdot \binom{n-w-d}{t_1} + \Theta(n^{t_1-1}),$$

and

$$\frac{N_J(d, n, t_1, t_2)}{p} = \frac{1}{p} \cdot \sum_{b=0}^{t_2-t_1} \binom{d}{b} \binom{w-d}{t_1+b-d} \cdot \binom{n-w-d}{t_1} + \Theta(n^{t_1-1}).$$

Thus, we compare between

$$\frac{1}{2p-1} \cdot \binom{w-d}{t_1-d}$$

and

$$\frac{1}{p} \cdot \sum_{b=0}^{t_2-t_1} \binom{d}{b} \binom{w-d}{t_1+b-d}.$$ 

The following holds

$$\frac{1}{2p-1} \cdot \binom{w-d}{t_1-d} < \frac{1}{p} \cdot \sum_{b=0}^{t_2-t_1} \binom{d}{b} \binom{w-d}{t_1+b-d}$$

if and only if

$$\frac{p}{2p-1} < \sum_{b=0}^{t_2-t_1} \binom{d}{b} \frac{\binom{w-d}{t_1+b-d}}{t_1-d} \frac{1}{(t_1-d+b)^2 \cdot (t_1-d+1)^2}.$$ 

□
According to Corollary 3, Lemma 36, and Lemma 37, we can now summarize the results for transposition errors.

**Corollary 5.** For any fixed $p$ and $n$ sufficiently large the following holds. If $0 < p \leq 1/2$ and $d > 2t_1$ then $N^u_j(T,P,d,n) = 1$. Otherwise, $N^u_j(T,P,d,n) = \Theta(n^{t_1})$ More specifically,

- For $0 < p \leq 1/2$:
  \[
  N^u_j(T,P,d,n) = \begin{cases} 
  1 & \text{if } d > 2t_1, \\
  \lfloor N_j(d,n,t_1,t_2)/p \rfloor + 1 & \text{otherwise.}
  \end{cases}
  \]

- For $1/2 < p < 1$:
  \[
  N^u_j(T,P,d,n) = \min\{\left\lfloor \frac{N_j(d,n,t_1,t_2)}{p} \right\rfloor, N_j'(d,t_1,p)\} + 1 = \Theta(n^{t_1}).
  \]

To understand the results for transposition errors better, we demonstrate some of them. Let $L_1 = N_j(d,n,t_1) + 1$ and $L_2 = N_j(d,n,t_2) + 1$ be the solutions for the cases where all the channels are identical and cause at most $t_1$ and $t_2$ errors, respectively. Additionally, for $T = (t_1,t_2)$ and $P = (p,1)$, we denote $L^u = N^u_j(T,P,d,n)$ and $L^k = N^k_j(T,P,d,n)$. That is, $L^u$ and $L^k$ are the solutions to Problem 1 with $\ell = 2$ for the non-sequenced and for the sequenced models, respectively. Clearly, $L_1 \leq L^k \leq L^u \leq L_2$.

The following examples compare between the four values, $L_1$, $L_2$, $L^k$, and $L^u$. These examples emphasize the benefit produced from knowing that a fraction $p$ of the channels are $t_1$-error, and not $t_2$-error, where $t_1 < t_2$. Note that for the Johnson distance function we have, $\Theta(L^k) = \Theta(L^u)$.

- For fixed $p$, $d = 1$, and $T = (2,4)$:
  \[
  L_1 = L^k = L^u = \Theta(n^2), \text{ while } L_2 = \Theta(n^4),
  \]

- For fixed $p$, $d = 1$, and $T = (2,8)$:
  \[
  L_1 = L^k = L^u = \Theta(n^2), \text{ while } L_2 = \Theta(n^8),
  \]

- For fixed $p$, $d = 2$, and $T = (4,5)$:
  \[
  L_1 = L^k = L^u = \Theta(n^4), \text{ while } L_2 = \Theta(n^5).
  \]
6.A.5 Problem 2 - Minimum Distance

In this section we solve Problem 2. That is, we assume that the size of the system, \( N \), is given, and we study the minimum distance of the code, \( d \), that is required for exact reconstruction. Let \( t_1 \leq t_2 \leq \cdots \leq t_N \) be \( N \) positive integers that denote the maximum number of errors in the channels. In a similar way to Problem 1, we consider the following three cases:

1. a \((T,N)\)-sequenced-channel system, where the \( N \)-tuple \( T = (t_1,t_2,\ldots,t_N) \) is known,
2. a \((T,N)\)-non-sequenced-channel system, where the multiset \( T = \{t_1,t_2,\ldots,t_N\} \) is known, and
3. a \((t,N)\)-channel system, where \( t = \sum_{i=1}^{N} t_i \) is known.

The solution to Problem 2 is given by reduction to Problem 1. Let \( T = (t_1,\ldots,t_N) \) where \( t_1 \leq t_2 \leq \cdots \leq t_N \in \mathbb{N} \), \( T = (t_1,\ldots,t_N) \) denotes the multiset \( T = \{t_1,t_2,\ldots,t_N\} \) for the non-sequenced model, or the \( N \)-tuple, \( T = (t_1,t_2,\ldots,t_N) \), in the sequenced case, where the exact meaning will be clear from the context. We define \( T' = (t'_1,t'_2,\ldots,t'_\ell) \) which consists of the set of \( T \), where \( t_1 = t'_1 < t_2 < \cdots < t_\ell = t_N \). We define the function \( f : [N] \to [\ell] \) such that \( f(i) = i' \) if \( t_i = t'_i \), and the function \( g : [\ell] \to [N] \) is defined as follows: \( g(i') = \max\{i : f(i) = i'\} \). We also define \( P = (p_1,p_2,\ldots,p_\ell) \), such that for \( i' \in [\ell] \), \( p_j = \frac{g(i')}{N} \).

The following theorem establishes the connection between a solution to Problem 1 and Problem 2, by using the definition of \( T' \) and \( P \). We remind that \( D^k(T,N,V) \), \( D^u(T,N,V) \), and \( D^a(t,N,V) \) are the minimum distances of the codes which are required for exact reconstruction in the sequenced, non-sequenced, and average models, respectively. Clearly, by the definitions of the models, \( D^k(T,N,V) \leq D^u(T,N,V) \leq D^a(t,N,V) \).

**Theorem 30.** For all \( N \geq 1 \) the following properties hold:

- for the average model,
  \[
  \lceil 2t \rceil \leq D^a(t,N,V) \leq \lceil 2t \rceil + 1,
  \]
  and if \( t \) is integer then \( D^a(t,N,V) = 2t + 1 \),

155
• for the non-sequenced model,

\[ D^u(T, N, V) = \min \{ d : A(T', P)-\text{non-sequenced-channel} \]

\[ \text{system of size } N \]

\[ \text{supports exact reconstruction} \]

\[ \text{for all } U \subseteq V \text{ such that } d(U) \geq d \}, \]

• and for the sequenced model,

\[ D^k(T, N, V) = \min \{ d : N^k(T', P, d, V) \leq N \}. \]

**Proof.** The solutions for all cases are derived immediately by reduction to Problem 1. For the non-sequenced case we could not find a more explicit solution, since \( N > N^u(T', P, U, V) \) does not guarantee an exact reconstruction as explained in Remark 1. However for the average model we apply Theorem 27 to find an explicit solution, and for the sequenced model we could simplify the solution to Problem 2 by the property that a \((T', P)\)-sequenced-channel system of size \( N \) supports exact reconstruction for \( U \subseteq V \) if and only if \( N \geq N^k(T', P, U, V) \).

In more details, for the average model, by the reduction to Problem 1, it can be readily verified that

\[ D^a(t, N, V) = \min \{ d : \text{a } t\text{-channel system of size } N \]

\[ \text{supports exact reconstruction for all } x, z \]

\[ \text{such that } \rho(x, z) \geq d \}. \]

Using Theorem 27 we conclude the value of \( D^a(t, N, V) \).

The sequenced model is derived similarly. By the reduction to Problem 1 and by \((T', P)\) definition, we have

\[ D^k(T, N, V) = \min \{ d : A(T', P)-\text{sequenced-channel} \]

\[ \text{system of size } N \]

\[ \text{supports exact reconstruction} \]

\[ \text{for all } U \subseteq V \text{ such that } d(U) \geq d \}, \]

and since a \((T', P)\)-sequenced-channel system of size \( N \) supports exact re-
construction for $U$ if and only if $N \geq N^k(T', P, U, V)$, we have

$$D^k(T, N, V) = \min\{d : N^k(T', P, d, V) \leq N\}.$$ 

Yet, for the non-sequenced model, by Remark 1, a $(T', P)$-non-sequenced-channel system of size $N$, $N > N^k(T', P, U, V)$, might not support exact reconstruction for $U$. Therefore, in this model, unlike in the sequenced case, we could not simplify the result which obtained by the reduction to Problem 1.

For the average model, Theorem 30 provides an explicit solution to Problem 2, that is, $D^a(t, N, V)$ has an explicit expression. But, for the other two models, even though Theorem 30 provides a solution to Problem 2, it does not give an explicit expression for the minimum distance. Thus, in the rest of this section we will show how to derive an explicit solution to Problem 2 in these two models for both substitution and transposition errors. The technique which will be presented can be used also for other distance functions.

We first prove Lemma 38, which provides an insight about the reduction to Problem 1 for the general case. Then, we apply this lemma to find an explicit solution, for transpositions in Theorem 31, and for substitutions in Theorem 32.

Let $T$, $N$, $T'$ and $P$ be as defined earlier in this section, where the $(T, N)$-channel system is given, and $(T', P)$ is defined by the reduction. In Subsection 6.A.3.2, we presented conditions for supporting exact reconstruction for $U$ in a $(T', P)$-channel system of size $N$. The following lemma provides equivalent conditions on the given parameters $T$ and $N$. For the sequenced model, by Theorem 24 and Proposition 20, a $(T', P)$-sequenced-channel system of size $N$ supports exact reconstruction for $U = \{x, z\}$ if and only if there exists $i' \in [\ell]$ such that $N(x, z, t_{i'}') < \lceil p_{i'} N \rceil$. Similarly, in the non-sequenced model, by Lemma 25, a $(T', P)$-non-sequenced-channel system of size $N$ supports exact reconstruction for $U = \{x, z\}$ if and only if there exist $i', j' \in [\ell]$ such that $N(x, z, t_{i'}', t_{j'}') < \lceil p_{i'} N \rceil + \lceil p_{j'} N \rceil - N$. The following lemma, establishes an equivalent condition on $T$ and $N$.

**Lemma 38.** Let $T = (t_1, \ldots, t_N)$ such that $t_1 \leq t_2 \leq \cdots, t_N \in \mathbb{N}$, and let $T' = (t_1', \ldots, t_{\ell}')$, $f$, $g$, and $P = (p_1, \ldots, p_{\ell})$ as defined earlier in this section. Then, the followings hold:
• there exists \( i' \in [\ell] \) such that \( N(x, z, t'_i) < [p_{\ell}N] \) if and only if there exists \( i \in [N] \) such that \( N(x, z, t_i) < i \), and \( N(x, z, t'_i) = N(x, z, t_i) \),

• there exist \( i', j' \in [\ell] \) such that \( N(x, z, t'_i, t'_j) < [p_{\ell}N] + [p_{j'}N] - N \) if and only if there exist \( i, j \in [N] \) such that \( N(x, z, t_i, t_j) < i + j - N \), and \( N(x, z, t'_i, t'_j) = N(x, z, t_i, t_j) \).

**Proof.** For the easier direction, assume that there exists \( i' \in [\ell] \) such that \( N(x, z, t'_i) < [p_{\ell}N] \). Let \( i = g(i') \in [N] \), then, by the definitions of \( f, g \), and \( P \) we have: \( N(x, z, t_{g(i')}) < \left[ \frac{g(i')}{N} \cdot N \right] \), that is, \( N(x, z, t_i) < i \). Similarly, if there exist \( i', j' \in [\ell] \), such that \( N(x, z, t'_i, t'_j) < [p_{\ell}N] + [p_{j'}N] - N \), then we can conclude that for \( i = g(i') \) and \( j = g(j') \) \( N(x, z, t_i, t_j) < i + j - N \).

For the second direction, assume that \( N(x, z, t_i) < i \), and let \( r = \max \{ k : f(k) = f(i) \} \), and \( i' = f(i) = f(r) \). That is, \( t_r = t_i = t'_i \), \( i \leq r, r = g(i') \), and \( p_{\ell} = \frac{r}{N} \). Thus we conclude,

\[
N(x, z, t'_i) = N(x, z, t_i) < i \leq r = p_{\ell}N = [p_{\ell}N].
\]

The second part in the direction is proved similarly. We assume that \( N(x, z, t_i, t_j) < i + j - N \), and we define \( r_i = \max \{ k : f(k) = f(i') \} \), \( r_j = \max \{ k : f(k) = f(j) \} \), \( i' = f(i) = f(r_i) \), and \( j' = f(i) = f(r_j) \). That is, \( t_{r_i} = t_i = t'_i \), \( i \leq r_i, r_i = g(i') \), and \( p_{\ell} = \frac{r_i}{N} \), and the same holds for \( j \). Thus we conclude,

\[
N(x, z, t'_i, t'_j) = N(x, z, t_i, t_j) < i + j - N \\
\leq r_i + r_j - N \\
= [p_{\ell}N] + [p_{j'}N] - N.
\]

Now, we focus on substitution and transposition errors. As mentioned in Section 6.A.4, for each \( n \in \mathbb{N} \) and \( x, z \in \{0, 1\}^n \), \( N_H(x, z, t, t') \) and \( N_J(x, z, t, t') \) depend on \( d = \rho(x, z) \) and \( n \), and do not depend on the specific words \( x \) and \( z \). Thus, in Section 6.A.4 we denoted the values of \( N_H(x, z, t, t') \) and \( N_J(x, z, t, t') \) by \( N_H(d, n, t, t') \) and \( N_J(d, n, t, t') \) for the Hamming and for the Johnson cases, respectively. We also use Propositions 28 and 29 for substitutions and transpositions, respectively. These
propositions allow us to replace between $N_H(x, z, t, t')$, $N_J(x, z, t, t')$ and $N(d, n, t, t')$, $N(d, n, t, t')$ where $x, z \in \{0, 1\}^n$ and $\rho(x, z) = d$.

