UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Hans Kristjan Veri

Low-Complexity Decoding of Best Known Quasi-Cyclic Linear Codes

Bachelor's Thesis (9 ECTS)

Supervisor(s): Irina Bocharova, PhD
Boris Kudryashov, PhD
Vitaly Skachek, PhD

Tartu 2023

Low-Complexity Decoding of Best Known Quasi-Cyclic Linear Codes

Abstract: Error-correcting codes are widely used in modern communications to detect and correct errors that can occur during the transmission of digital information. These codes help ensure reliable transmission by enabling the receiver to reconstruct the original information even if some bits of data were lost or corrupted during transmission. This thesis focuses on improving the decoding schemes of quasi-cyclic error-correcting codes. A classical theoretical framework is provided for understanding error-correcting codes. Quasi-cyclic codes which have the best parameters for given length are considered as generalized low-density parity-check codes to perform decoding. Exhaustive search for parity-check matrices of these codes is done to find ones as suitable as possible for iterative decoding. A new low-complexity decoding algorithm is proposed that uses multiple sub-decoders for the same code. It is shown that for a length 24 quasi-cyclic error-correcting code, this algorithm approaches the best achievable error rates and is competitive with previous results.

Keywords:
Decoding algorithms, error correction, generalized LDPC codes, quasi-cyclic LDPC codes.

CERCS: P170: Computer science, numerical analysis, systems, control

Parimate kvaasi-tsükliliste koodide vähekeerukas dekodeerimine

Lühikokkuvõte:

Veaparanduskoode kasutatakse laialdaselt tänapäeva kommunikatsioonis, et tuvastada ja parandada vigu, mis võivad tekkida digitaalse info edastamisel. Veaparanduskoodid aitavad tagada usaldusväärset infoedadsust, võimaldades vastuvõtjal taastada algse info, isegi kui mõned bitid edastamisel rikuti. Antud uurimistöö keskendub kvaasi-tsükliliste veaparanduskoodide dekodeerimisskeemide parandamisele. Selles töös antakse klassikaline teoreetiline raamistik veaparanduskoodide mõistmiseks. Parimaid tuntuid kvaasi-tsüklilisi koode vaadeldakse kui üldistatud hõredaid paarsuskontrolli koode ning neile otsitakse dekodeerimiseks hästi sobivaid paarsuskontrolli maatrikseid. Pakutakse välja uus dekodeerimisalgoritm, mis kasutab mitut alam-dekodeerijat. Näidatakse, et pikkusega 24 kvaasi-tsüklilise veaparanduskoodi puhul läheneb antud algoritm parimatele võimalikele veamääradele, mis on üldiselt saavutatavad suure ajalise keerukusega algoritmidega.

Võtmesõnad:
Decoding algorithms, error correction, generalized LDPC codes, quasi-cyclic LDPC codes.

CERCS: P170: Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Contents

1 Introduction

Error-correcting codes are needed whenever information is transmitted over noisy communication channels. The main idea of error-correcting codes is to add some redundancy to all transmitted messages, which allows for detection and correction of errors that may occur during transmission. This helps to protect integrity of all transmitted data. Error-correcting codes are used intensively in modern communications. Digital cellular networks use a number of error-correcting codes. Starting from 3G, the two most popular classes of codes are Turbo codes and LDPC codes, which are suitable for low-complexity iterative decoding. However, these codes are not best for given parameters. At the same time, the best known codes do not allow low-complexity decoding. Applying iterative methods to the best codes will be done. Let us introduce the framework that is used to construct error-correcting schemes, starting with Shannon's communication model.

1.1 Communications model

We will first introduce the Shannon's Communication Model. The communications system consists of an information source, an encoder, a modulator, a noisy channel, a demodulator, a decoder, and a destination. [Sha48]

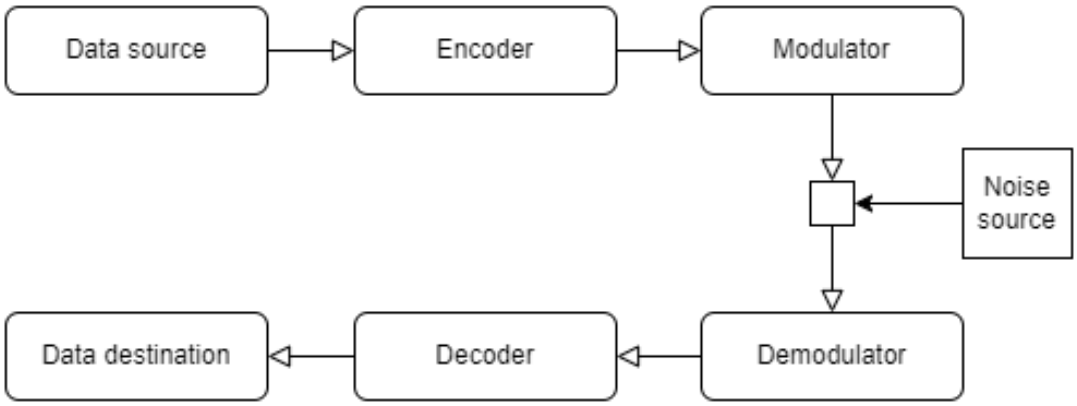

Figure 1. Shannon's Communication Model

1.2 Channel Models

Channel models provide a mathematical framework to describe the characteristics of different communication channels, which are often affected by various types of noise and interference. This chapter will focus on two widely studied channel models: the binary symmetric channel (BSC) and the additive white Gaussian noise (AWGN) channel. The

BSC is given as the most simple but important channel model that captures the errors introduced by a binary channel with a fixed probability of bit error. On the other hand, the AWGN channel is a channel model that introduces the notion of noise, which is present in many communication systems. Even the AWGN channel is a simplified model and does not fully capture the complex characteristics of real-world communication systems. In particular, real-world channels can exhibit a wide range of non-linear effects, such as fading, interference, and multipath propagation, that are not captured by the AWGN model. Despite its limitations, the AWGN model is still widely used as a benchmark for evaluating the performance of communication systems. This thesis uses the AWGN channel model to analyze decoding performance in all chapters that follow.

1.2.1 Binary Symmetric Channel

The simplest channel model is the **Binary Symmetric Channel**. The transmitter and receiver both have alphabet $\{0, 1\}$. There is an equal probability that noise will corrupt a transmitted zero to a one or a one to a zero.

$$p(0 \text{ received} \mid 1 \text{ transmitted}) = p(1 \text{ received} \mid 0 \text{ transmitted}) = p$$

p is called **crossover probability**.