In addition, by Lemma 35, for all $d$, $t$, and $t'$, the following holds $N_J(d, n, t, t') \to \infty$ as $n \to \infty$, or $N_J(d, n, t, t') \equiv 0$ (i.e., $N_J(d, n, t, t') = 0$ for all $n \in \mathbb{N}$). For the Hamming distance function, by Lemma 31, we have the same property for almost all the parameters $d$, $t$, and $t'$.

In general we use the notation $N(d, n, t, t')$ (without the subscript $J$ or $H$), and if $N(d, n, t, t')$ holds the property described in this paragraph, we say that $N(d, n, t, t')$ increases on $n$ for $d$, $t$, and $t'$.

In the sequel, we apply Lemma 38 to find an explicit solution to Problem 2 for the sequenced and for the non-sequenced models, for both substitution and transposition errors over the binary alphabet. We use the property of $N(d, n, t, t')$ to simplify the solution. The same technique can be applied to find an explicit solution for other distance functions in which $N(d, n, t, t')$ increases on $n$.

We start with Theorem 31 for transposition errors, since this distance function holds the property of $N(d, n, t, t')$ for all parameters $d$, $t$, and $t'$, that is for the Johnson distance function, $N_J(d, n, t, t')$ increases on $n$ for all parameters $d$, $t$, and $t'$.

**Theorem 31.** Let $V_J = J^w_0$, $\rho_J$ is the Johnson distance function, $N$ is a positive constant number, and $T = (t_1, \ldots, t_N)$ where $t_1 \leq \cdots \leq t_N \in \mathbb{N}$. Then, for $n$ sufficiently large,

$$D^k_J(T, N, V_J) = 2t_1 + 1$$

and

$$D^n_J(T, N, V_J) = \min\{t_i + t_j + 1 : i, j \in [N], i + j = N + 1\}.$$  

**Proof.** For the Johnson distance function, by Lemma 35, $N_J(d, n, t, t')$ increases on $n$, for all $t$, $t'$ and $d$. That is, $N_J(d, n, t, t') = 0$ for all $n$, or $N_J(d, n, t, t') \to \infty$ as $n \to \infty$. Let $T' = (t'_1, t'_2, \ldots, t'_\ell)$ and $P = (p_1, p_2, \ldots, p_\ell)$ be defined as described earlier in this section.

Let $U = \{x, z\}$ be of distance $d$. Thus, by Theorem 24 and Proposition 20, a $(T', P)$-sequenced-channel system of size $N$ supports exact reconstruction for $U = \{x, z\}$ of distance $d$ if and only if there exists $i' \in [\ell]$ such that $N_J(d, n, t'_i) = N_J(x, z, t'_i) < [p_{i'}N]$, and by Lemma 38 this holds if and only if there exists $i \in [N]$ such that $N_J(d, n, t_i) < i$. But,
$N_J(d, n, t_i) = N_J(d, n, t_i, t_i)$ increases on $n$, and therefore, for $n$ sufficiently large, $N_J(d, n, t_i) < i$ implies $N_J(d, n, t_i) = 0$ which is equivalent to $d > 2t_i$. Recall that $D^k_J(T, N, V_H)$ equals to the minimum between all these possible $d$. Thus, we can conclude that in the sequenced model, for $n$ sufficiently large,

$$D^k_J(T, N) = \min\{2t_i + 1 : i \in [N]\} = 2t_1 + 1.$$

Similarly, in the non-sequenced model, by Lemma 25, a $(T', P)$-non-sequenced-channel system of size $N$ supports exact reconstruction for $U = \{x, z\}$ of distance $d$ if and only if there exist $i', j' \in [\ell]$ such that $N_J(d, n, t_{i'}, t_{j'}) = N_J(x, z, t_{i'}, t_{j'}) < \lceil p_{i'} N \rceil + \lceil p_{j'} N \rceil - N$, and by Lemma 38 this holds if and only if there exist $i, j \in [N]$ such that $N_J(d, n, t_i, t_j) < i + j - N$. But, $N_J(d, n, t_i, t_j)$ increases on $n$, and therefore, for $n$ sufficiently large, $N_J(d, n, t_i, t_j) < i + j - N$ implies $N_J(d, n, t_i, t_j) = 0$ and $i + j - N > 0$, where the condition $N_J(d, n, t_i, t_j) = 0$ is equivalent to $d > t_i + t_j$. Recall that $D^3_J(T, N, V_H)$ equals to the minimum $d$ required for exact reconstruction. Thus, we can conclude that in the non-sequenced model, for $n$ sufficiently large,

$$D^3_J(T, N) = \min\{t_i + t_j + 1 : i, j \in [N], i + j - N > 0\} = \min\{t_i + t_j + 1 : i, j \in [N], i + j = N + 1\}.$$

The following theorem provides an explicit solution to Problem 2 for the sequenced and for the non-sequenced models for substitution errors, by applying Theorem 30, and Lemma 38. Note that in this case $N_H(d, n, t, t')$ increases on $n$ for almost all the parameters $d$, $t$, and $t'$, and we use this property in the proof.

**Theorem 32.** Let $V_H = \{0, 1\}^n$, $\rho_H$ is the Hamming distance function, $N$ is a positive constant number, and $T = (t_1, \ldots, t_N)$ where $t_1 \leq \cdots \leq t_N \in \mathbb{N}$. Then, for $n$ sufficiently large,

$$D^k_H(T, N, V_H) = \min\{D_1, D_2, D_3\}$$
where

\[ D_1 = 2t_1 + 1, \]
\[ D_2 = \min \left\{ 2t_i : i \in [N], \left( \frac{2t_i}{t_i} \right) < i \right\}, \text{ and } \]
\[ D_3 = \min \left\{ 2t_i - 1 : i \in [N], \left( \frac{2t_i}{t_i - 1} \right) < i \right\}, \]

and

\[ D^u(T, N, V_H) = \min \{ D_1, D_2, D_3 \}, \]

where

\[ D_1 = \min \{ t_i + t_j + 1 : i, j \in [N], i + j = N + 1 \}, \]
\[ D_2 = \min \left\{ t_i + t_j : i \leq j \in [N], \left( \frac{t_i + t_j}{t_i} \right) < i \right\}, \text{ and } \]
\[ D_3 = \min \left\{ t_i + t_j - 1 : i \leq j \in [N], \left( \frac{t_i + t_j}{t_i - 1} \right) < i \right\}. \]

Proof. By Lemma 31 \( N_H(d, n, t, t') \) increases on \( n \) for all \( t, t' \) and \( d \), except for two cases: \( d = t + t' \) and \( d = t + t' - 1 \). If \( d = t + t' \) then \( N_H(d, n, t, t') = \left( \frac{t + t'}{t} \right) \), and for \( d = t + t' - 1 \) we have \( N_H(d, n, t, t') = \left( \frac{t + t' - 1}{t} \right) \).

Let \( T' = (t'_1, t'_2, \ldots, t'_{\ell}) \) and \( P = (p_1, p_2, \ldots, p_\ell) \) be defined as described earlier in this section.

Let \( U = \{ x, z \} \) be of distance \( d \). Thus, by Theorem 24 and Proposition 20, a \( (T', P) \)-sequenced-channel system of size \( N \) supports exact reconstruction for \( U = \{ x, z \} \) of distance \( d \) if and only if there exists \( i' \in [\ell] \) such that \( N_H(d, n, t_i') = N_H(x, z, t_i') < \lfloor p_{i'} N \rfloor \), and by Lemma 38 this holds if and only if there exists \( i \in [N] \) such that \( N_H(d, n, t_i) < i \). Next we present equivalent conditions to \( N_H(d, n, t_i) < i \) for all \( i \). If \( d = 2t_i - 1 \) then we substitute \( N_H(d, n, t_i) = \left( \frac{2t_i}{t_i} \right) \), and if \( d = 2t_i \) then \( N_H(d, n, t_i) = \left( \frac{2t_i}{t_i} \right) + \left( \frac{2t_i}{t_i-1} \right) \).

Otherwise, \( N_H(d, n, t_i) = N_H(d, n, t_i, t_i) \) increases on \( n \), and therefore, for \( n \) sufficiently large, \( N_H(d, n, t_i) < i \) implies \( N_H(d, n, t_i) = 0 \) which is equivalent to \( d > 2t_i \). Recall that \( D^k_H(T, N, V_H) \) equals to the minimum between all these possible \( d \). Thus, we can conclude that in the sequenced model, for \( n \) sufficiently large,

\[ D^k_H(T, N, V_H) = \min \{ D_1, D_2, D_3 \} \]

161
Problem 2 for some parameters of $T_n$ sequenced model, for required for exact reconstruction. Thus, we can conclude that in the non-
implies $N$ increases on $n$ and therefore, for all $n$ sufficiently large, $N_H(d, n, t_i, t_j) < i + j - N$ implies $N(d, n, t_i, t_j) = 0$ and $i + j - N > 0$, were $N_H(d, n, t_i, t_j) = 0$ is equivalent to $d > t_i + t_j$. Recall that $D^0_H(T, N, V_H)$ equals to the minimum $d$ required for exact reconstruction. Thus, we can conclude that in the non-
sequence model, for $n$ sufficiently large,

$$D^0_H(T, N, V_H) = \min\{D_1, D_2, D_3\}$$

where

$$D_1 = \min\{t_i + t_j + 1 : i, j \in [N], i + j - N > 0\} = \min\{t_i + t_j + 1 : i, j \in [N], i + j = N + 1\},$$

$$D_2 = \min\left\{t_i + t_j : i \leq j \in [N], \frac{t_i + t_j}{t_i} < i \right\},$$

$$D_3 = \min\left\{t_i + t_j - 1 : i \leq j \in [N], \left(\frac{t_i + t_j}{t_i}\right) + \left(\frac{t_i + t_j}{t_i - 1}\right) < i \right\}.$$
errors in the two models, the sequenced and the non-sequenced. The examples are presented in Table 6.1, where $J$ and $H$ in the left column are abbreviations for Johnson and Hamming, and represent the transposition and substitution cases, respectively. In the left column we write the parameter $T$ and some conditions on it. For example, $(t_1, t_2, t_3), J, H(t_3 > 1)$ means that this row presents the solution for transpositions ($J$) for all $(t_1, t_2, t_3)$, and for substitutions ($H$) for all $(t_1, t_2, t_3)$ such that $t_3 > 1$. The parameter $t$ denotes the average of $T$, that is, $t = \frac{\sum_{k=1}^{N} t_k}{N}$.

<table>
<thead>
<tr>
<th>$T$</th>
<th>$D^n(T, N, V)$</th>
<th>$D^n(T, N, V)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(t_1, t_2)$, $J, H$</td>
<td>$2t_1 + 1$</td>
<td>$t_1 + t_2 + 1$</td>
</tr>
<tr>
<td>$(t_1, t_2, t_3), J, H(t_3 &gt; 1)$</td>
<td>$2t_1 + 1$</td>
<td>$\min{t_1 + t_3, 2t_2} + 1$</td>
</tr>
<tr>
<td>$(1, 1, 1), H$</td>
<td>$2$</td>
<td>$2$</td>
</tr>
<tr>
<td>$(t_1, t_2, t_3, t_4), J, H(t_3 &gt; 1)$</td>
<td>$2t_1 + 1$</td>
<td>$\min{t_1 + t_4, t_2 + t_3} + 1$</td>
</tr>
<tr>
<td>$(1, 1, 1, 1), H$</td>
<td>$1$</td>
<td>$1$</td>
</tr>
<tr>
<td>$(1, 1, 1, t_4), H$</td>
<td>$2$</td>
<td>$2$</td>
</tr>
<tr>
<td>$(t_1, t_2, t_3, t_4, t_5), J, H(t_3 &gt; 1)$</td>
<td>$2t_1 + 1$</td>
<td>$\min{t_1 + t_5, t_2 + t_4, 2t_3} + 1$</td>
</tr>
</tbody>
</table>

Table 6.1: Examples for the solution to Problem 2.

6.6 Special Systems for $T = (t_1, t_2)$

In this section we study special cases of two types of channels. First, we define a new problem, and then present its solution. For $T = (t_1, t_2), t_1 < t_2$, and a constant positive integer $a$, a channel system with $N$ combinatorial channels is called a $(T, i, a)$-channel system, $i \in \{1, 2\}$, if $a$ of the channels are $t_i$-error channels, while the rest are $t_3-i$-error channels. If the size of a system is smaller than $a$, then all the channels are $t_i$-error.

Under this model, we consider both cases, sequenced- and non-
sequenced-channel systems. For $U \subseteq V$, we denote by $N^u(T, i, a, U, V)$ and $N^k(T, i, a, U, V)$ the value of the minimum size of a $(T, i, a)$-non-sequenced- and $(T, i, a)$-sequenced-channel system such that each $x \in U$ has exact reconstruction, respectively. This problem is formulated as follows.

**Problem 3.** Let $V$ be a finite set with some distance function $\rho : V \times V \to \mathbb{N}$, for all $U \subseteq V$ and $i \in \{1, 2\}$, find the values of $N^u(T, i, a, U, V)$ and $N^k(T, i, a, U, V)$.

As before, we focus on sets of the form $U = \{x, z\}$ since $N^g(T, i, a, U, V) = \max\{N^g(T, i, a, \{x, z\}, V) : x, z \in U\}$.

The solution for this problem is presented in the next three theorems. The first theorem solves the problem for constant number of $t_1$-error channels. In this case, the minimum number of channels which are required for exact reconstruction does not depend on knowing the behavior of each channel. The last two theorems present the solutions for constant number of $t_2$-error channels; Theorem 35 for the non-sequenced-channel system, and Theorem 34 for the sequenced one. In the rest of this section we denote $U$ to be $U = \{x, z\} \subseteq V$, $T = (t_1, t_2)$ where $t_1 < t_2$ and $a$ is a constant positive integer.

**Theorem 33.** $N^k(T, 1, a, U, V) = N^u(T, 1, a, U, V) = N + 1$, where

$$N = \begin{cases} N(x, z, t_2) & \text{if } N(x, z, t_1) \geq a, \\ N(x, z, t_1) & \text{otherwise.} \end{cases}$$

Furthermore,

- a $(T, 1, a)$-sequenced-channel system of size $J$ supports exact reconstruction for $U$ for all $J \geq N^k(T, 1, a, U, V)$, and

- a $(T, 1, a)$-non-sequenced-channel system of size $J$ supports exact reconstruction for $U$, for all $N(x, z, t_1) < J \leq a$, and $N(x, z, t_2) < J$.

**Proof.** If $N(x, z, t_1) < a$, then a $(T, 1, a)$-channel system of size at most $N(x, z, t_1) + 1$ contains only $t_1$-channels. Thus, according to Levenshtein [8], $N^k(T, 1, a, U, V) = N^u(T, 1, a, U, V) = N(x, z, t_1) + 1$ Let $J$ be a size of a $(T, 1, a)$-channel system. where $N(x, z, t_1) < a$. Then, in the sequenced model, for all $J \geq N^k(T, 1, a, U, V) = N(x, z, t_1) + 1$, the first
\(N^k(T, 1, a, U, V)\) channels are \(t_1\)-error channels, and only their outputs will be used by the decoder. In the non-sequenced model, if \(N(x, z, t_1) + 1 \leq J \leq a\) then all the channels are \(t_1\)-error, and exact reconstruction is supported. In addition, if \(J > N(x, z, t_2)\) exact reconstruction is supported by Levenshtein result for a system where all the channels are \(t_2\)-error. But, for \(N(x, z, t_1) + 1 \leq a < J \leq N(x, z, t_2)\) exact reconstruction may not be supported as explained in Remark 3.

Otherwise, \(N(x, z, t_1) \geq a\), and then it is clear that a \((T, 1, a)\)-channel system of size \(J, J \geq N(x, z, t_2) + 1\), supports exact reconstruction for the two models. For the second direction, we prove that a \((T, 1, a)\)-channel system of size \(J, J < N(x, z, t_2) + 1\) does not support exact reconstruction in the two models. Without loss of generality, let us assume that \(x\) is transmitted over the system. If the first \(a\) outputs are in \(I(x, z, t_1)\) and all the \(J\) outputs are in \(I(x, z, t_2)\), then \(z\) can also be a possible output of the decoder in both the sequenced and the non-sequenced models.

We note that according to Theorem 33 in almost all cases

\[N^k(T, 1, a, U, V) = N^u(T, 1, a, U, V) = N(x, z, t_2) + 1.\]

The following remark explains the undesirable phenomena in which a \((T, 1, a)\)-non-sequenced-channel system of size \(J > N^u(T, 1, a, U, V)\) does not support exact reconstruction.

**Remark 3.** A \((T, 1, a)\)-non-sequenced-channel system of size \(J > N^u(T, 1, a, U, V)\) may not support exact reconstruction for \(U\). By Theorem 33 it could happen only if \(a < J \leq N(x, z, t_2)\). The reason is that in this model it is not guaranteed which of the \(a\) outputs are the outputs of \(t_1\)-error channels. For example, \(U = \{x, z\} \subseteq \{0, 1\}^n\) where \(\rho_H(x, z) = 3\), and \(t_1 = 1, t_2 = 2\) and \(a = 2\). For these parameters we have \(N(x, z, t_1) = 0 < 2 = a\) and \(N(x, z, t_2) = 6\). Thus, \(N^u(T, 1, a, U, V) = 1\), but a \((T, 1, a)\)-non-sequenced-channel system of size \(J = 4\) does not support exact reconstruction for \(U\). To prove it, we present a set of \(J\) outputs, which any order of them may be an outputs sequence when transmitting either \(x\) or \(z\). We choose \(a = 2\) outputs from \(I(x, z, t_1, t_2)\) and another \(a = 2\) outputs from \(I(z, x, t_1, t_2)\). Thus, the decoder can not distinguish between transmitting \(x\) or \(z\). One can readily verify that in this case, exact reconstruction is not supported also for \(J \in \{5, 6\}\).
In the second case, we have that $i = 2$ and $a$ is the number of channels with at most $t_2$ errors. First, we state the solution to Problem 3 for the sequenced model.

**Theorem 34.** $N^k(T, 2, a, U, V) = N + 1$, where

$$N = \min\{N(x, z, t_1) + a, N(x, z, t_2)\}.$$  

Furthermore, for all $J \geq N^k(T, 2, a, U, V)$, a $(T, 2, a)$-sequenced-channel system of size $J$ supports exact reconstruction for $U$.

**Proof.** For the first direction, we prove that a channel system of size $J$, such that $J > \min\{N(x, z, t_1) + a, N(x, z, t_2)\}$, supports exact reconstruction for $x, z$. If $J > \min\{N(x, z, t_1) + a, N(x, z, t_2)\}$ then either $J > N(x, z, t_1) + a$ or $J > N(x, z, t_2)$. Since $t_1 < t_2$, a $t_1$-channel system is also a $t_2$-channel system. Thus, if $J > N(x, z, t_2)$ we can apply the solution by Levenshtain for a system where all the channels are $t_2$-error. For the second condition, if $J > N(x, z, t_1)+a$ then the first $N(x, z, t_1)+1$ channels are $t_1$-error channels, and we can use only these outputs, since we can apply for this subsystem the Levenshtein’s solution for a system where all the channels are $t_1$-error channels.

For the second direction, we assume that the system is of size $J$ where $J \leq \min\{N(x, z, t_1) + a, N(x, z, t_2)\}$. We present a sequence of $J$ outputs, which can be an outputs sequence when transmitting either $x$ or $z$. The first $(J - a)$ outputs will be in $I(x, z, t_1)$, which is possible by the condition $J \leq N(x, z, t_1) + a$, and the other outputs will be chosen from $I(x, z, t_2)$, which is possible by $J \leq N(x, z, t_2)$. Thus, for this outputs sequence, the decoder can not distinguish between transmitting $x$ or $z$.  

Here, we note again that in almost all cases

$$N^k(T, 2, a, U, V) = N(x, z, t_1) + a + 1.$$  

Lastly, we solve Problem 3 for the non-sequenced model.

**Theorem 35.** $N^u(T, 2, a, U, V) = N + 1$, where

$$N = \min\{N(x, z, t_1, t_2) + a, N(x, z, t_2), N(z, x, t_1, t_2) + a, N(x, z, t_1) + 2a\}.$$  

166
Furthermore, for all $J \geq N^w(T, 2, a, U, V)$, a $(T, 2, a)$-non-sequenced-channel system of size $J$ supports exact reconstruction for $U$.

Proof. The proof is similar to the one of Theorem 22. If a $(T, 2, a)$-channel system consists of $J \geq N + 1$ channels, then, by the definition of $N$, at least one of the following conditions holds:

1. $J - a > N(x, z, t_1, t_2)$,
2. $J - a > N(z, x, t_1, t_2)$,
3. $J > N(x, z, t_2)$, or
4. $2(J - a) - N(x, z, t_1) > J$.

The above conditions are symmetric for $x$ and $z$. Thus, without loss of generality, let $x$ be the transmitted word. If Condition ((1)) or ((3)) holds, since $J - a$ of the outputs are in $B_{t_1}(x)$ and $J$ outputs in $B_{t_2}(x)$, then not all the outputs are in $B_{t_2}(z)$. If Condition ((2)) holds, there are no $J - a$ outputs in $B_{t_1}(z)$. Thus, if one of the conditions ((1)), ((2)), or ((3)) holds, then $z$ will not be decoded. Regarding Condition ((4)), let us denote $m = N(x, z, t_1)$ and assume that we have $m$ outputs in $I(x, z, t_1)$, $m \leq N(x, z, t_1)$. In order for $z$ to be a possible output of the decoder, we must have at least $J - a - m$ outputs in $I(z, x, t_1, t_2) \setminus I(x, z, t_1)$. Furthermore, since $x$ was transmitted at least $J - a - m$ outputs are in $I(x, z, t_1, t_2) \setminus I(x, z, t_1)$. Thus, we must have that $2(J - a) - m \leq J$ in contradiction to Condition ((4)).

For the second direction we have to prove that $N$ channels are not sufficient for exact reconstruction for $U$. The following four conditions hold simultaneously:

1. $N - a \leq N(x, z, t_1, t_2)$,
2. $N - a \leq N(z, x, t_1, t_2)$,
3. $N \leq N(x, z, t_2)$, and
4. $2(N - a) - N(x, z, t_1) \leq N$.

For this part, we present a set of $N$ outputs which any order of them can be an outputs sequence of transmitting either $x$ or $z$. Let $m = N(x, z, t_1)$. If $m < N - a$, then $m$ outputs are in $I(x, z, t_1)$, at least $N - a - m$ in
\( I(x, z, t_1, t_2) \setminus I(x, z, t_1) \) (by Conditions ((1)) and ((4))), at least \( N - a - m \) in \( I(z, x, t_1, t_2) \setminus I(x, z, t_1) \) (by Conditions ((2)) and ((4))), and all the others in \( I(x, z, t_2) \) (by Condition ((3))). Otherwise, \( m \geq N - a \), and then at least \( N - a \) outputs are in \( I(x, z, t_1) \) and \( a \) in \( I(x, z, t_2) \) (by Condition ((3))). Thus, at least \( N - a \) of the outputs are in \( B_{t_1}(x) \), and all the \( N \) outputs are in \( B_{t_2}(x) \), and the same holds for \( z \).

Once again, we note that in almost all the cases

\[ N^u(T, 2, a, U, V) = N(x, z, t_1) + 2a + 1. \]

According to the previous theorem, one can verify that for the Hamming case with \( a = 2, t_1 = 1, t_2 = 2, \) and \( \rho(x, z) = 3 \), we get that \( N^u(T, 2, a, U, V) = 5 \), while if all channels cause at most 2 errors, then the number of channels for exact reconstruction is 7 [8].

Note that Theorem 35 can also be derived by a slight modification in Theorem 22. We denote \( m = N(x, z, t_1) \) and we define here

\[ N'(x, z, t_1, p) = \min\{L : 2 \lceil pL \rceil - L > m, \lceil pL \rceil > m, L \geq 1\} - 1, \]

instead of the previous definition, where

\[ N'(x, z, t_1, p) = \min\{L : 2 \lceil pL \rceil - L > m, L \geq 1\} - 1. \]

This change has no affect on Theorem 22, since for fixed \( p, 0 < p < 1, 2 \lceil pL \rceil - L \leq \lceil pL \rceil \). Then, by substituting \( \lceil pL \rceil = L - a \) in Theorem 22 we can conclude Theorem 35.

6.A.7 Conclusion

In this paper we study a generalization of the reconstruction problem studied by Levenshtein. We assume here that all channels do not behave the same and the number of errors in different channels can vary. In the first problem we assume that the behavior of the channel system is known and for a given code we study the required number of channels for exact reconstruction. Under this problem we considered the cases in which the decoder knows the number of errors in every channel, only the distribution for the number of
errors, or the average number of errors. In the second problem the number of channels is given and then we followed the same study in order to find the minimum distance of the code that guarantees exact reconstruction.

Appendix A  Proofs for Subsection 6.A.3.3

In this part we present the omitted proofs in Subsection 6.A.3.3. The following proofs apply Lemmas 26 and 27, which were already proved in Subsection 6.A.3.3.

Lemma 28. If a $t$-channel system of size $J$, for some even $J$, supports exact reconstruction for $U$, then for all even positive integer, $N$, a $t$-channel system of size $N$ supports exact reconstruction for $U$.

Proof. Denote $d = d(U)$, and suppose that a $t$-channel system of size $J$, for some even $J$, supports exact reconstruction for $U$. First, we prove that a $t$-channel system of size 2 supports exact reconstruction for $U$. Eventually, we prove an equivalent claim; if a $t$-channel system of size 2 does not support exact reconstruction for $U$, then for every even $N$, a $t$-channel system of size $N$ does not support exact reconstruction for $U$. We prove it by induction on $N$.

For the second part, we prove by induction on $N$, that if a $t$-channel system of size 2 supports exact reconstruction for $U$, then for each even $N$, a $t$-channel system of size $N$ supports exact reconstruction for $U$.

For the first part, if a $t$-channel system of size 2 does not support exact reconstruction for $U$, then, by Lemma 26,

$$[2t] > [((2d - 1)/2)],$$

and we have to prove that for each even $N$

$$[tN] > [(dN - 1)/2].$$

The basis of the induction is $N = 2$. For the step, we assume correctness
for an even $N$ and for 2, and we prove for $N + 2$. This holds since

$$[t(N + 2)] \geq [tN] + [2t]$$

$$> [(dN - 1)/2] + [(2d - 1)/2] + 1$$

$$= (dN - 2)/2 + (2d - 2)/2 + 1$$

$$= (d(N + 2) - 2)/2$$

$$= [(d(N + 2) - 1)/2].$$

For the second part we assume that a $t$-channel system of size 2 supports exact reconstruction for $U$. Then, by Lemma 26,

$$[2t] \leq [(2d - 1)/2],$$

and we have to prove that for each even $N$

$$[tN] \leq [(dN - 1)/2].$$

The basis of the induction is $N = 2$. For the step, we assume correctness for an even $N$ and for 2, and we prove for $N + 2$. Again, this holds since

$$[t(N + 2)] \leq [tN] + [2t] + 1$$

$$\leq [(dN - 1)/2] + [(2d - 1)/2] + 1$$

$$= (dN - 2)/2 + (2d - 2)/2 + 1$$

$$= (d(N + 2) - 2)/2$$

$$= [(d(N + 2) - 1)/2].$$

\[\square\]

**Lemma 29.** Let $N^a(t,d,V) = 1$. If $d$ is even or if a $t$-channel system of size 2 supports exact reconstruction for $U$ of distance $d$, then for all $N \geq 1$ a $t$-channel system of size $N$ supports exact reconstruction for $U$.

**Proof.** The proof is by induction on $N$. For even $d$, the basis is $N = 1$, where for odd $d$ the basis is $N = 1$ and $N = 2$. Now, let $N \in \mathbb{N}$. We assume that for $L \leq N$, a $t$-channel system of size $L$, supports exact reconstruction for $U$, and we prove that a $t$-channel system of size $N + 1$ also supports exact reconstruction for $U$. By Lemma 30, $[tL] \leq [(dL - 1)/2]$ for all $L \leq N$, and we have to prove that $[t(N + 1)] \leq [(d(N + 1) - 1)/2]$. For an even $d$
the following holds

\[
[t(N + 1)] \leq [tN] + [t] + 1 \\
\leq [(dN - 1)/2] + [(d - 1)/2] + 1 \\
= (dN - 2)/2 + (d - 2)/2 + 1 \\
= (d(N + 1) - 2)/2 \\
= [(d(N + 1) - 1)/2],
\]

and for an odd \(d\), if \(N\) is even we get the claim by Lemma 28. Otherwise, \(N \geq 3\) is odd, and we conclude that

\[
[t(N + 1)] \leq [t(N - 1)] + [2t] + 1 \\
\leq [(d(N - 1) - 1)/2] + [2(d - 1)/2] + 1 \\
= (d(N - 1) - 2)/2 + (2d - 2)/2 + 1 \\
= (d(N + 1) - 2)/2 \\
= [(d(N + 1) - 1)/2].
\]

Theorem 27. Let \(t \geq 0, d\) a positive integer, and \(U \subseteq V\) where \(d(U) = d\). Then

1. If \(d < \lfloor 2t \rfloor\) then \(N^a(t, d, V) = \infty\);

2. If \(d > \lceil 2t \rceil\) then \(N^a(t, d, V) = 1\), and exact reconstruction is supported for \(U\) for any size of the system.