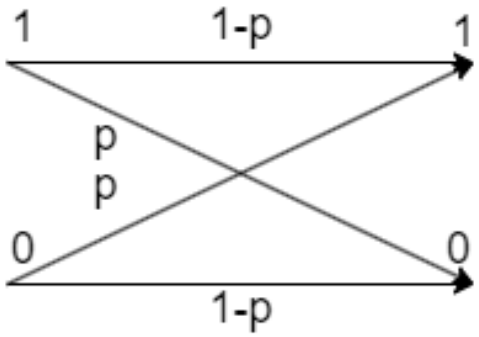

Figure 2. Binary Symmetric Channel

1.2.2 AWGN Channel

This thesis will focus on a discrete time channel with **AWGN** (additive white Gaussian noise) model. The model assumes that the signal is discrete in time, the noise is additive and white. Let $\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{n} \in \mathbb{R}^n$. Additive noise means that channel output $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{n}$ can be represented as a sum of two independent vectors: the transmitted message $\boldsymbol{x} = (x_1, ..., x_k)$ and the noise vector $\boldsymbol{n} = (n_1, ..., n_k)$. This channel is memoryless,

that is, each component $y_i = x_i + n_i$. White noise means that the noise has a flat power spectral density over all frequencies. The noise is called Gaussian because each component n_i of $\boldsymbol{n}$ is a random variable with Gaussian probability density function (PDF) having zero mean and variance σ^2. The probability density function for some expected value μ is of the form

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{\sigma^2}}$$

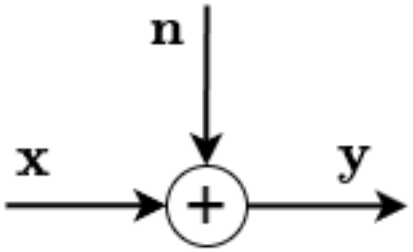

Figure 3. AWGN Channel

Adding noise to the signal degrades the quality of the signal, which makes it harder to decode. One way to measure the quality of the signal is by its **signal-to-noise ratio** (SNR). SNR is defined as the ratio of the power of the signal to the power of the noise. Mathematically, SNR is expressed in decibels (dB) as

$$\mathrm{SNR} = 10 \log_{10} \left(\frac{P_\mathrm{s}}{P_\mathrm{n}} \right),$$

where P_s is the signal power and P_n is the noise power. The logarithmic scale is used to express SNR in dB because it allows for a wider range of values to be expressed in a more manageable scale.

In the context of the Additive White Gaussian Noise (AWGN) channel, with noise being Gaussian and signals having average energy E, the SNR per signal is

$$\mathrm{SNR} = 10 \log_{10} \frac{E}{\sigma^2},$$

where E is the average energy per signal and σ is the noise variance. SNR is an important parameter in communication systems because it affects the performance of the system. A higher SNR means that the signal is stronger relative to the noise and can be more easily detected and decoded by the receiver. On the other hand, a lower SNR means that the signal is weaker relative to the noise and can be more difficult to detect and decode, which can result in errors in the decoded information.

The first chapter focuses on introducing all the necessary definitions and concepts related to linear codes. The second chapter derives the rules for maximum-likelihood decoding and the BCJR algorithm for symbol maximum a posteriori decoding. The third chapter of the thesis focuses on the class of low-density parity-check codes and the belief propagation algorithm that makes them viable. Generalizations of LDPC codes are explained and multi-base decoding is proposed to further improve the generalized belief propagation algorithm. Finally, we search for code representations suitable for low-complexity iterative decoding to use in multi-base decoding. The aforementioned ideas are combined into a new decoding algorithm. The final chapter presents results as plots of error rate as a function of the signal-to-noise ratio of the communication channel. The goal of this thesis is to approach the error rates of optimal decoding using low-complexity algorithms.

1.3 Overview of Literature and Related Works

1. "A Mathematical Theory of Communication" by Claude E. Shannon. Shannon's theory, introduced in his 1948 paper "A Mathematical Theory of Communication," established a mathematical framework for understanding communication and information transfer in systems that use symbols or signals. His work focused on how to measure the amount of information in a message and how to efficiently encode and transmit that information through a noisy channel. "A Mathematical Theory of Communication" is considered as a cornerstone for information theory. This thesis relies on the concepts and models introduced by Shannon.

2. "Low Density Parity Check Codes" by Robert Gallager. Gallager introduced the concept of LDPC codes, a then new class of error-correcting codes that can achieve near-Shannon limit performance with low decoding complexity. Published in 1963, the thesis later became a monumental work in the field of coding theory and inspired a new wave of research in the development of practical applications in digital communication and storage systems. In this thesis, Gallager's ideas are used to decode codes with higher density than he intended.

3. "Tailbiting Codes: Bounds and Search Results" by Irina E. Bocharova, Rolf Johannesson, Boris D. Kudryashov, Per Ståhl The codes presented in this paper are the best quasi-cyclic codes for given lengths. Because of this, they are used as a starting point for search of good code representations in this thesis.

4. "Multiple-Bases Belief-Propagation for Decoding of Short Block Codes" by Thorsten Hehn, Johannes B. Huber, Stefan Laendnert, Olgica Milenkovict. This paper introduces a simple multiple base belief propagation decoding algorithm. This thesis will try to make some improvements and also use the algorithm they propose as a baseline for comparison.

2 Linear Codes

Definition 2.1 (Error-Correcting Code)**.** An error-correcting code C is a set of vectors over some finite alphabet $\mathbb{F}$. Elements of C are called **codewords**.

Definition 2.2 (Linear Subspace)**.** $C \subset A$ is a linear subspace of vector space A over finite field $\mathbb{F}$, iff

1. C is closed under addition, meaning that if $c \in C$ and $d \in C$, then $c + d \in C$.
2. C is closed under scalar multiplication, meaning that if $c \in C$ and $a \in \mathbb{F}$, then $a \cdot c \in C$.

Definition 2.3 (Linear Code)**.** Let $\mathbb{F} = (\Sigma, +, \cdot)$ be a finite field. Any $C \subset \mathbb{F}^n$ that is a k-dimensional subspace of $\mathbb{F}^n$ is called a $[n, k]$ **linear code**. We call n code length and k code dimension.

2.1 Code Parameters

A code can be characterized by a certain set of parameters. This thesis will focus on binary codes. This means that the alphabet for codes examined in this paper is $\mathbb{F}_2 = \{0, 1\}$. The transmitted messages $\boldsymbol{x} = (x_1, x_2, ..., x_k)$ are vectors of length k, which are encoded to codewords $\boldsymbol{c} = (c_1, c_2, ..., c_n)$, vectors of length n.

2.1.1 Minimum Distance

Definition 2.4 (Hamming distance)**.** The Hamming distance between two binary vectors is the number of coordinates where the vectors don't coincide. It is denoted by $d(\boldsymbol{x}, \boldsymbol{y})$. The number of 1-s in a binary vector $\boldsymbol{x}$, denoted by $w(\boldsymbol{x})$ is called the **Hamming weight** of $\boldsymbol{x}$

For example, if $\boldsymbol{x} = (0, 0, 0, 1, 0)$, $\boldsymbol{y} = (1, 1, 0, 0, 0)$, then the Hamming distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ is $d(\boldsymbol{x}, \boldsymbol{y}) = 3$. The vectors don't coincide on indices $0, 1, 3$. It is clear that in binary arithmetic, $d(\boldsymbol{x}, \boldsymbol{y}) = w(\boldsymbol{x} + \boldsymbol{y})$. $+$ is interpreted as symbol-wise modulo 2 addition.