3. If \(d = \lceil 2t \rceil\) and \(d\) is even then \(N^a(t, d, V) = \infty\).

4. If \(d = \lfloor 2t \rfloor\) and \(d\) is odd then for all even \(N\), a \(t\)-channel system of size \(N\) does not support exact reconstruction.

Thus, for an integer \(t\), if \(d > 2t\) then \(N^a(t, d, V) = 1\), and exact reconstruction is supported for \(U\) for any size of the system. Otherwise, \(N^a(t, d, V) = \infty\).

Proof. In all the cases we use the claim from Lemma 26, a \(t\)-channel system of size \(N\) supports exact reconstruction for \(U\) of distance \(d\) if and only if \([tN] \leq [(dN - 1)/2]\).
(1) By Lemma 27 we have to prove that a $t$-channel system of size $N = 1$ does not support exact reconstruction for $U$. This holds since
\[
\left\lfloor (dN - 1)/2 \right\rfloor = \left\lfloor (d - 1)/2 \right\rfloor \leq \left\lfloor (2t - 1 - 1)/2 \right\rfloor \\
\leq [t - 1] < [t] = [tN].
\]

(2) By Lemma 29 we have to prove that a system of size $N = 1$ and a system of size $N = 2$ support exact reconstruction for $U$. For $N = 1$ we have,
\[
\left\lfloor (dN - 1)/2 \right\rfloor = \left\lfloor (d - 1)/2 \right\rfloor \geq \left\lfloor (2t + 1 - 1)/2 \right\rfloor \\
= [t] = [tN].
\]
and for $N = 2$
\[
\left\lfloor (dN - 1)/2 \right\rfloor = \left\lfloor (2d - 1)/2 \right\rfloor \geq \left\lfloor (2(2t + 1) - 1)/2 \right\rfloor \\
= [2t + 1/2] \\
\geq [2t] = [tN].
\]

(3) By Lemma 27 we have to prove that a $t$-channel system of size $N = 1$ does not support exact reconstruction for $U$. This holds since
\[
\left\lfloor (dN - 1)/2 \right\rfloor = \left\lfloor (d - 1)/2 \right\rfloor = \left\lfloor (\lfloor 2t \rfloor - 1)/2 \right\rfloor \\
= (\lfloor 2t \rfloor - 2)/2 \\
= \lfloor 2t \rfloor /2 - 1 \\
\leq [t] - 1 < [t] = [tN].
\]

(4) By Lemma 28, it is sufficient to prove that a $t$-channel system of size $N = 2$ does not support exact reconstruction for $U$. This holds since
\[
\left\lfloor (dN - 1)/2 \right\rfloor = \left\lfloor (2d - 1)/2 \right\rfloor = \left\lfloor (2 \lfloor 2t \rfloor - 1)/2 \right\rfloor \\
= (2 \lfloor 2t \rfloor - 2)/2 \\
= \lfloor 2t \rfloor - 1 \\
< \lfloor 2t \rfloor = [tN].
\]
Thus, for an integer $t$, if $d > 2t$ then by (2)) we conclude that $N^a(t, d, V) = 1$ and exact reconstruction is supported for $U$ for any size of the system. Otherwise, by (1)) and (3)), $N^a(t, d, V) = \infty$. 

\[\square\]
Appendix B  Proofs for Section 6.A.4

For the completeness of the results in the paper, we present in this section some omitted proofs of lemmas which are used in this paper.

Let $V = \{0, 1\}^n$ be the set of all length $n$ words over the binary alphabet, $x, z \in V$, and $\rho : V \times V \rightarrow \mathbb{N}$ is a distance function. Recall that $N(x, z, t_1, t_2)$ is the number of elements from $V$ which are of distance at most $t_1$ from $x$, and at most $t_2$ from $z$. If $N(x, z, t_1, t_2)$ depends only on $d = \rho(x, z)$ we denoted this value by $N(d, t_1, t_2)$.

In this section we prove the value of $N(d, t_1, t_2)$ for two types of errors: the value for substitution errors (Hamming distance) which is presented in Lemma 31, is proved in Appendix Appendix B, and for transposition errors (Johnson distance), which is presented in Lemma 35, is proved in Appendix Appendix B.

A Proof for Lemma 31

In this subsection we prove Lemma 31, that is, we prove the value of $N_H(d, n, t_1, t_2)$ for substitution errors. Recall that the Hamming distance function $\rho_H : V_H \times V_H \rightarrow \mathbb{N}$, where $V_H = \{0, 1\}^n$, is defined by $\rho_H(x, z) = |\{i : x_i \neq z_i\}|$, and in this case, $N_H(x, z, t_1, t_2)$ depends only on $d = \rho_H(x, z)$. Thus, we denote by $N_H(d, n, t_1, t_2)$ the value $N_H(x, z, t_1, t_2)$ where $d = \rho_H(x, z)$.

**Lemma 31.** For $1 \leq t_1 \leq t_2$ and $d \geq 1$,

$$N_H(d, n, t_1, t_2) = \sum_{i=0}^{\left\lfloor \frac{t_1+d-t_2}{2} \right\rfloor} \binom{n-d}{i} \cdot \sum_{k=d-t_2+i}^{t_1-i} \binom{d}{k}.$$

**Proof.** Let $x = (x_1, \ldots, x_n)$ and $z = (z_1, \ldots, z_n)$ such that $\rho_H(x, z) = d$. Let $A$ be the set of indices where $x_i \neq z_i$, and $B = [n] \setminus A$, i.e., $A = \{j : x_j \neq z_j\}$, and $B = \{j : x_j = z_j\}$.

Let $y \in I(x, z, t_1, t_2)$, that is, $m = \rho_H(x, y) \leq t_1$ and $\rho_H(y, z) \leq t_2$. The $m$ positions in which $x$ and $y$ differ can be partitioned into two disjoint sets, $i$ positions in $B$, and $k = m - i$ indices in $A$. That is, $i = |\{j : x_j = y_j\}$ and $x_j \neq z_j\}|$ and $k = |\{j : x_j = y_j\}$ and $x_j \neq y_j\}|$. 

173
By $\rho_H(x, y) \leq t_1$ we have $i + k \leq t_1$, and from $\rho_H(z, y) \leq t_2$ we get $i + d - k \leq t_2$. Thus, $d - t_2 + i \leq k \leq t_1 - i$, which implies $d - t_2 + i \leq t_1 - i$, and hence $i \leq \left\lfloor \frac{t_1 + t_2 - d}{2} \right\rfloor$.

For the second direction, we note that two different choices of $i$ indices from $A$, and $k$ elements from $B$, where $0 \leq i \leq \left\lfloor \frac{t_1 + t_2 - d}{2} \right\rfloor$, and $d - t_2 + i \leq k \leq t_1 - i$, yield two different elements in $I(x, z, t_1, t_2)$.

**A Proof for Lemma 35**

In this subsection we prove Lemma 35, that is we prove the value of $N_J(d, n, t_1, t_2)$ for transposition errors. For $x = (x_1, \ldots, x_n) \in \{0, 1\}^n$, a transposition error transposes the symbols $x_i$ and $x_j$. Note that transpositions do not change the Hamming weight of a word. Therefore we consider $V_J = J_w$, the set of all length $n$ words over the binary alphabet with Hamming weight $w$. The Johnson distance function $\rho_J : V_J \times V_J \to \mathbb{N}$ is defined by $\rho_J(x, z) = \left| \{ i : x_i \neq z_i \} \right|$.

Note, that $N_J(x, z, t_1, t_2)$ depends only on $d = \rho_J(x, z)$. Thus, in this case, we denote by $N_J(d, n, t_1, t_2)$ the value $N_J(x, z, t_1, t_2)$ where $d = \rho_J(x, z)$.

**Lemma 35.** For $t_1 \leq t_2$:

$$N_J(d, n, t_1, t_2) = \sum_{i=0}^{t_1} \binom{n - w - d}{i} \cdot \sum_{a=0}^{t_1-i} \sum_{b=0}^{t_2-i} \binom{d}{a} \binom{d}{b} \binom{w - d}{a + b + i - d}.$$

**Proof.** Let $x = (x_1, \ldots, x_n)$ and $z = (z_1, \ldots, z_n)$, two binary words of Hamming weight $w$, where $\rho_J(x, z) = d$. Let us partition the $n$ indices into four disjoint sets:

- $I = \{ j : x_j = z_j = 1 \}$, $|I| = w - d$,
- $A = \{ j : x_j = 1, \; z_j = 0 \}$, $|A| = d$,
- $B = \{ j : x_j = 0, \; z_j = 1 \}$, $|B| = d$, and
- $C = \{ j : x_j = z_j = 0 \}$, $|C| = n - w - d$.
Let \( y \in I(x, z, t_1, t_2) \) where \( y \) is of Hamming weight \( w \). The \( w \) positions in which \( y_j = 1 \) can be partitioned into four sets as follows: \( w - d - i \) positions in \( I \), \( d - a \) in \( A \), \( d - b \) in \( B \), and the rest, \( a + b + i - d \), in \( C \).

By the definition of the sets,

\[
\rho_J(x, y) = \frac{|\{j:j\in I\cup A \land y_j=0\}|+|\{j:j\in B\cup C \land y_j=1\}|}{2} = \frac{i+a+(d-b)+(a+b+i-d)}{2} = i + a,
\]

and

\[
\rho_J(z, y) = \frac{|\{j:j\in I\cup B \land y_j=0\}|+|\{j:j\in A\cup C \land y_j=1\}|}{2} = \frac{i+b+(d-a)+(a+b+i-d)}{2} = i + b.
\]

Recall that \( y \in I(x, z, t_1, t_2) \), i.e., \( \rho_J(x, y) \leq t_1 \) and \( \rho_J(y, z) \leq t_2 \). Therefore, \( i + a \leq t_1 \) and \( i + b \leq t_2 \).

For the second direction, we note that two different choices of \( i \) indices from \( I \) (to be the 0s in \( I \)), \( a \) indices from \( A \) (to be the 0s in \( A \)), \( b \) indices from \( B \) (to be the 0s in \( B \)), and \( a + b + i - d \) indices from \( C \) (to be the 1s in \( C \)), where \( 0 \leq i + a \leq t_1 \) and \( 0 \leq i + b \leq t_2 \), yield two different elements in \( I(x, z, t_1, t_2) \) of Hamming weight \( w \). \( \square \)
Bibliography


6.B Conference Version

Abstract

Motivated by the error behavior in DNA storage channels, in this work we extend the previously studied sequence reconstruction problem by Levenshtein. The reconstruction problem studies the model in which the information is read through multiple noisy channels, and the decoder, which receives all channel estimations, is required to decode the information. For the combinatorial setup, the assumption is that all the channels cause at most some \( t \) errors. However, since the channels do not necessarily have the same behavior, we generalize this model and assume that the channels are not identical and thus may cause a different maximum number of errors. For example, we assume that there are \( N \) channels that cause at most \( t_1 \) or \( t_2 \) errors, where \( t_1 < t_2 \), and the number of channels with at most \( t_1 \) errors is at least \( \lceil pN \rceil \), for some fixed \( 0 < p < 1 \). If the information codeword belongs to a code with minimum distance \( d \), the problem is then to find the minimum number of channels that guarantees successful decoding in the worst case.

6.B.1 Introduction

The sequence reconstruction problem was first proposed and studied by Levenshtein in [8, 9]. In this model, a codeword is transmitted over multiple channels and a decoder, which receives all channel outputs, decodes the transmitted word. The assumption is that all channels are the same and are uncorrelated, with the only exception that all channel outputs are different from each other. This model was originally motivated by chemical and biological processes where the information is replicated and can be read from different noisy sources. However, it was also shown to be relevant in storage technologies, where the stored information has multiple copies or a single copy is read by several different read heads. Specifically, the applicability of this model is most relevant to DNA storage [1, 2, 16, 17, 18]. Both for in vitro and in vivo storage systems, the information has a large number of copies stored in DNA strands and the goal is to read these strands and reconstruct the data, while every estimation of the data is erroneous.

The reconstruction model studied by Levenshtein and later by others
was combinatorial. Suppose all words belong to some space $V$ with distance function $\rho$. It is assumed that the information codeword $x$ belongs to a code with minimum distance $d$ and the number of errors in every channel is at most $t$. Then, the goal is to find the minimum number of channels that guarantees unique decoding in the worst case. Clearly, if $t < \lfloor (d - 1)/2 \rfloor$, then a single channel is sufficient. Otherwise, it was shown that this number has to be greater than the largest intersection of two balls with radius $t$ and minimum distance $d$ between their centers, that is,

$$\max\{|B_t(x) \cap B_t(z)| : x, z \in V, \rho(x, z) \geq d\},$$

where $B_t(x) = \{y \in V : \rho(x, y) \leq t\}$. Later, this combinatorial problem was studied for several channels. In [8], Levenshtein studied the cases of substitution errors, the Johnson graphs, and several more general metric distances. More results for other general error graphs were given in [10, 11], and in [5, 6, 7], it was studied for permutations. The case of permutations with the Kendall’s $\tau$ distance were investigated in [15] as well as the Grassmann graph case. Levenshtein’s results for deletions and insertions in [9] were extended in [13] for insertions and in [3] for deletions. In [14], the connection between the reconstruction problem and associative memories was studied, and in [4] it was analyzed for the purpose of asymptotically improving the Gilbert-Varshamov bound.

Motivated by the error behavior in DNA storage, in this work, we generalize Levenshtein’s model and assume a combinatorial model where the channels are not identical. When reading the data stored in DNA strands, it may happen that some estimations are more noisy than the others [16]. In the reconstruction model this is translated to channels that cause a different maximum number of errors. For example, it is known that for substitution errors, if the transmitted word belongs to a code with minimum Hamming distance 3 and there are at most 2 errors in every channel, then 7 channels are necessary and sufficient for successful decoding. However, if at most 2 channels cause two errors (and the rest 1 error), then we show that 5 channels are necessary and sufficient for successful decoding. In [12], a similar problem was studied for the setup in which every channel can cause a different number of insertions.