Definition 2.5 (Minimum distance)**.** The minimum distance of a code is the minimum of the Hamming distances of pairs of distinct codewords. In other words, let the set of codewords be C, then the minimum distance is $d = \min\{d(\boldsymbol{x}, \boldsymbol{y}) \mid \boldsymbol{x}, \boldsymbol{y} \in C, \boldsymbol{x} \neq \boldsymbol{y}\}$.

The minimum distance is generally a good measure of a code error correcting capability, because when two codewords have a small Hamming distance from each other, a smaller number of bit changes during transmission results in a decoding error. If C is a linear code with codeword length n, code dimension k and minimum distance d, it is called a $[n, k, d]$ **code** over $\mathbb{F}$.

2.1.2 Code Rate

The code rate characterizes the efficiency of information transmission. The simplest possible solution to increase reliability of information transmission over a noisy channel is repeating the message m times. Decoding such a code is choosing the most common symbol for all positions among all repetitions of the message. This process resembles having the copies of the transmitted message vote for a symbol on any given position. The major drawback of this solution is that the **code rate**, the ratio of the number of bits of useful information to the total number of transmitted bits, is $\frac{1}{m}$. As m increases, $\frac{1}{m}$ decreases. There are far more effective codes with higher code rates.

Definition 2.6 (Code rate)**.** For linear codes, the **code rate** is the ratio of code dimension k to code length n. We usually denote code rate by $R = \frac{k}{n}$.

2.2 Generator Matrix

Definition 2.7 (Generator matrix)**.** Let C be a $[n, k, d]$ linear code over a finite field $\mathbb{F}$. A $k \times n$ matrix G is a **generator matrix** of C if its rows form a basis of C. That is, any codeword $\boldsymbol{c}$ in C can be expressed as a linear combination of the rows of $G = (\boldsymbol{g_1}, \ldots \boldsymbol{g_n})^T$, i.e., $\boldsymbol{c} = a_1\boldsymbol{g_1} + a_2\boldsymbol{g_2} + ... + a_k\boldsymbol{g_k}$, where $a_1, a_2, ..., a_k$ are elements of $\mathbb{F}$.

The generator matrix G is often used to encode data into a codeword. To encode a message m, we multiply the message by G to get the codeword $\boldsymbol{c} = \boldsymbol{m}G$. This operation maps the k-dimensional message space onto the n-dimensional codeword space.

2.3 Parity-Check Matrix

Definition 2.8 (Parity-check matrix)**.** Let C be a $[n, k, d]$ linear code over some finite field $\mathbb{F}$. The **parity check matrix** $H = (\mathbf{h_1}, ..., \mathbf{h_n})^T$ of C is an $(n - k) \times n$ matrix whose rows span the orthogonal complement of C. That is, for any codeword $\boldsymbol{c} \in C$ and any row $\mathbf{h_i}$ of H, we have the scalar product $\langle \boldsymbol{h_i}, \boldsymbol{c} \rangle = 0$. The scalar product of vectors $\boldsymbol{x}, \boldsymbol{y}$ is defined as $\sum_j x_j y_j$. Equivalently, H is a matrix such that $\boldsymbol{c}H^T = 0$ for all $c \in C$. It follows that $GH^T = 0$.

The parity check matrix H is often used to decode channel output data. To check whether a received vector $\boldsymbol{y}$ is a codeword, we compute the **syndrome** $\boldsymbol{s} = \boldsymbol{y}H^T$. If the syndrome is all zeros, then $\boldsymbol{y}$ is a codeword of C. We can construct G from H by using linear row operations to obtain form

$$G = (I_k P).$$

Then

$$H = (P^T I_r).$$

It is clear that $GH^T = \mathbf{0}$

2.4 Quasi-Cyclic Codes

In practice, **quasi-cyclic** codes are often used, because they are known to be as good as arbitrary linear codes and because they are convenient to store.

Definition 2.9. Linear code is called l-**quasi-cyclic (QC)** if for some $l \in \mathbb{N}$ a cyclic shift of a codeword by l positions is a codeword of the same code.

2.4.1 Example of QC Code

This generator matrix for a rate $R = \frac{1}{2}$ code is constructed by cyclically shifting the first row by 2 positions. It is possible to check that code defined by G is QC.

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

3 Optimal Decoding

This chapter examines the maximum-likelihood (ML) and symbol-wise maximum a posteriori (symbol MAP) decoding principles. Let us introduce two closely related metrics to measure the error-correcting capability of codes.

Definition 3.1. The frame error rate (FER) of a communications channel is the ratio of decoder mistakes to the number of decoding attempts. If N codewords are transmitted and e of them are decoded incorrectly (decoder output is not the same as encoder output), then

$$FER = \frac{e}{N}.$$

Definition 3.2. The bit error rate (BER) of a communications channel is the ratio of incorrectly decoded bits to the total number of bits transmitted. If N codewords of length n are transmitted and e number of bit errors were made (decoder output has e different bits than encoder output), then

$$BER = \frac{e}{Nn}.$$

ML decoding is optimal in terms of FER and symbol MAP decoding is optimal in terms of BER. We call ML and symbol MAP decoding **optimal decoding**. Because higher levels of noise result in more decoding errors, we evaluate the capabilities of decoding schemes by giving their FER and BER as a function of SNR.

3.1 Maximum Likelihood Decoding

Let C be a $[n, k, d]$ linear code over finite field $\mathbb{F}$. Maximum likelihood (ML) decoding is a decoding technique that aims to find a codeword $\boldsymbol{c}$ maximizing probability of channel output $\boldsymbol{y}$ given $\boldsymbol{c}$ was transmitted. We will denote the probability $p(\boldsymbol{y}$ received $\mid$ $\boldsymbol{c}$ transmitted) as $p(\boldsymbol{y} \mid \boldsymbol{c})$. A maximum likelihood decoder outputs $\boldsymbol{c} \in C$ so that $p(\boldsymbol{y} \mid \boldsymbol{c})$ is maximized. Denote the output of a maximum-likelihood decoder given channel output $\boldsymbol{y}$ by $\hat{x}(\boldsymbol{y})$. If the channel is discrete and memoryless:

$$\hat{x}(\boldsymbol{y}) = \arg\max_{\boldsymbol{c} \in C} p(\boldsymbol{y} \mid \boldsymbol{c}) = \prod_{i=1}^{n} p(y_i \mid c_i).$$

Since $\ln(x)$ is a monotonic function of x

$$\arg\max_{\boldsymbol{c} \in C} p(\boldsymbol{y} \mid \boldsymbol{c}) = \arg\max_{\boldsymbol{c} \in C} \ln p(\boldsymbol{y} \mid \boldsymbol{c}) = \arg\max_{\boldsymbol{c} \in C} \sum_{i=1}^{n} \ln p(y_i \mid c_i).$$

If binary phase-shift keying (BPSK) is used for modulation, the conditional probability density function for each transmitted symbol c_i is

$$f(y_i \mid c_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y_i - b_i)^2}{2\sigma^2}},$$

where $b_i = (1 - 2c_i)\sqrt{E}$ and E is signal energy. We get

$$\ln p(\boldsymbol{y} \mid \boldsymbol{c}) = -\frac{\|\boldsymbol{y} - \boldsymbol{b}\|^2}{2\sigma^2} + n \ln \frac{1}{\sigma\sqrt{2\pi}},$$

where $\|\boldsymbol{y} - \boldsymbol{b}\|^2$ denotes the squared Euclidean distance between vectors $\boldsymbol{y}$ and $\boldsymbol{b}$. Because $n \ln \frac{1}{\sigma\sqrt{2\pi}}$ is constant for all channel outputs and because $\frac{1}{-2\sigma^2}$ is always negative

$$\arg\max_{\boldsymbol{c} \in C} \ln p(\boldsymbol{y} \mid \boldsymbol{c}) = \underset{\boldsymbol{c} \in C}{argmin} \|\boldsymbol{y} - \boldsymbol{b}\|^2 = \underset{\boldsymbol{c} \in C}{argmin} \sum_i (y_i - b_i)^2.$$

Therefore maximum-likelihood decoding for AWGN channel with BPSK modulation is equivalent to finding the codeword $\boldsymbol{c}$ which has smallest Euclidean distance from channel output $\boldsymbol{y}$.

Further simplifications can be made when we use modulation such that all signals have the same energy E, as is the case for BPSK modulation:

$$\underset{\boldsymbol{c} \in C}{argmin} \sum_i (y_i - b_i)^2 = \underset{\boldsymbol{c} \in C}{argmin} \sum_i (y_i^2 - 2y_i b_i + b_i^2) =$$

$$= \underset{\boldsymbol{c} \in C}{argmin} \left(\|\boldsymbol{y}\| + \|\boldsymbol{b}\| - \sum_i (2y_i b_i)\right) = \arg\max_{\boldsymbol{c} \in C} \langle \boldsymbol{y}, \boldsymbol{b} \rangle.$$

This means ML decoding for AWGN channel with BPSK modulation is finding the codeword $\boldsymbol{c}$ which has the largest scalar product with channel output $\boldsymbol{y}$. The time complexity of scalar product ML decoding is that of exhaustive search over all 2^k keywords.

3.2 Symbol MAP Decoding

The symbol a posteriori probabilty for symbol c in position i given channel output $\boldsymbol{y}$ is

$$p(c_i = c \mid \boldsymbol{y}) = \frac{p(c_i = c, y)}{p(\boldsymbol{y})}$$

where $p(c_i = c, \boldsymbol{y}) = \sum_{\boldsymbol{c} \in C_i} p(\boldsymbol{c}, \boldsymbol{y})$ where $C_i(c)$ is the set of codewords which have c in position i.

3.3 Trellis Diagrams

Due to exponential complexity of optimal decoding, it becomes unfeasible as code length grows. The search for lower complexity of decoding leads us to labelled digraphs called **trellises** or trellis diagrams. Furthermore, trellis based decoding called BCJR decoding is used as a sub-algorithm in chapter 4.2

Definition 3.3. $G = (V, \mathcal{E}, L)$ is a labeled directed graph. V is the set of nodes, $\mathcal{E}$ is the set of directed edges, L is the labelling $L : \mathcal{E} \to \mathbb{F}$, where $\mathbb{F}$ is a finite alphabet. G is a **trellis** or a trellis diagram if

1. there exists a set of subsets of $P = \{V_1, ..., V_l\}$ such that $\bigcup_{i=1}^{l} V_i = V$. V_i are non-empty and non-intersecting subsets of nodes. The set V_i is called the i-th level of the trellis;

2. edges exist only between neighbouring levels. Futhermore, they are directed from the i-th level to the $i+1$-st. $\forall uv \in E : \ u \in V_i \Rightarrow v \in V_{i+1}$;

3. the initial and the last levels contain only one node each $\mid V_1 \mid = \mid V_l \mid = 1$.

The first node $v_1 \in V_1$ is called the $root$ and the last node $v_s \in V_l$ is called the $toor$. We call the nodes of a trellis diagram **states**.

Definition 3.4 (State complexity)**.** The state complexity ν of a trellis diagram is

$$\nu = \max_{i=1,...,l} \log_2 \mid V_i \mid$$

Paths of a code trellis are codewords. It is possible to construct a trellis for linear codes such that every labelled path from $root \in V_1$ to $toor \in V_l$ is a codeword.

3.3.1 Example of Trellis Diagram

Let linear code $C = \{(1,1,0), (0,1,1), (0,0,0), (1,0,1)\}$ be defined by its generator matrix G.

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

Then the corresponding trellis is

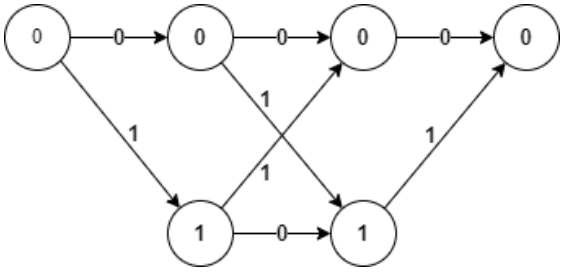

Figure 4. Trellis

It is easy to check that every path from root to toor corresponds to a codeword. Every linear code has an unique trellis representation such that the state complexity is minimal [BCJR74].

3.4 BCJR algorithm

The BCJR algorithm is used for trellis-based symbol MAP decoding. It was named after L. Bahl, J. Cocke, F. Jelinek, and J. Raviv after they published it in [BCJR74]. BCJR is a soft-input soft-output (SISO) algorithm. It receives input $\boldsymbol{y} \in \mathbb{R}^n$ and outputs a vector of symbol-wise LLRs (log-likelihood ratios of the code symbols)

$$\boldsymbol{\lambda} = \left(\ln \frac{p(c_0 = 1 \mid \boldsymbol{y})}{p(c_0 = 0 \mid \boldsymbol{y})}, \ln \frac{p(c_1 = 1 \mid \boldsymbol{y})}{p(c_1 = 0 \mid \boldsymbol{y})}, \ldots, \ln \frac{p(c_n = 1 \mid \boldsymbol{y})}{p(c_n = 0 \mid \boldsymbol{y})} \right).$$