Formally, we define this model as follows. Let $\ell$ be the number of possible
types of channels. For $T = (t_1, \ldots, t_\ell)$ and $P = (p_1, \ldots, p_\ell)$, where $t_1 < \cdots < t_\ell \in \mathbb{N}$ and $0 < p_1 < \cdots < p_{\ell-1} < p_\ell = 1$, we say that a system with $N$ channels is a $(T, P)$-channel system if for $1 \leq i \leq \ell$, $[p_i, N]$ of the channels cause at most $t_i$ errors. For example, Levenshtein’s model is a special case with $\ell = 1$ and $p_1 = 1$. Our goal in this work is to study the minimum number of channels $N$ required for a $(T, P)$-channel system for successful decoding when the information is a codeword which belongs to a code with minimum distance $d$. Note that in this case there are two setups we can study, namely, given a $(T, P)$-channel system, the decoder may or may not know the type of each channel. Our main focus will be on substitution errors while other channels are left for future work.

The rest of the paper is organized as follows. In Section 6.B.2, we formally define the models. In Section 6.B.3 we solve the reconstruction problem for the case $\ell = 2$, and we apply this solution for substitution errors in Section 6.B.4. Then, in Section 6.B.5, we extend this analysis for arbitrary $\ell$, and finally in Section 6.B.6, we consider special cases of $\ell = 2$. Due to the lack of space, some of the proofs in the paper are omitted or shortened.

### 6.B.2 Definitions and Problem Setup

For a positive integer $h$, we denote by $[h]$ the set $\{1, 2, \ldots, h\}$. Let $V$ be a finite set with a distance function $\rho: V \times V \to \mathbb{N}$. For $x \in V$, the ball of radius $t$ centered at $x$ is the set $B_t(x) = \{y: \rho(x, y) \leq t\}$. A combinatorial channel $C$ is called a $t$-error channel, if for any input $x \in V$ the output of $C$ is in $B_t(x)$.

A channel system is a system consisting of some $N$ combinatorial channels $C_1, C_2, \ldots, C_N$. We say that a word $x \in V$ is transmitted over the channel system if $x$ is transmitted over $C_i$ for all $i \in [N]$, and $y_i$ is the output of the $i$th channel. The receiver applies a decoding function $D(y_1, \ldots, y_N)$ in order to reconstruct the transmitted word $x$, and exact reconstruction happens when $x = D(y_1, \ldots, y_N)$. In this paper we only refer to the exact reconstruction problem and we assume that all channel outputs are different from each other.

Let $T = (t_1, \ldots, t_\ell)$ and $P = (p_1, \ldots, p_\ell)$ such that $t_1 < t_2 < \cdots < t_\ell \in \mathbb{N}$ and $0 < p_1 < p_2 < \cdots < p_{\ell-1} < p_\ell = 1$. A channel system with $N$
combinatorial channels is called a \((T, P)\)-channel system if for each \(i \in [\ell]\), 
\([p_i, N]\) of the channels are \(t_i\)-error channels. The size of a channel system is 
the number of channels comprised in it.

We consider two models, which depend upon whether the behavior of 
each specific channel is known or unknown to the decoder. In the first 
channel system, called the sequenced-channel system, the decoder knows 
the maximum number of errors in every channel. However, in the second channel 
system, called the non-sequenced-channel system, only the distribution of the 
errors in the channels is known to the decoder, but the number of errors in 
each individual channel is unknown. For example, the decoder may know 
that half of the channels are \(t_1\)-error channels, and the rest are \(t_2\)-error 
channels, but it does not know what the exact type of each channel is.

For \(U \subseteq V\), we denote by \(N^u(T, P, U)\) the minimum size of a \((T, P)\)-non- 
sequenced-channel system such that every \(x \in U\) has exact reconstruction. 
Similarly, \(N^k(T, P, U)\) is defined for the sequenced-channel system. Note 
that \(N^k(T, P, U) \leq N^u(T, P, U)\). In the rest of the paper, whenever we 
write \(g\), we refer to \(g \in \{k, u\}\).

The main problem we study in this paper is formulated as follows.

**Problem 4.** Let \(V\) be a finite set with distance function \(\rho : V \times V \to \mathbb{N}\), 
\(T = (t_1, \ldots, t_\ell)\), and \(P = (p_1, \ldots, p_\ell)\). For all \(U \subseteq V\), find the values of 
\(N^u(T, P, U)\) and \(N^k(T, P, U)\).

### 6.B.3 The case \(\ell = 2\)

In this section we study Problem 4 for two types of channels. This result 
generalizes the case studied by Levenshtein when all the channels are identical [8].

For \(x, z \in V\) and \(t_1 < t_2 \in \mathbb{N}\) we define 
\[I(x, z, t_1, t_2) = B_{t_1}(x) \cap B_{t_2}(z), \quad I(x, z, t_1) = B_{t_1}(x) \cap B_{t_1}(z),\]
and 
\[N(x, z, t_1, t_2) = |I(x, z, t_1, t_2)|, \quad N(x, z, t_1) = |I(x, z, t_1)|.\]

In the sequel, we assume that \(x \neq z\). It is clear that 
\[N^g(T, P, U) = \max\{N^g(T, P, \{x, z\}) : x, z \in U\}.\]
Hence, we focus on finding the value of $N^g(T, P, \{x, z\})$ for all $x, z \in U$. Recall that a $(T = (t_1, t_2), P = (p, 1))$-channel system of size $N$ is a set of $N$ combinatorial channels, where $\lceil pN \rceil$ of the channels are $t_1$-error channels and the others are $t_2$-error channels.

The following theorem solves Problem 4 for the sequenced model. We omit its proof since it is a simplified version of the non-sequenced case.

**Theorem 36.** If $U = \{x, z\} \subseteq V$, $T = (t_1, t_2)$, and $P = (p, 1)$ then $N^k(T, P, U) = N + 1$, where $N = \min \{\lceil N(x, z, t_1)/p \rceil, N(x, z, t_2)\}$.

In the rest of this section, we present the solution for the non-sequenced model. We define

$$N'(x, z, t_1, p) = \min \{L : 2 \lceil pL \rceil - L > N(x, z, t_1), L \geq 1\} - 1,$$

where $\min \emptyset = \infty$. This value will be used in calculating the value of $N^u(T, P, \{x, z\})$. The following proposition studies the value of $N'(x, z, t_1, p)$.

**Proposition 37.** For $0 < p \leq 1/2$:

$$N'(x, z, t_1, p) = \begin{cases} 0 & \text{if } N(x, z, t_1) = 0, \\ \infty & \text{otherwise}. \end{cases}$$

For $1/2 < p < 1$:

$$\left\lfloor \frac{N(x, z, t_1) - 2}{2p - 1} \right\rfloor \leq N'(x, z, t_1, p) \leq \left\lceil \frac{N(x, z, t_1)}{2p - 1} \right\rceil.$$

We note that if $x$ is transmitted over a channel system with $N$ channels, then at least $\lceil pN \rceil$ of the outputs are in $B_{t_1}(x)$, and all the $N$ outputs in $B_{t_2}(x)$. Thus, to support exact reconstruction for $x$, we require that for every $z \in U$, there are no $N$ outputs such that all the following three conditions hold simultaneously

1. at least $\lceil pN \rceil$ of the outputs are in $B_{t_1}(x)$,
2. at least $\lceil pN \rceil$ of the outputs are in $B_{t_1}(z)$,
3. all the $N$ outputs are in $B_{t_2}(x) \cap B_{t_2}(z)$.
The following theorem establishes our result in calculating the value of \( N^u(T,P,U) \).

**Theorem 38.** If \( U = \{x,z\} \subseteq V \), \( T = (t_1,t_2) \), and \( P = (p,1) \) then \( N^u(T,P,U) = N + 1 \), where

\[
N = \min \{ \left\lfloor \frac{N(x,z,t_1,t_2)}{p} \right\rfloor, \frac{N(x,z,t_2)}{p}, \frac{N(z,x,t_1,t_2)}{p}, \frac{N'(x,z,t_1,p)}{} \}.
\]

**Proof.** If a \((T,P)\)-channel system consists of \( J = N + 1 \) channels, then, by the definition of \( N \), at least one of the following conditions holds:

1. \( J \geq \left\lfloor \frac{N(x,z,t_1,t_2)}{p} \right\rfloor + 1 \),
2. \( J \geq \left\lfloor \frac{N(z,x,t_1,t_2)}{p} \right\rfloor + 1 \),
3. \( J \geq N(x,z,t_2) + 1 \),
4. \( 2 \left\lceil \frac{pJ}{N(x,z,t_1)} \right\rceil > J \).

The first condition implies

\[
\left\lceil \frac{pJ}{N(x,z,t_1)} \right\rceil \geq \frac{pJ}{N(x,z,t_1)} \geq p \cdot \left( \left\lfloor \frac{N(x,z,t_1,t_2)}{p} \right\rfloor + 1 \right) > p \cdot \frac{N(x,z,t_1,t_2)}{p} = \frac{N(x,z,t_1,t_2)}{p}.
\]

By the same computation for the second condition, we conclude that at least one of the following conditions holds:

1. \( \left\lceil \frac{pJ}{N(x,z,t_1)} \right\rceil > N(x,z,t_1,t_2) \),
2. \( \left\lceil \frac{pJ}{N(x,z,t_1)} \right\rceil > N(z,x,t_1,t_2) \),
3. \( J > N(x,z,t_2) \),
4. \( 2 \left\lceil \frac{pJ}{N(x,z,t_1)} \right\rceil - N(x,z,t_1) > J \).

The above conditions are symmetric for \( x \) and \( z \). Thus, without loss of generality, let \( x \) be the transmitted word. If Condition ((1)) or ((3)) holds, since \( \lfloor pJ \rfloor \) of the outputs are in \( B_{t_1}(x) \) and \( J \) outputs are in \( B_{t_2}(x) \), then

\[1\]Note that for \( J > N + 1 \), a \((T,P)\)-non-sequenced-channel system of size \( J \) may not support exact reconstruction. That could happen only if \( J \leq \min\{ \left\lfloor \frac{N(x,z,t_1,t_2)}{p} \right\rfloor, \left\lfloor \frac{N(z,x,t_1,t_2)}{p} \right\rfloor, N(x,z,t_2) \} \).
not all the outputs are in $B_2(z)$. If Condition ((2)) holds, there are no $[pJ]$ outputs in $B_1(z)$. Thus, if one of conditions ((1)), ((2)), or ((3)) holds, then $z$ will not be decoded incorrectly. For Condition ((4)), assume that we have $m$ outputs in $I(x, z, t_1)$, where $m \leq N(x, z, t_1)$. In order to decode $z$ incorrectly we must have at least $[pJ] - m$ outputs in $I(z, x, t_1, t_2) \setminus I(x, z, t_1)$. Furthermore, since $x$ was transmitted at least $[pJ] - m$ outputs are in $I(x, z, t_1, t_2) \setminus I(x, z, t_1)$. Thus, we must have that $2 \cdot [pJ] - m \leq J$ in contradiction to Condition ((4)).

For the second direction we have to prove that $N$ channels are not sufficient for exact reconstruction where $U = \{x, z\}$. The following four conditions hold simultaneously.

1. $[pN] \leq N(x, z, t_1, t_2)$,
2. $[pN] \leq N(z, x, t_1, t_2)$,
3. $N \leq N(x, z, t_2)$,
4. $2 \cdot [pN] - N(x, z, t_1) \leq N$.

The first condition is derived as follows. If $[pN] = pN$, then

$$[pN] = pN \leq p \cdot \lfloor N(x, z, t_1, t_2)/p \rfloor \leq p \cdot N(x, z, t_1, t_2)/p = N(x, z, t_1, t_2).$$

Otherwise, $[pN] = \lfloor pN \rfloor + 1$, and

$$[pN] = \lfloor pN \rfloor + 1 < p \cdot \lfloor N(x, z, t_1, t_2)/p \rfloor + 1 \leq p \cdot N(x, z, t_1, t_2)/p + 1 = N(x, z, t_1, t_2) + 1.$$  

Thus, for both cases, $[pN] \leq N(x, z, t_1, t_2)$. The second condition is obtained by the same way.

For this part, we present a set of $N$ outputs which can be the outcome when transmitting either $x$ or $z$. Let $m = N(x, z, t_1)$. If $m < [pN]$ then $m$ outputs are in $I(x, z, t_1)$, at least $[pN] - m$ outputs are in $I(x, z, t_1, t_2) \setminus I(x, z, t_1)$ (by Conditions ((1)) and ((4))), at least $[pN] - m$ in $I(z, x, t_1, t_2) \setminus I(x, z, t_1)$ (by Conditions ((2)) and ((4))), and all the outputs are in $I(x, z, t_2)$ (by Condition ((3))). Otherwise, $m \geq [pN]$, and then
outputs are in $I(x, z, t_1)$ and all the rest are in $I(x, z, t_2)$ (by Condition ((3))). Thus, at least $\lceil pN \rceil$ of the outputs are in $B_{t_1}(x)$, and all the $N$ outputs are in $B_{t_2}(x)$, and the same holds for $z$. $\square$

Note that the setup where all the channels are $t$-error channels is a special case of $N^u(T, P, U)$ and $N^k(T, P, U)$ for $T = (t, t_2)$ and $P = (1, 1)$.

The following corollary is deduced immediately by Proposition 37 and Theorem 38.

**Corollary 6.** $N^u(T, P, \{x, z\}) = N + 1$ where $N$ is defined as follows. For $0 < p \leq 1/2$:

$$N = \begin{cases} 
0 & \text{if } N(x, z, t_1) = 0 \\
\min\{\lfloor N(x, z, t_1, t_2)/p \rfloor, \lfloor N(z, x, t_1, t_2)/p \rfloor, N(x, z, t_2)\} & \text{otherwise.}
\end{cases}$$

For $1/2 < p < 1$:

$$N = \min\{\lfloor N(x, z, t_1, t_2)/p \rfloor, N(x, z, t_2), \lfloor N(z, x, t_1, t_2)/p \rfloor, N'(x, z, t_1, p)\}.$$ 

In the following section we show how to apply the result from Corollary 6 to explicitly solve Problem 4 with $\ell = 2$ for substitution errors over the binary alphabet.