We know that the a posteriori probability of symbol $c \in \{0, 1\}$ at position t is given by $\boldsymbol{y}$ is

$$p(c_t = c \mid \boldsymbol{y}) = \frac{p(c_t = c, y)}{p(\boldsymbol{y})},$$

where $p(c_i = c, \boldsymbol{y}) = \sum\limits_{\boldsymbol{c} \in C_i} p(\boldsymbol{c}, \boldsymbol{y})$, where $C_i(c)$ is the set of codewords which have c in position i. We replace summation over codewords by summation over trellis states

$$p(c_t, \boldsymbol{y}) = \sum_{(m'm) \in S_t(c)} p(s_{t-1} = m', s_t = m, \boldsymbol{y}),$$

where s_t is a trellis state at level t, $S_t(c)$ is a set of pairs of states such that

$$\forall (m'm) \in S_t(c)\ L(m'm) = c,$$

meaning (m', m) is a pair of trellis states at levels $t-1$ and t corresponding to code symbol c. Then the LLR for symbol at position t using the notation $\sigma_t(m', m) = p(s_{t-1} = m', s_t = m, \boldsymbol{y})$ becomes

$$\lambda_t = \ln \frac{\sum_{(m'm) \in S_t(1)} p(s_{t-1} = m', s_t = m, \boldsymbol{y})}{\sum_{(m'm) \in S_t(0)} p(s_{t-1} = m', s_t = m, \boldsymbol{y})} = \ln \frac{\sum_{(m'm) \in S_t(1)} \sigma_t(m', m)}{\sum_{(m'm) \in S_t(0)} \sigma_t(m', m)}.$$

We use the denotation $\boldsymbol{y}_i^j = (y_i, y_{i+1}, \ldots, y_j)$ for the part of $\boldsymbol{y}$ that is between indices i and j. We will now decompose $p(s_{t-1} = m', s_t = m, \boldsymbol{y})$ into three events:

- A_1: The state at level $t-1$ is m' and the beginning of the received sequence is $\boldsymbol{y}_1^{t-1}$;
- A_2: The received symbol on index t is y_t and the state on level t is m;
- A_3: The end of the received sequence is $\boldsymbol{y}_{t+1}^n$.

Using the conditional probability formulas gives

$$Pr(A_1A_2A_3) = Pr(A_1)Pr(A_2 \mid A_1)Pr(A_3 \mid A_1A_2).$$

For memoryless channels we assume that symbols of the received vector $\boldsymbol{y}$ are conditionally independent.

$$\begin{aligned}
Pr(A_1) &= p(s_{t-1} = m', \boldsymbol{y}_1^{t-1}) \\
Pr(A_2 \mid A_1) &= p(s_t = m, y_t \mid s_{t-1} = m', \boldsymbol{y}_1^{t-1}) && (1)\\
&= p(s_t = m, y_t \mid s_{t-1} = m') && (2)\\
Pr(A_3 \mid A_1A_2) &= p(\boldsymbol{y}_{t+1}^n \mid s_{t-1} = m', \boldsymbol{y}_1^{t-1}, s_t = m, y_t), \\
&= p(\boldsymbol{y}_{t+1}^n \mid s_{t-1} = m', \boldsymbol{y}_1^{t}, s_t = m) && (3)\\
&= p(\boldsymbol{y}_{t+1}^n \mid s_t = m), && (4)
\end{aligned}$$

where $(1) \equiv (2)$ because y_t does not depend on $\boldsymbol{y}_1^{t-1}$ and $\boldsymbol{y}_1^{t-1}$ does not influence the event $s_t = m$ when $s_{t-1} = m$ is known. $(3) \equiv (4)$ because neither $s_{t-1} = m'$ or $\boldsymbol{y}_1^t$ influences the next symbols $\boldsymbol{y}_{t+1}^n$ We introduce the notation

$$\begin{aligned}
\alpha_t(m) &:= Pr(A_1) = p(s_t = m, \boldsymbol{y}_1^t), \\
\gamma_t(m', m) &:= Pr(A_2 \mid A_1) = p(s_t = m, y_t \mid s_{t-1} = m'), \\
\beta_t(m) &:= Pr(A_3 \mid A_1A_2) = p(\boldsymbol{y}_{t+1}^n \mid s_t = m).
\end{aligned}$$

We can write $\sigma_t(m', m) = \alpha_{t-1}(m')\gamma_t(m', m)\beta_t(m)$. From the law of total probability we get

$$\alpha_t(m) = \sum_{m'} p(s_{t-1} = m', \boldsymbol{y}_1^{t-1}) p(s_t = m, y_t \mid s_{t-1} = m', \boldsymbol{y}_1^{t-1}).$$

After omitting $\boldsymbol{y}_1^{t-1}$ when s_{t-1} is given we obtain the *forward recursion*

$$\alpha_t(m) = \sum_{m'} \alpha_{t-1}(m')\gamma_t(m', m).$$

We define

$$\alpha_0(m) = \begin{cases} 1, & \text{if } m = 0; \\ 0, & \text{otherwise.} \end{cases}$$

Similarly it is possible to obtain the *backward recursion*

$$\beta_t(m) = \sum_{m'} \beta_{t+1}(m')\gamma_{t+1}(m', m).$$

with the initial conditions

$$\beta_n(m) = \begin{cases} 1, & \text{if } m = 0; \\ 0, & \text{otherwise.} \end{cases}$$

We can calculate the edge metric γ as

$$\gamma_t(m', m) = \sum_{c_t} Pr(s_t = m, c_t, y_t \mid s_{t-1} = m) = \sum_{c_t} p(c_t \mid m, m')p(y_t \mid c_t) =$$

$$= p(c_{t,m,m'})p(y_t \mid c_{t,m,m'}),$$

where $c_{t,m,m'}$ is the code symbol associated with edge $m' \to m$ at layer t. Define $\gamma_t(m', m) = 0$ for those (m', m) which are not connected in the trellis. We have now derived recursive formulas which we can use to calculate $\lambda(c_t)$ for every position. The absolute value $\mid \lambda(c_t) \mid$ is called the reliability of the symbol. We can obtain hard decision of BCJR decoder $\hat{c_t}$ by

$$\hat{c_t} = \begin{cases} 1, & \text{if } \lambda_t \geq 0; \\ 0, & \text{if } \lambda_t \leq 0. \end{cases}$$

Note that if $\lambda_t = 0$ a decision is made randomly. Pseudo-code of the BCJR algorithm is presented as Algorithm 1.

Algorithm 1: BCJR decoding algorithm

Input: The received sequence $\boldsymbol{y}$ a priori probabilities $p(y_t \mid c_t)$ of symbols $c_1, ..., c_n$

Result: LLRs $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$

Initialization: For layers $1, \ldots n$ and pairs of states (m', m) compute

$$\sum_{m'} \alpha_{t-1}(m')\gamma_t(m', m)$$

foreach *layer from* 0 *to* n **do**
  foreach *state* m *of the layer* **do**
    calculate $\alpha_t(m) = \sum_{m'} \alpha_{t-1}(m')\gamma_t(m', m)$

foreach *layer from* n *to* 0 **do**
  foreach *state* m *of the layer* **do**
    calculate $\beta_t(m) = \sum_{m'} \beta_{t+1}(m')\gamma_{t+1}(m', m)$

foreach *layer* V_t *of trellis* **do**
  foreach *pair of states* (m', m), $m' \in V_{t-1}$, $m \in V_t$ *of the layer* **do**
    calculate $\sigma_t(m', m) = \alpha_{t-1}(m')\gamma_t(m', m)\beta_t(m)$

foreach *layer* t *from 1 to* n **do**
  calculate $\lambda_t = \ln \dfrac{\sum_{(m'm)\in S_t(1)} \sigma_t(m', m)}{\sum_{(m'm)\in S_t(0)} \sigma_t(m', m)}$

4 Low-Density Parity-Check (LDPC) Codes

LDPC codes were first introduced in [Gal62]. Despite not having the best minimum distance for given code length n, LDPC codes have good error-correcting capability. Gallager proposed low-complexity iterative decoding of LDPC codes. It is called belief propagation (BP) decoding. The BP decoding principle can be explained by using graph representations of parity-check matrices called **Tanner graphs**.

Definition 4.1. Low-density parity-check (LDPC) codes are codes whose parity-check matrix H has relatively few 1-s and many 0-s.

Definition 4.2. A binary linear $[n, k]$-code determined by a parity-check matrix H is called a (J, K) regular LDPC code if each column of H contains J ones and each row contains K ones.

Definition 4.3. The Tanner graph of a linear code determined by the parity-check matrix $H = \{h_{ij}\}, i = 1, \ldots, r, j = 1, \ldots, n$ is a bipartite graph whose one set of nodes corresponds to the checks of H (check nodes) and the other set of nodes corresponds to the set of code symbols (variable nodes). A check node c_i is connected with a variable node v_j if $h_{ij} = 1$.

4.1 Example of LDPC code

Let us consider a $(2, 4)$ regular binary $[8, 4]$ LDPC code $\mathbb{C}$ defined by it is parity-check matrix

$$H = \begin{pmatrix} v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & c_0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & c_1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & c_2 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & c_3 \end{pmatrix}.$$

Since every codeword $\boldsymbol{c}$ has to satisfy $\boldsymbol{c}H^T = \mathbf{0}$ we can also consider H as the following set of equations in binary field $\mathbb{F}_2$. The corresponding Tanner graph is given in Figure 5.

$$\begin{cases} v_1 + v_2 + v_3 + v_4 &= 0 \\ v_2 + v_3 + v_4 + v_5 &= 0 \\ v_4 + v_5 + v_6 + v_7 &= 0 \\ v_0 + v_1 + v_6 + v_7 &= 0 \end{cases}$$

The main advantage of LDPC codes, is that the **belief propagation** algorithm can be used for decoding. The belief propagation algorithm time complexity is proportional

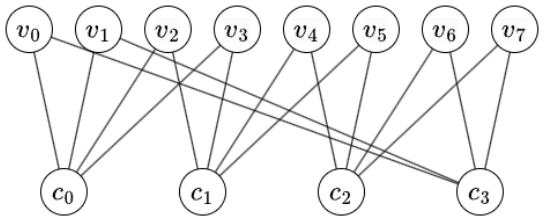

Figure 5. Tanner graph of $(2, 4)$ regular binary $[8, 4]$ LDPC code

to code length [Gal62]. The main idea of belief propagation decoding is to consider the rows of parity-check matrix H as parity-check matrices of single parity-check codes and calculate symbol probabilities based on the fact that every parity check has to be satisfied for any codeword. It would be possible to construct a single-row trellis for each row to make MAP decisions for each symbol, but it is possible to simplify further. Belief propagation is not MAP decoding, because it assumes that all single parity check rows are independent, meaning there are no cycles in the Tanner graph. First introduce the following notation:

- S_d is the event that the d-th check is satisfied.
- Let $p_i = Pr(x_i = 1 \mid \boldsymbol{y})$ be the probability that the i-th symbol is a 1, conditional on the received sequence $\boldsymbol{y}$.
- Let $p_{jh} = Pr(x_h = 1 \mid \boldsymbol{y}, S_j)$ be the probability of 1 in position h of the jth check given received vector $\boldsymbol{y}$ and event that check S_j is satisfied.
- K is the number of ones in each row of the parity check matrix.

We will give the following theorem from [Gal62] without proof.

Theorem 1. If checks containing the symbol x_i do not overlap except position i

$$\frac{P(x_i = 0 \mid \boldsymbol{y}, S)}{P(x_i = 1 \mid \boldsymbol{y}, S)} = \frac{1 - p_i}{p_i} \prod_{j=1}^{i} \frac{1 + \prod_{h=1, h\neq i}^{K} (1 - 2p_{jh})}{1 - \prod_{h=1, h\neq i}^{K} (1 - 2p_{jh})}$$

Decoding based on this formula is called the sum-product algorithm (SPA), but only relatively short LDPC codes can be decoded using the SPA, because the required arithmetic precision increases with code length. [Gal62]. The sum-product decoding algorithm can be interpreted as iterative exchange of probabilities (LLRs) between variable and check nodes of the Tanner graph. This problem can be solved by reformulating the problem in terms of LLRs. We introduce the following notations:

- Let α_i be the sign of the LLR of the ith symbol.

- Let β_i be the absolute of the LLR of the ith symbol.
- Let $f(\beta) = \ln \frac{e^\beta+1}{e^\beta-1} = -\ln(\tanh(\frac{\beta}{2})), \beta > 0$.
- $\tanh(x) = \frac{e^x-e^{-x}}{e^x+e^{-x}} = \frac{e^{2x}-1}{e^{2x}+1}$ is the hyperbolic tangent.

If $p = Pr(x_i = 1)$ then the corresponding LLR is $L_i = \ln \frac{1-p}{p} = \alpha_i\beta_i$. Simplifying the expression $1 - 2p$ that appears on the right-hand side of 1 gives

$$1 - 2p = \frac{(1-p)/p - 1}{(1-p)+1} = \alpha \tanh \beta/2 = -\alpha e^{(f(\beta))}.$$

Taking logarithm of both sides of (1) gives

$$LLR(x_i) = L_i + \sum_{i=1}^{J} \ln \frac{\prod_{h=1,h\neq i}^{K} \alpha_{jh} e^{f(\beta_{jh})} + 1}{\prod_{h=1,h\neq i}^{K} \alpha_{jh} e^{f(\beta_{jh})} + 1} =$$

$$= L_i + \sum_{i=1}^{J} \left(\prod_{h=1,h\neq i}^{K} \alpha_{jh} \right) f \left(\sum_{h=1,h\neq i}^{K} f(\beta_{jh}) \right).$$

Let α_i' and β_i' denote the sign and absolute value of $LLR(x_i)$ calculated on the next iteration of the algorithm. Then

$$\alpha_i'\beta_i' = \alpha_i\beta_i + \sum_{i=1}^{J} \left(\prod_{h=1,h\neq i}^{K} \alpha_{jh} \right) f \left(\sum_{h=1,h\neq i}^{K} f(\beta_{jh}) \right) \tag{5}$$

The general idea of the algorithm is as follows:

1. Calculate LLR for every symbol based on channel output $\boldsymbol{y}$.
2. Update LLRs based on (5).
3. Make hard decisions based on LLRs, if the resulting vector is a codeword, stop and return. Otherwise go to 2.

We will now give the pseudocode of the sum-product algorithm in logarithmic domain as Algorithm 2.