### 6.B.4 Substitution errors

Let $V = \{0, 1\}^n$ be the set of all length $n$ words over the binary alphabet. The Hamming distance function $\rho : V \times V \to \mathbb{N}$ is defined by $\rho(x, z) = |\{i : x_i \neq z_i\}|$.

Note, that for all $x, z \in V$, $N(x, z, t_1, t_2)$ and $N(x, z, t)$ depend only on $d = \rho(x, z)$. Thus, for $x, z \in V$ such that $d = \rho(x, z)$, we denote by $N(d, t_1, t_2)$ and $N(d, t)$ the values $N(x, z, t_1, t_2)$ and $N(x, z, t)$, respectively. Let $N^g(T, P, d)$ be defined as the maximum value of $N^g(T, P, U)$, for all $U$ such that $d(U) \geq d$, where $d(U) = \min\{\rho(x, z) : x, z \in U\}$. As before we get
that
\[ N^g(T, P, d) = \max\{ N^g(T, P, \{ x, z \}) : x, z \in V, \rho(x, z) \geq d \} . \]

The next theorem proves that for all \( d \geq 1 \), \( N^g(T, P, d) \geq N^g(T, P, d + 1) \). This desirable property, known as the monotonicity by intersection [8], holds also in our case.

**Theorem 39.** For fixed \( p \), \( 0 < p < 1 \), \( d \geq 1 \), and \( t_1 < t_2 \), \( N^g(T, P, d) \geq N^g(T, P, d + 1) \), where \( T = (t_1, t_2) \) and \( P = (p, 1) \).

According to Theorem 39, in order to calculate the value of \( N^g(T, P, d) \) it is enough to find the value of \( N^g(T, P, \{ x, z \}) \), where \( \rho(x, z) = d \). Therefore, according to Theorem 36 and Theorem 38, for \( T = (t_1, t_2) \) and \( P = (p, 1) \), we conclude that
\[
N^k(T, P, d) = \min \{ \left\lfloor N(d, t_1)/p \right\rfloor, N(d, t_2) \}, \quad \text{and} \quad N^u(T, P, d) = \min \{ \left\lfloor N(d, t_1, t_2)/p \right\rfloor, N(d, t_2), N'(d, t_1, p) \},
\]

where \( N'(d, t_1, p) = \min \{ L : 2 \left\lceil pL \right\rceil - L > N(d, t_1), L \geq 1 \} - 1 \).

In the sequel, we find explicitly the value of \( N^u(T, P, d) \). We focus on the non-sequenced model, since the sequenced one can be easily derived from Theorem 36 and Levenshtein’s results in [8].

The following lemma was shown in [8].

**Lemma 39.** For \( t, d \geq 1 \),
\[
N(d, t) = \sum_{i=0}^{\left\lfloor t - \frac{d}{2} \right\rfloor} \binom{n - d}{i} \cdot \sum_{k=d-t+i}^{t-i} \binom{d}{k},
\]

where \( \binom{a}{b} = 0 \) if \( a < b \) or \( b < 0 \).

Note that \( t - \left\lfloor \frac{d}{2} \right\rfloor = \left| t - \frac{d}{2} \right| \). By similar combinatorial computation, we can compute the value of \( N(d, t_1, t_2) \).

**Lemma 40.** For \( t_1 \leq t_2 \):
\[
N(d, t_1, t_2) = \sum_{i=0}^{\left\lfloor \frac{t_1 + t_2 - d}{2} \right\rfloor} \binom{n - d}{i} \cdot \sum_{k=d-t_2+i}^{t_1-i} \binom{d}{k}.
\]

186
The following two lemmas compare between the three components which determine the value of $N^n(T, P, d)$, for $d \geq 1$, $t_1 < t_2 \in \mathbb{N}$, and fixed $0 < p < 1$. Lemma 41 compares between $\lfloor N(d, t_1, t_2)/p \rfloor$ and $N(d, t_2)$.

**Lemma 41.** For any fixed $p$ and $n$ sufficiently large the following holds. If $d$ is odd, $p \leq 1/2$, and $t_2 = t_1 + 1$, then

$$N(d, t_2) < \lfloor N(d, t_1, t_2)/p \rfloor.$$  

Otherwise,

$$N(d, t_2) \geq \lfloor N(d, t_1, t_2)/p \rfloor.$$  

**Proof.** Note that

$$N(d, t_2) = \Theta(n^{2t_2-d})$$  

and $N(d, t_1, t_2) = \Theta(n^{t_1+t_2-d})$.

Thus, we compare between the powers $\left\lfloor \frac{2t_2-d}{2} \right\rfloor$ and $\left\lfloor \frac{t_1+t_2-d}{2} \right\rfloor$. If $t_2 = t_1 + 1$ and $d$ is odd then $\left\lfloor \frac{2t_2-d}{2} \right\rfloor = \left\lfloor \frac{t_1+t_2-d}{2} \right\rfloor$. In all other cases, $\left\lfloor \frac{2t_2-d}{2} \right\rfloor > \left\lfloor \frac{t_1+t_2-d}{2} \right\rfloor$, and hence $N(d, t_2) > \lfloor N(d, t_1, t_2)/p \rfloor$.

For the case of $t_2 = t_1 + 1$ and odd $d$, we compare the coefficients of the dominant powers. Denote $d = 2m + 1$.

$$N(d, t_2) = \left( \binom{d}{m} \right) = \binom{n-d}{t_1-m} + \sum_{k=m+1}^{m+2} \binom{d}{k} \cdot \binom{n-d}{t_1-m-k} + \Theta(n^{t_1-m-2}),$$  

$$N(d, t_1, t_2) = \binom{d}{m} \cdot \binom{n-d}{t_1-m} + \sum_{k=m+1}^{m+1} \binom{d}{k} \cdot \binom{n-d}{t_1-m-k} + \Theta(n^{t_1-m-2}).$$

Thus, the coefficient of the dominant powers in $N(d, t_2)$ is twice the coefficient of the corresponding term in $N(d, t_1, t_2)$. But, $N(d, t_1, t_2)$ is multiplied by $1/p$. Thus, for $p > 1/2$ we have

$$\lfloor N(d, t_1, t_2)/p \rfloor \leq N(d, t_2),$$

and for $p < 1/2$,

$$\lfloor N(d, t_1, t_2)/p \rfloor > N(d, t_2).$$

For $p = 1/2$, we compare the coefficient of the second dominant powers in these two terms and get that $\sum_{k=m-1}^{m+2} \binom{d}{k} < 2 \cdot \sum_{k=m-1}^{m+1} \binom{d}{k}$. Thus, we conclude that for this case $\lfloor N(d, t_1, t_2)/p \rfloor > N(d, t_2)$.
The following lemma compares between the values of \( N'(d, t_1, p) \) and \( \min\{\lfloor N(d, t_1, t_2)/p\rfloor, N(d, t_2)\} \). Recall that according to Proposition 37, for \( 0 < p \leq 1/2 \), \( N'(d, t_1, p) \in \{0, \infty\} \), and by Lemma 41 if \( 1/2 < p < 1 \) then \( N(d, t_1, t_2)/p \leq N(d, t_2) \). Thus, in Lemma 42 we compare only between \( \lfloor N(d, t_1, t_2)/p\rfloor \) and \( \lfloor N(d, t_1)/2p-1\rfloor \) for \( 1/2 < p < 1 \).

**Lemma 42.** For any fixed \( p \) and \( n \) sufficiently large the following holds. If \( d \) is even, \( t_2 = t_1 + 1 \), and

\( (1/2 < p \leq 2/3 \) or \( 2/3 < p < 3/4 \) and \( d < 2-2p \)), then

\[
\left\lfloor \frac{N(d, t_1)}{2p-1} \right\rfloor > \lfloor N(d, t_1, t_2)/p\rfloor.
\]

Otherwise,

\[
\left\lfloor \frac{N(d, t_1)}{2p-1} \right\rfloor \leq \lfloor N(d, t_1, t_2)/p\rfloor.
\]

**Proof.** Note that

\[ N(d, t_1) = \Theta\left(n^{\frac{2t_1-d}{2}}\right) \quad \text{and} \quad N(d, t_1, t_2) = \Theta\left(n^{\frac{t_1+t_2-d}{2}}\right). \]

Thus, we compare the powers \( \left\lfloor \frac{2t_1-d}{2} \right\rfloor \) and \( \left\lfloor \frac{t_1+t_2-d}{2} \right\rfloor \). If \( t_2 = t_1 + 1 \) and \( d \) is even then

\[
\left\lfloor \frac{2t_1-d}{2} \right\rfloor = \left\lfloor \frac{t_1+t_2-d}{2} \right\rfloor. \]

In all other cases, \( \left\lfloor \frac{2t_1-d}{2} \right\rfloor < \left\lfloor \frac{t_1+t_2-d}{2} \right\rfloor \), and hence,

\[
\left\lfloor \frac{N(d, t_1)}{2p-1} \right\rfloor < \lfloor N(d, t_1, t_2)/p\rfloor.
\]

For the case of \( t_2 = t_1 + 1 \) and even \( d \), we compare the coefficients of the dominant powers.

\[
N(d, t_1) = \left(d_{d/2} \cdot n^{d_{t_1-d/2}} + \Theta(n^{t_1-d/2-1})\right),
\]

\[
N(d, t_1, t_2) = \left(d_{d/2} \cdot n^{d_{t_1+t_2-d/2}} + \Theta(n^{t_1+d/2-1})\right).
\]

Thus, the coefficient of the dominant term in

\[
\left\lfloor \frac{N(d, t_1)}{2p-1} \right\rfloor
\]

is

\[
\frac{1}{2p-1} \left(d_{d/2}\right),
\]

while the corresponding coefficient in \( \lfloor N(d, t_1, t_2)/p\rfloor \) is

\[
\frac{1}{p} \left(d_{d/2} + d_{d/2-1}\right) = \frac{2d + 2d_{d/2-1}p}{(d+2)p} \left(d_{d/2}\right).
\]
The inequality
\[
\frac{2d + 2}{(d + 2)p} < \frac{1}{2p - 1}
\]
holds if and only if
\[
(p \leq 2/3) \text{ or } (2/3 < p < 3/4 \text{ and } d < \frac{2 - 2p}{3p - 2}).
\]

Therefore, we conclude that
\[
\left\lfloor \frac{N(d, t_1, t_2)}{p} \right\rfloor < \left\lfloor \frac{N(d, t_1)}{2p - 1} \right\rfloor
\]
if and only if \(d\) is even, \(t_2 = t_1 + 1\), and
\((1/2 < p \leq 2/3) \text{ or } (2/3 < p < 3/4 \text{ and } d < \frac{2 - 2p}{3p - 2}).\)

According to Corollary 6, Lemma 41, and Lemma 42, we can now summarize the results for the binary substitutions case.

**Corollary 7.** For any fixed \(p\) and \(n\) sufficiently large the following holds.

- For \(0 < p \leq 1/2:\)

  \[N^u(T, P, d) = \begin{cases} 1 & \text{if } d > 2t_1, \\ \Theta(n^{\left\lfloor \frac{t_1 + t_2 - d}{2} \right\rfloor}) & \text{otherwise.} \end{cases}\]

- For \(1/2 < p < 1:\)

  \[N^u(T, P, d) = \Theta(n^{\left\lfloor \frac{2t_1 - d}{2} \right\rfloor}).\]

More specifically,

- For \(0 < p \leq 1/2:\)

  \[N^u(T, P, d) = \begin{cases} 1 & \text{if } d > 2t_1, \\ N(d, t_2) + 1 & \text{otherwise, if } d \text{ is odd} \\ \left\lfloor N(d, t_1, t_2)/p \right\rfloor + 1 & \text{otherwise.} \end{cases}\]
• For $1/2 < p < 1$:

$$N^u(T, P, d) = \begin{cases} 
\left\lceil \frac{N(d, t_1, t_2)}{p} \right\rceil & \text{if } d \text{ is even, } t_2 = t_1 + 1, \\
\left\lceil \left( \frac{1}{2} < p \leq \frac{3}{4} \right) \lor \left( \frac{2}{3} < p < \frac{3}{4} \land d < \frac{2-2p}{3p-2} \right) \right\rceil, & \left( \frac{1}{2} < p \leq \frac{3}{4} \right) \\
N'(d, t_1, p) & \text{otherwise.}
\end{cases}$$

We note that we can generalize Lemma 41 for non fixed values of $p$, i.e, for $p$ which is a function of $n$, however this part is omitted due to lack of space.

Lastly, we discuss some special cases of this model. Let $L_1 = N(d, t_2) + 1$ be the solution for the case where all the channels are identical, and $L_2 = N^u(T = (t_1, t_2), P = (p, 1), d)$ be the solution for our general problem.

• For fixed $p$, $0 < p \leq 1/2$, $d = 1$, and $T = (2, 4)$, $L_2 = \Theta(n^2)$, while $L_1 = \Theta(n^3)$,

• For fixed $p$, $0 < p \leq 1/2$, $d = 1$, and $T = (2, 8)$, $L_2 = \Theta(n^4)$, while $L_1 = \Theta(n^7)$,

• For fixed $p$, $1/2 < p \leq 2/3$, $d = 2$, and $T = (4, 5)$, $L_2 = \Theta(n^3)$, while $L_1 = \Theta(n^4)$.

6.B.5 Problem 4 - The General Case

In this section, we extend the solution from Section 6.B.3. We provide a combinatorial translation for the general case of Problem 4, where $T = (t_1, \ldots, t_\ell)$ and $P = (p_1, \ldots, p_{\ell-1}, p_\ell)$, $t_1 < t_2 < \ldots < t_\ell \in \mathbb{N}$, and $0 < p_1 < p_2 < \ldots < p_{\ell-1} < p_\ell = 1$. A $(T, P)$-channel system of size $N$ consists of $N$ channels, where for each $i \in [\ell]$, $[p_i N]$ channels are $t_i$-error channels.

Theorem 40 and Theorem 41 generalize Theorem 36 and Theorem 38 for arbitrary $\ell$, respectively.

**Theorem 40.** For $x, z \in V$, $N^k(T, P, \{x, z\}) = N + 1$, where

$$N = \min\{\lfloor N(x, z, t_i) / p_i \rfloor : i \in [\ell] \}.$$
Now, we consider the non-sequenced case. Recall that if $x$ is transmitted over a $(T, P)$-channel system of size $N$, then for all $i \in [\ell - 1]$ at least $\lceil p_i N \rceil$ of the outputs are in $B_{t_i}(x)$, and all the $N$ outputs are in $B_{t_\ell}(x)$. Then, $x$ does not have exact reconstruction if there exists a different word $z$, where for all $i \in [\ell - 1]$ at least $\lceil p_i N \rceil$ of the outputs are in $B_{t_i}(z)$, and all the $N$ outputs are in $B_{t_\ell}(z)$.