4.1.1 Sum-Product Algorithm Pseudocode

Algorithm 2: SPA in logarithmic domain

Input: Symbol LLRs $\boldsymbol{l} = (l_1, \ldots, l_n)$ computed from channel output $\boldsymbol{y}$, parity-check matrix H

Result: Decoder output $\hat{\boldsymbol{x}}$

Initialization: $L_{ji}^{(0)} = l_j, t = 0$

while $t < MaxIter$ **do**

  / Horisontal step. Every iteration estimates LLR of x_h based on row i:*

  foreach $i = 1$ *to* r **do**

$$\alpha_{hi}^{(t-1)} = sign\left(L_{hi}\right)$$

$$Z_{ij} = \left(\prod_{h \neq j} \alpha_{hi}^{(t-1)}\right) f\left(\sum_{h=1, h\neq j^K} f\left(\mid L_{hi}^{(t-1)} \mid\right)\right)$$

  / Vertical step*

  foreach $j = 1$ *to* n **do**

$$L_{ji}^{t} = l_j + \sum_{k=1, k\neq i}^{J} Z_{kj}^{(t)}$$

$$L_{j}^{t} = l_j + \sum_{k=1}^{J} Z_{kj}^{(t)}$$

    / Make hard decision $\hat{x}_j$*

    if l_j*<0* **then**

      $\hat{x}_j = 1$

    else

      $\hat{x}_j = 0$

  Calculate syndrome $\boldsymbol{s} = \hat{\boldsymbol{x}} H^T$

  if $\boldsymbol{s} == \boldsymbol{0}$ **then**

    break

  $t = t + 1$

return $\hat{\boldsymbol{x}}$

4.2 Generalized Low-Density Parity-Check (GLDPC) Codes

Definition 4.4 (Generalized Low-Density Parity-Check (GLDPC) codes)**.** LDPC codes whose row constituent codes are linear codes which are not limited to single-parity-check codes are called generalized LDPC (GLDPC) codes. In other words, each row of the LDPC code parity-check matrix is replaced with a few rows of parity-check matrix of a constituent code. GLDPC codes were introduced in [Tan81].

The main idea behind the decoding of GLDPC codes is to combine the BCJR and BP as sub-algorithms by replacing single row parity-check codes in the BP algorithm with constituent codes not restricted to single parity-check codes. Let $\mathbb{C}$ be a linear code determined by it is $r \times n$ parity-check matrix H. We will now consider some number m (such that $m \mid r$) of consecutive rows as parity-check matrices H_i and represent H in the form

$$H = \begin{pmatrix} H_1 \\ H_2 \\ \dots \\ H_{\frac{r}{m}} \end{pmatrix}$$

The generalized BP algorithm executes the BCJR algorithm on the parity-check matrix H_i instead of single row parity-check matrices. H_i are called the constituent codes of H. For all H_i the horizontal step of BP decoding is performed as BCJR decoding but due to very simple parity-check matrix structure the BCJR implementation is simple. In the case of GLDPC, we perform standard BCJR decoding on the trellis of constituent code. Every iteration, the vertical step combines the LLR outputs of each constituent code as Algorithm 2. We call the number of the rows of the constituent code the **strip length** of the GLDPC code. This algorithm is a trade-off between complexity and error probability. The time complexity is proportional to 2^{ν} where ν is the maximum state complexity of constituent code trellises. This allows us to approach the error-correcting performance of MAP decoding for larger parity-check matrices for which genuine ML or MAP decoding is infeasible. This thesis studies iterative decoding of codes that are generally considered too dense for BP decoding. The main idea behind the proposed approach is to interpret them as GLDPC codes. We aim to approach optimal decoding performance by adding different decoders for the same code, thus leveraging the strengths of each individual decoder.

4.3 Example of GLDPC Code

This is the generator matrix of the QC $[24, 12, 8]$ code given in [BJKS02].

$$G = \left(\begin{array}{c} 010010101110110000000000 \\ 000100101011101100000000 \\ 000001001010111011000000 \\ 000000100101011101100000 \\ \hline 000000000100101011101100 \\ 000000000001001010111011 \\ 110000000000010010101110 \\ 101100000000000100101011 \\ \hline 111011000000000001001010 \\ 101110110000000000010010 \\ 101011101100000000000100 \\ 001010111011000000000001 \end{array}\right)$$

Consider it as a GLDPC code with strip length 4, as shown by the horizontal lines. One iteration of the generalized BP algorithm would be as follows: Horisontal step: Perform BCJR for constituent codes

$$G_1 = \begin{pmatrix} 010010101110110000000000 \\ 000100101011101100000000 \\ 000001001010111011000000 \\ 000000100101011101100000 \end{pmatrix}$$

$$G_2 = \begin{pmatrix} 000000000100101011101100 \\ 000000000001001010111011 \\ 110000000000010010101110 \\ 101100000000000100101011 \end{pmatrix}$$

$$G_3 = \begin{pmatrix} 111011000000000001001010 \\ 101110110000000000010010 \\ 101011101100000000000100 \\ 001010111011000000000001 \end{pmatrix}$$

Obtain three corresponding soft outputs $\boldsymbol{g_1}, \boldsymbol{g_2}, \boldsymbol{g_3}$ and combine them as per the vertical step of Algorithm 2.

4.4 Multi-Base Decoding

Multi-base (MB) decoding is the general term for decoding schemes where more than one decoder is used. Notice that for linear code C the generator matrix G nor the parity-check matrix H is unique. Any set of k linearly independent codewords can be used as a generator matrix as they form a basis of the same linear subspace. The same is valid for parity-check matrices, because the rows of H are $r = n - k$ linearly independent vectors that form the basis for the orthogonal complement of C, which is also a subspace of $\mathbb{F}^n$. It is possible for one decoder to fail for some channel output $\boldsymbol{y}$ and another one to succeed. The structure of a multi-base decoder is shown in Figure 5. The informal description of a multi-base decoder M containing sub-decoders $D_1, \ldots, D_s$ is as follows:

- M contains s decoders $D_1, \ldots, D_s$ that determine the same code.
- For channel output $\hat{\boldsymbol{y}}$ decoder M performs decoding in each of the s sub-decoders.
- Using sub-decoder outputs $\boldsymbol{x_1}, \ldots, \boldsymbol{x_s}$ for decoding result $\hat{\boldsymbol{x}}$.

The important characteristics of a multi-base decoder are

- Decoding algorithms used in $D_1, \ldots, D_s$;
- Number of decoders;
- The way in which sub-decoder outputs are combined to make decision about the decoder output.

We will present the pseudo-code of multi-base decoder given in [HHLM07] as Algorithm 3. It uses ordinary BP decoders for sub-decoders, remembers which of the sub-decoder outputs are valid codewords and chooses the sub-decoder output which has the smallest squared Euclidean distance from the channel output. It will be used as a reference point to measure obtained results. Denote by $D_i(\boldsymbol{y})$ the output of decoder D_i on input $\boldsymbol{y}$ by and by S the set of indices of those decoders with outputs that are valid codewords.