Theorem 41. For $x, z \in V$, $N^u(T, P, \{x, z\}) = N + 1$, where

$$N = \min \{ \lfloor N(x, z, t_i, t_\ell)/p_i \rfloor : i \in [\ell - 1] \} \cup \{ \lfloor N(x, z, t_i, t_\ell/p_i) \rfloor : i \in [\ell - 1] \} \cup \{ N(x, z, t_\ell) \} \cup \{ N'(x, z, t_i, t_j, p_i, p_j) : i, j \in [\ell - 1] \},$$

$$N'(x, z, t_i, t_j, p_i, p_j) = \min \{ L : \lceil p_i L \rceil + \lceil p_j L \rceil - L > N(x, z, t_i, t_j), L \geq 1 \} - 1,$$

and $\min \emptyset = \infty$.

6.B.6 Special systems for $T = (t_1, t_2)$

In this section we study special cases of two types of channels. For $T = (t_1, t_2)$, $t_1 < t_2$, and a constant integer $a$, a channel system with $N$ combinatorial channels is called a $(T, i, a)$-channel system, $i \in \{1, 2\}$, if $a$ of the channels are $t_i$-error channels, while the others are $t_{3-i}$-error channels. If the size of a system is smaller than $a$, then all the channels are $t_i$-error.

In this model, we consider both cases, sequenced and non-sequenced. For $U \subseteq V$, we denote by $N^u(T, i, a, U), N^k(T, i, a, U)$ the minimum size of a $(T, i, a)$-non-sequence, $(T, i, a)$-sequenced -channel system such that each $x \in U$ has exact reconstruction, respectively.

In this section we solve the following problem for $i \in \{1, 2\}$.

Problem 5. Let $V$ be a finite set with some distance function $\rho : V \times V \rightarrow \mathbb{N}$, for all $U \subseteq V$, find the values of $N^u(T, i, a, U)$ and $N^k(T, i, a, U)$.

As before, we focus on sets of the form $U = \{x, z\}$ since $N^a(T, i, a, U) = \max\{N^a(T, i, a, \{x, z\}) : x, z \in U\}$.
The solution for this problem is presented in the next three theorems. The first theorem solves the problem for constant number of \( t_1 \)-error channels. In this case, the minimum number of channels which are required for exact reconstruction does not depend on knowing the behavior of each channel. The last two theorems present solutions for constant number of \( t_2 \)-error channels; Theorem 44 for non-sequence system, and Theorem 43 for the sequence one.

**Theorem 42.** For \( U = \{x, z\} \subseteq V \) and \( T = (t_1, t_2) \), \( N^k(T, 1, a, U) = N^u(T, 1, a, U) = N + 1 \), where

\[
N = \begin{cases} 
N(x, z, t_2) & \text{if } N(x, z, t_1) \geq a, \\
N(x, z, t_1) & \text{otherwise.}
\end{cases}
\]

Note, that in almost all the cases

\[
N^k(T, 1, a, U) = N^u(T, 1, a, U) = N(x, z, t_2) + 1.
\]

**Proof.** If \( N(x, z, t_1) < a \), then a \((T, 2, a)\)-channel system of size at most \( N(x, z, t_1) + 1 \) contains only \( t_1 \)-channels. Thus, according to Levenshtein [8], \( N^k(T, 1, a, U) = N^u(T, 1, a, U) = N(x, z, t_1) + 1 \). If \( N(x, z, t_1) \geq a \), then it is clear that \( N(x, z, t_2) + 1 \) channels are sufficient.

For the second direction, without loss of generality, let us assume that \( x \) is transmitted over the system. If \( a \) outputs are in \( I(x, z, t_1) \) and all the \( N(x, z, t_2) \) in \( I(x, z, t_2) \), then \( z \) may be decoded incorrectly. \( \square \)

In the second case \( i = 2 \) and \( a \) is the number of channels with maximum \( t_2 \) errors. First, we state the solution for the case where the type of the channels is known.

**Theorem 43.** For \( U = \{x, z\} \subseteq V \) and \( T = (t_1, t_2) \), \( N^k(T, 2, a, U) = N + 1 \), where

\[
N = \min\{N(x, z, t_1) + a, N(x, z, t_2)\}.
\]

Note, that in almost all the cases

\[
N^k(T, 2, a, U) = N(x, z, t_1) + a + 1.
\]

Lastly, we solve Problem 5 for the non-sequenced model.
**Theorem 44.** For $U = \{x, z\} \subseteq V$ and $T = (t_1, t_2)$, $N^u(T, 2, a, U) = N + 1$, where

$$N = \min \{ N(x, z, t_1, t_2) + a, \ N(x, z, t_2), \ N(z, x, t_1, t_2) + a, \ N(x, z, t_1) + 2a \}.$$  

Note, that in almost all the cases

$$N^u(T, 2, a, U) = N(x, z, t_1) + 2a + 1.$$  

**Proof.** The proof is similar to the one of Theorem 38. If a $(T, 2, a)$-channel system consists of $J = N + 1$ channels, then, by the definition of $N$, at least one of the following conditions exists:

1. $J - a > N(x, z, t_1, t_2)$,
2. $J - a > N(z, x, t_1, t_2)$,
3. $J > N(x, z, t_1)$,
4. $2(J - a) - N(x, z, t_1) > J$.

The above conditions are symmetric for $x$ and $z$. Thus, without loss of generality, let $x$ be the transmitted word. If Condition (1) or (3) holds, since $J - a$ of the outputs are in $B_{t_1}(x)$ and $J$ outputs in $B_{t_2}(x)$, then not all the outputs are in $B_{t_2}(z)$. If Condition (2) holds, there are no $J - a$ outputs in $B_{t_1}(z)$. Thus, if one of the conditions (1), (2), or (3) holds, then $z$ will not be decoded incorrectly. Regarding Condition (4), assume that we have $m$ outputs in $I(x, z, t_1)$, where $m \leq N(x, z, t_1)$. In order to decode $z$ incorrectly we must have at least $J - a - m$ outputs in $I(z, x, t_1, t_2) \setminus I(x, z, t_1)$. Furthermore, since $x$ was transmitted at least $J - a - m$ outputs are in $I(x, z, t_1, t_2) \setminus I(x, z, t_1)$. Thus, we must have that $2(J - a) - m \leq J$ in contradiction to Condition (4).

For the second direction we have to prove that $N$ channels are not sufficient to exact reconstruction where $U = \{x, z\}$. The following four conditions hold simultaneously.

1. $N - a \leq N(x, z, t_1, t_2)$,
2. $N - a \leq N(z, x, t_1, t_2)$,
3. $N \leq N(x, z, t_2)$,
4. $2(N - a) - N(x, z, t_1) \leq N$.  

193
For this part, we present a set of $N$ outputs which can be the output of both $x$ and $z$. Let $m = N(x, z, t_1)$. If $m < N - a$ then, $m$ outputs are in $I(x, z, t_1)$, at least $N - a - m$ in $I(x, z, t_1) \setminus I(x, z, t_1)$ (by Conditions ((1)) and ((4))), at least $N - a - m$ in $I(z, x, t_1, t_2) \setminus I(x, z, t_1)$ (by Conditions ((2)) and ((4))), and all the others in $I(x, z, t_2)$ (by Condition ((3))). Otherwise, $m \geq N - a$, and then at least $N - a$ outputs are in $I(x, z, t_1)$ and $a$ in $I(x, z, t_2)$ (by Condition ((3))). Thus, at least $N - a$ of the outputs are in $B_{t_1}(x)$, and all the $N$ outputs in $B_{t_2}(x)$, and the same holds for $z$.

According to the previous theorem, one can verify that for the Hamming case with $a = 2$, $t_1 = 1$, $t_2 = 2$, and $\rho(x, z) = 3$, we get that $N^u(T, 2, a, U) = 5$, while if all channels cause at most 2 errors, then the number of channels for exact reconstruction is 7 [8].

Note that Theorem 44 can be also derived by a slight modification in Theorem 38. We denote $m = N(x, z, t_1)$ and we define here

$$N'(x, z, t_1, p) = \min\{L: 2\lceil pL \rceil - L > m, \lceil pL \rceil > m, L \geq 1\} - 1,$$

instead of the previous definition, where

$$N'(x, z, t_1, p) = \min\{L: 2\lceil pL \rceil - L > m, L \geq 1\} - 1.$$

This change has no affect on Theorem 38, since for fixed $p$, $0 < p < 1$, $2\lceil pL \rceil - L \leq \lceil pL \rceil$. Then, by substituting $\lceil pL \rceil = L - a$ in Theorem 38 we can conclude Theorem 44.
Bibliography


Discussion
Chapter 7

Discussion

This dissertation discussed coding schemes for non-volatile memories. It is based on my papers which were published and submitted to the main journals and conferences in the fields of coding theory and information theory. This topic combines the theory and practicality of computer sciences.

Flash memory is the most common non-volatile technology because its high storage density and relative low cost. It is a solid-state non-volatile storage medium which consists of cells that are electrically erased and reprogrammed. A distinguishable property of this technology is the asymmetry between adding and removing charge. While adding charge to a single cell is fast, removing charge requires erasing of a block of cells which damages the speed, performance, and the lifetime of the memory. We consider two known schemes which their study is promoted by the constraints of flash memories, namely, write-once memory (WOM) codes and rank modulation (RM). Both schemes aim to increase the amount of data that is written to the memory before an erasure.

The last subject discussed in this thesis is the sequence reconstruction problem. This study is relevant to DNA storage, which has the potential to be the next generation of non-volatile memories.

7.1 WOM codes

WOM consists of cells where each cell store one of $q$ levels according to its exact discrete charge level. A $q$-ary $t$-write WOM code is a coding scheme
for storing \( t \) messages in \( n \) \( q \)-ary cells such that the charge level in each cell can only increase. These codes make it possible to record data \( t \) times in a write-once storage medium.

We study four models of WOM which introduced first in [39] for the binary case. These models are defined according to the input of the encoder and the decoder, that is, whether they know the message before the new data was encoded. We denote by EI/EU the case where the encoder is informed/uninformed with the memory state before encoding, i.e., the encoder reads/do not read memory before encoding new data, respectively. The DI vs. DU are defined similarly for the decoder.

The most practical model, the EI:DU, is rigorously studied in the literature. However, just a little was known regarding the other models. In Chapter 4 [20] we refer to all these four models. First, we follow the definition in [26], and define these models formally for both the \( \epsilon \)-error and the zero-error cases for all \( q \), see Definition 7. Then, we present constructions for WOM codes in models 3 and 4, the EU models, especially for the binary case. We present simple constructions for the zero-error case which achieve relatively high sum-rates. We also provide codes by reduction of these models to two known channels. The binary erasure channel (BEC) is used for constructing binary WOM codes in model 3 (EU:DI), and codes in the \( Z \) channel provide constructions for binary WOM codes in model 4 (EU:DU). The reductions provide maximum sum-rate achieving codes, as long as we have capacity achieving codes in these channels.

In addition, we study the capacity regions and the maximum sum-rates in all these models. Wolf et al. [39] studied extensively all the four models in the binary case, but, much less was known for the non-binary case.

Regarding the binary case, it was proved [39] that the \( \epsilon \)-error and the zero-error capacity regions in models 1 and 2 are all identical, and thus the maximum sum-rates in these cases is equal. Furthermore, the capacity region for model 3 for the \( \epsilon \)-error in the binary case is equal to models 1 and 2. Note that the capacity region and the maximum sum-rate in model 3 for the zero-error case are still unknown. Wolf et al. also partially solved the \( \epsilon \)-error capacity region for model 4 in the binary case, by finding an achievable region where it is unknown whether this is a tight region. However, they could still calculate the maximum sum-rate in this model. For example, for two-write binary WOM in model 4, the maximum sum-rate is 1.3881 and for
For the non-binary case, only the zero-error capacity region for model 2 was known and the maximum sum-rate in models 1 and 2 was shown to be $\log_2 \left( \frac{q^{-1+t}}{q-1} \right)$ [9, 10, 15].

In [18, 20], see Section 4.5, we proved that in the non-binary case, exactly as for the binary case, the $\epsilon$-error and the zero-error capacity regions are all identical and thus the maximum sum-rates in these cases are the same as well. Yet, in Section 4.6 we present the capacity region for model 3 the $\epsilon$-error case which, unlike the binary case, is a proper subset of models 1 and 2. Furthermore, although we do not know the exact value of the maximum sum-rate for model 3 in the non-binary case, we still could prove that it is lower than the maximum sum-rate in models 1 and 2. We also prove that the capacity region of the WOM in the non-binary case in model 4 is a proper region of the capacity region in model 3, and its maximum sum-rate is lower.

We can summarize the open questions regarding the capacity region and the maximum sum-rate in all the models. Models 1 and 2, the EI models, are completely proved. The capacity region and the maximum sum-rate for models 3 and 4, the EU models, in the zero-error case are still unknown. Regarding the $\epsilon$-error case in models 3 and 4, we conclude with the following open problems.

- Model 3, EU:DI –
  - Find explicitly the maximum sum-rate for all $q > 2$.

- Model 4, EU:DU –
  - In the binary case, find the capacity region. In particular, prove/disprove that the achievable region which is presented in [39] is exactly the capacity region.
  - In the non-binary case, find the capacity region and the maximum sum-rate for all $q > 2$.

### 7.2 Rank modulation

The second scheme for flash memories we focus on is rank modulation (RM). In this scheme the data is stored in the memory by the permutation induced in the charge levels of the cells. This scheme was proposed in [24] to cope
with common problems in flash memories, e.g., overshooting, charge drift, and charge leakage from one cell to another. The only programming operation is push-to-the-top which means raising the charge level of one cell above all the others. Thus, both overshooting and drift problems can be overcome this way.

This dissertation contributes for the RM scheme in two topics. The first is providing efficient constructions for snake-in-the-box codes under the Kendall’s $\tau$-metric, Chapter 3, [13]. The second topic we discuss is local rank modulation scheme, Chapter 5, [16], in which, we propose a new family of local rank modulation schemes.

### 7.2.1 Snake-in-the-Box Codes

In Chapter 3, snake-in-the-box codes for permutations using the operation push-to-the-top and the Kendall’s $\tau$-metric were discussed. In abbreviation, we call such codes as $K$-snakes. A $K$-snake is a rank modulation Gray code (RMGC) with Kendall’s $\tau$-distance at least 2, where an $n$-RMGC is defined by a sequence of permutations over $S_n$ where each permutation is followed by a result of a push-to-the-top operation.

The goal in this case is to design snake-in-the-box codes of largest possible size. Recently, Yehezkeally and Schwartz [46] constructed a snake of length $(2n+1)(2n-1)M$ for permutations of $S_{2n+1}$, from a snake of length $M$ for permutations of $S_{2n-1}$. We improve this result ([13], Chapter 3), by constructing a snake of length $((2n+1)2n-1)M$ for permutations of $S_{2n+1}$, from a snake of length $M$ for permutations of $S_{2n-1}$, where both the initial and the constructed snakes hold some properties as detailed in [13]. This recursive construction with the initial value $M_3 = 3$ attains $\lim_{n \to \infty} \frac{M_{2n+1}}{|S_{2n+1}|} \approx 0.4338$, improving on the previous known ratio of $\frac{1}{\sqrt{\pi n}}$ [46], while it is proved [46] that $\frac{M}{|S_n|} \leq 0.5$ where $M$ is maximal size of a $K$-snake over $S_n$.

We also propose a direct construction to form a snake of length $\frac{(2n+1)!}{2} - 2n + 1$ for permutations of $S_{2n+1}$. We applied the direct construction successfully for $S_7$ and $S_9$. The proof of this construction is completed later in [47] by Zhang and Ge, who also propose a method for increasing the size of such a $K$-snake by two permutations, that is, the size of their $K$-snake is $\frac{(2n+1)!}{2} - 2n + 3$. 201
We note that all the above constructions are over the set of permutations $S_n$, where $n$ is odd. The case of even $n$ was discussed later by Wang and Fu [37], who construct a snake in $S_{2n+2}$ with exactly one more permutation than an optimal snake in $S_{2n+1}$ and later Zhang and Ge proposed in [48] an explicit construction of a snake in $S_{2n+2}$ with size asymptotically approaching $\frac{1}{4}|S_{2n+2}|$.

Several questions are still open for future research regarding $K$-snakes.

1. What is the maximum size of a $K$-snake in $S_{2n+1}$? For example, if $2n + 1 = 3$ then the maximum is $|S_{2n+1}| = 6$, but if $2n + 1 = 5$ then the maximum is $|S_{2n+1}| - 3 = 57$. Perhaps, the largest size for $n > 2$ is $\frac{(2n+1)!}{2} - 2n + 3$, or maybe we can do better and achieve the size $|S_{2n+1}| - 3$?

2. Find the largest $K$-snake in $S_{2n}$.

7.2.2 Local Rank Modulation

The local rank modulation (LRM) scheme was defined [6, 7] to overcome the disadvantages of the RM scheme, where many cells are compared, which requires many different charge levels, and the reading is slower.

In [16], Chapter 5, encoding, decoding, and enumeration of the $(1, t, n)$-LRM scheme are studied. In this scheme, $n$ physical cells are view cyclically, where each $t$ consecutive cells represent one symbol from $t!$, according to their permutation. The simplest case, $t = 2$, is solved in [6]. We provide a complete solution for the $(1, 3, n)$-LRM scheme. A simple encoding for the $(1, t, n)$-LRM scheme for any $t \geq 3$ is also presented. For the $(1, 3, n)$-LRM scheme a related decoding was presented. However, we could not prove such a successful decoding for $t > 3$. We also proved that if $M_t$ is the number of legal codewords in the $(1, t, n)$-LRM scheme then $\lim_{n \to \infty} \frac{M_t}{t^n} = 1$. We conclude this section with open problems which are listed in [16] regarding the $(1, t, n)$-LRM scheme for $t \geq 4$,

1. Find an efficient algorithm to determine if a given codeword in the $(1, t, n)$-LRM scheme is legal or not.

2. Prove that the encoding algorithm for the $(1, t, n)$-LRM scheme induces a bijection between the realizable base-words and the legal codewords.

3. Find an efficient decoding algorithm for the $(1, t, n)$-LRM scheme.
7.3 Sequence Reconstruction

In [21], Chapter 6, we study a generalization of the sequence reconstruction problem which was defined and considered first by Levenshtein [29]. In this setup a codeword is transmitted over several channels, and the receiver should decode the correct word by all the outputs. In the original problem all the channels are identical. We study the case where the number of errors in different channels can vary. We consider two problems, where in both we define three models depend on whether the decoder knows the number of errors in every channel, only the distribution for the number of errors, or the average number of errors. In Problem 1, for a given code we study the required number of channels for exact reconstruction, while, in Problem 2, the number of channels is given and we have to find the minimum distance of the code that guarantees exact reconstruction. We also demonstrate the result for the Hamming and the Johnson distance functions.
Bibliography


205


Theorem: Let RM be a coding scheme that represents a set of
memory locations, each location representing a symbol from a
set of symbols. In the RM scheme, the memory locations are
represented by permutations of the charge levels in the locations.
Hence, the exact charge of a location is not relevant, but only its
relation to the other charges. In this scheme, the only
permitted operation on the locations is "move up," which
means moving the charge in a location to a higher location.
This operation is simple and fast. Our work contributes to
the study of this scheme in two areas: hash code in a
container for RM and development of local rank modulation
(local rank modulation, LRM).

A hash code container is a Gray code for RM that
is capable of detecting a single error in the function of
Kendall's Kendall's τ. In the Gray code for RM, the
codes are permutations and each permutation results
from moving up the previous permutation. We deal with
the distance function Kendall's τ due to its relevance to
problems encountered in these memories. Our goal is to
build hash code containers as large as possible. In this work,
we present two structures, one recursive and the other
direct, both of which are significant improvements over
the previous ones.

The drawback of the RM scheme is the need for many
levels and comparisons in reading. To overcome this,
Ginosar et al. defined the LRM scheme. In this scheme,
two consecutive locations define a symbol according to
the permutation of the charge levels in them, where the
difference between any two windows is s, and the sequence
of locations is circular. In this scheme, n divides s.
In a physical sense, the simplest scheme is with a
window size of 2.

In this work, we propose a coding for LRM
with the same window size s. In the last section,
we deal with the problem of hash code retrieval
because it is related to DNA storage. In DNA storage,
the information is copied and stored on different
tracks, and the reader must decode the data by
reading these tracks, as errors in each track may
occur. This problem was first presented by
Luwstein 01 and was explored by him for the case
in which all the errors are the same. We extend
the problem to the case in which the number of errors
in different tracks can be different. The
ambiguity between these two problems:
The first problem deals with the case in which we
are given the maximum number of errors in each
track, whether we know the maximum number of
errors in each track, only the distribution
of the number of errors in each track, or only the
mean number of errors.

אני מודה לטכניון, לקרן גוטווירט, לקרן ג'ייקובס, ולקרן על שם סלים ורחל בנין על התמיכה הכספית הנדיבה בהשתלמותי.

**תקציר**

תזה זו עוסקת בסכימות קידוד לזיכרונות בלתי נדיפים. הבעיות הנידוגות בעבודת מחקר זו קשורות ברובן לזכרון הבזק (flash) שמאדים מצוים היום. дополнительно, ישנה התייחסות לבעיה הקשורה לאחסון DNA. נראה כי הוא חלק אינטגרלי מהדור הבא של אחסון מידע.

זכרונות הבזק (flash) הוא סוג זיכרון בלתי נדיף המאפשר כתיבה, מחיקה, וכתיבה חוזרת. זיכרון זה הוא אחד הזיכרונות הפופולרים היום благодар בצפיפות, אמינות, מהירות, ומחירו הנמוך יחסית. זיכרון זה שומר את המידע בתאי זיכרון, אך בעוד שקריאת תא בודד היא פעולה מהירה מאוד, כתיבה לתא אחד איטית ודורשת מחיקה של תאים רבים (מאות או אף אלפי פאבים, חלול מספר פעמים שבאים↘). במילים אחרות, כתיבת מידע הוא פעולה מהירה, אך מחיקתו היא פעולה איטית.

זכרון המורכב מתתי זיכרון, אשר כל אחד מהם מכיל רמת מטען בין \( q \) רמות שונות, ומייצג אות בין \( 0 \) ל\( q-1 \). קוד WOM הוא סכימת קידוד המאפשרת לכתוב מספר פעמים לתאי הזיכרון כאשר בכל כתיבה ניתן להעלות את רמת המטען בתאים אך לא להוריד. במודל הרגיל המקודד קורא את מצב הזיכרון לפני הכתיבה, ומפענח אי אפשר לדעת מה היה מצב הזיכרון לפני הכתיבה האחרונה. מודל זה נחקר רבות כיוון שהוא הכי שימושי, אך ניתן לח碼 שלושה מודלים כתלות במידע הנתון למפענח ולא虛אום הרמנון. כל המודלים האלו הוגדרו שלוש פעמים בעת 1982,1984 ו-1999. הם חקרו את תחום הקיבול והקצב המקסימלי עבור המקרה הבינארי \( q = 2 \) ב모ודל הרגיל. אנחנו מרחיבים את המחקר הנ"ל גם למקרה הלא בינארי, עבורו היה ידוע הקיבול רק עבור המודל הרגיל לפני שnoon\( q \). בעבודה זו גם מוצעים קודים עבור המודלים בהם המקודד לא קורא את מצב הזיכרון לפני כתיבת ההודעה האחרונה.

מאחר שיש צורך בכתיבת מידע עם רמת מטען מדוייקת, הכתיבה נעשית על ידי פוטון במספר איטרציות. תהליך זה גורם להאטת פעולת הכתיבה. כמו כן ישנן בעיות נוספות כמו זשל מטען מהתאים וזליגה מטען לתא אחר. בכדי להתמודד עם בעיות אלו זיג'אנג והאחרים הציעו את סכימת הקידוד אפנון דירוגי (rank modulation, RM).

**טכניון - תחום מחקר לסטודנטים בדוקטורט - דיק nowrap - 2017**
תודה особенно לחרים, שהמליצו לי ללמוד ולהתרשם כבר מגיל צעיר. שותפתם לסלק את שערם, והישארתם לאן שיעוד ע AGAIN ומקורות. לתוך השמירה האנמונת אינדרודותר את הידיעה השקרית.坦克 לאחים ולאחייתו, בהופעת רחל ואחריתו, שמעתם, סירתם וביפל
תודה לאמור האורתודוקס, בפרס אוחין רחל ואחריתו א الخيارات, שמעתם, סירתם וביפל
בילדות, עדותם, והתאנות בכננה
תודה לבני המרוב, שהשלימוعلم וтяжב את הפיך שכול לשון. התאנות של האנמונת
בתבנה. נתן לא الشريف לבליבת תקיפה והתאנות, ול녂르ון זה. האחת מהתאנות
ליאוראל כל תרשים.坦克 ישמשות של בלין על התאנותnic
תודה לילר המผลกระทบ, אמש נתם לא מת בהפיך והתאנות, ל mudançaות את פרפרציהות העניקה
cיושרי היום, הרגלח דואת להיות ורזה ורזה ול지요 ידיב, כמבק_yield, ליפדר והרי
הרבח מביר על צעירים שלמים אלא וליהי דוערה.
תודה לכתב! זה מ gratuites לכל האנשים והירוקים של בלדרים ואחיין מתייגת חלום.坦克 לכתב!
המחקר בוצע בהנחייתם של פרופ' איתן יעקובי ופרופ' טובי עציון, בפקולטה למדעי המחשב בטכניון. תודה רבה לעמיתם של פרופ' איתן יעקובי ופרופ' טובי עציון, הנותרים, הבוחרים והמתים מחברות.

ברצוני להודות לבורא עולם ששלח לי את האנשים הנכונים, הרעיונות, הכח והיכולת לכתוב תזה זו. אני אסירת תודה למנחים שלי פרופ' איתן יעקובי ופרופ' טובי עציון אשר הדריכו את המחקר וליווני באופן צמוד במקצועיות ובאנושיות. פרופ' טובי עציון עמו התחלתי לעבוד לקראת תואר המאסטר, ופרופ' איתן יעקובי המנהלה האחראית מאז תחילת לימודי הדוקטורט.

טובי, אתה הצתתבי את החשק לחקור. לימדת אותי מה לשאול ואיך, והנחת אותי ברזי הכתיבה האקדמית. דלת משרדך הייתה תמיד פתוחה לך. בסבלנות שאין שני לה לימדתי אתיך לבצע מחקר מתחילתו ועד העלאת הדברים על הכתב בצורה ברורה. ייעצת לי בכל עניין וסיפחה פudadת כלום.

איתן, אתה הבערתבי את הרצון לחקור. לימדת אותי לחשוב מסודר ולהגדיר בעיה. זרקת אתיך למים ונתתבי אמון אותו היה עלי להוכיח. למדתי ממך להכין הרצאה ומצגת. כיוון אתה היהзав_rotation_correction: true דלת משרדך בפתים, וחמה נפלאה.

תודה לשניכם על תמיכה, עידוד והכוונה מפסיעה ראשונה בעולם האקדמי ועד לעצמאות מחקרית. דאגתם לי למלגות ושאצליח, אף יותר מאשר דאגתי לעצמי. תודה נתתם לי ללמוד ממכם בכל התחומים, ואני מבטיחה ליישם.

המון תודות לקבוצת המחקר היוותה עבורי חברה מקצערית ו산업ית גם יחד. תודה לכם المشارיכים ומליצוניכם ש jesteście הרפובליק ותומכים של ענני.

בפרט אודה לצוות הקורס "אינטגרטים" שה 자체 גרה שFillColor עולם ששלח לי את האנשים הנכונים, הרעיונות, הכח והיכולת לכתוב תזה זו. אני אסירת תודה/manihim יבגדי,一览ר, הבוחרים והמתים מחברות.

ברצוני להודות לעמיתם של פרופ' איתן יעקובי ופרופ' טובי עציון, הנותרים, הבוחרים והמתים מחברות.
שיטות קידוד לזכרונות לא נדיפים

דובר על מחקר

לשם מילוי חלק של הדרישות לקבלת התואר

Doctor of Philosophy

מיכל הורוביץ

 miglior התוכנית – מרכז סטטטיקלי לישראל

משרד התשתיות והمجموعات 2017
شيיטות קידוד לזכרונות לא דינמיים

מיכל הורוביץ