The multi-base decoder proposed for QC codes in this thesis uses the generalized BP algorithm in decoders $D_1, \ldots, D_s$. The parity-check matrices used in $D_1, \ldots, D_s$ are ranked based on empirically measuring error-correcting performance of individual decoders. Decoding will start with the best individual decoder D_1 and only run the next decoder D_{i+1} if D_i didn't return a codeword. The justification for this is that generalized BP decoders very rarely output the wrong codeword - they either don't converge to a codeword or converge to the transmitted one using this. If none of the sub-decoders output a valid codeword, we make hard decisions about sub-decoder outputs, calculate their Euclidean distances from channel output and then choose output so that distance is minimal. This makes sense as calculating distance between LLRs and channel output is

Algorithm 3: Multi-Base Belief Propagation

Input: The received sequence $\boldsymbol{y}$

Initialization: $S := \emptyset$, initialize decoders $D_1, \dots D_s$
foreach $i = 1, \dots s$ **do**
    $\boldsymbol{x_i} = D_i(\boldsymbol{y})$
    if $\boldsymbol{x_i} H_i^T = \mathbf{0}$ **then**
        $S = S \cup \{i\}$
if $S = \emptyset$ **then**
    $S = \{1, \dots, s\}$

$$\hat{\boldsymbol{x}} = \underset{i \in S}{argmin} \sum_{j=1}^{n} \mid y_j - x_{ij} \mid^2$$

return $\hat{\boldsymbol{x}}$

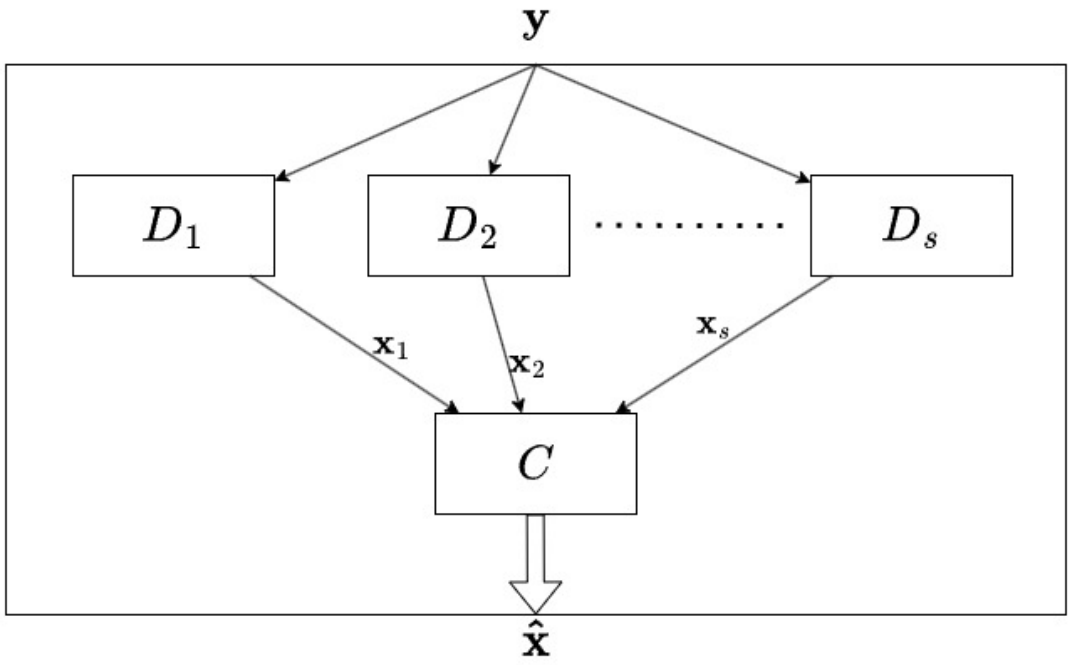

Figure 6. Multi-base decoder

unpredictable. Decoders may output LLRs with large absolute values and comparing the distance from those loses meaning. We propose this new low-complexity decoding scheme as Algorithm 4.

Algorithm 4: Multi-Base Generalized Belief Propagation

Input: The received sequence $\boldsymbol{y}$

Initialization: $S := \emptyset$, initialize **generalized BP** decoders $D_1, \dots D_s$
foreach $i = 1, \dots s$ **do**
    $\boldsymbol{x_i} = D_i(\boldsymbol{y})$
    if $\boldsymbol{x_i} H_i^T = \mathbf{0}$ **then**
        return $\boldsymbol{x_i}$
if $S = \emptyset$ **then**
    $S = \{1, \dots, s\}$
    / Convert vectors of LLRs to binary vectors. */*
    foreach $i = 1, \dots s$ **do**
        / Interpret logical values binary symbols. */*
        $\boldsymbol{x_i} = \boldsymbol{x_i} < 0$

$$\hat{\boldsymbol{x}} = \underset{i \in S}{argmin} \sum_{j=1}^{n} \mid y_j - x_{ij} \mid^2$$

return $\hat{\boldsymbol{x}}$

The next chapter focuses on the search for optimal representations (parity-check matrices) of the best known QC codes such that the generalized BP algorithm has good error-correcting performance. These representations are then used in decoders $D_1, \dots D_s$

4.5 Search for Representations of Best Known QC Codes

The following procedure was used to find optimal representations of best known QC codes.

1. Generate parity check matrix H for best minimum distance QC code from [BJKS02].
2. Perform exhaustive search over subspace defined by H to find parity-check matrices H' that
   (a) decode the same code as H;
   (b) consist of cyclic shifts of the first row;
   (c) have minimal row weight.
3. Simulate generalized BP decoding for some strip length m using H' over the AWGN channel for different signal to noise ratios.
4. Normalize the decoding results and rank the representations.

We will now give pseudo-code for searching for best representations of given QC code.

Algorithm 5: Procedure for finding good parity-check matrices for given QC code C

Input: Generator matrix G for $[n, \frac{n}{2}, d]$ QC code C with best minimum distance. Taken from [BJKS02]. Strip length m

Result: Ranked list of minimum weight rows which generate parity-check matrix H for code C

Initialization: Find a systematic form parity-check matrix H based on G.
Since $R = \frac{1}{2}$, H has dimensions $\frac{n}{2} \times n$.
Denote by d' the minimum distance of code defined by considering H as a generator matrix.
Initialize empty list S used for storing rows to generate parity-check matrices.
Initialize matrix RES for storing results of decoding simulations. Columns of RES correspond to SNRs, odd rows to FERs of decoders, even rows to BERs of decoders.
foreach i *from* 1 *to* $2^k - 1$ **do**
    Calculate binary vector $\boldsymbol{i}$ corresponding to decimal i. Consider H as generator matrix for dual code. Compute $\boldsymbol{c} = \boldsymbol{i}H$
    if $w(\boldsymbol{c}) == d'$ **then**
        Consider the matrix $H' = (\boldsymbol{c}, \boldsymbol{c}^{(2)}, \dots \boldsymbol{c}^{n})^T$.
        if $rank(H') == \frac{n}{2}$ *and no cyclic shifts of* $\boldsymbol{c}$ *are in* S **then**
            $S = S \cup \{c\}$
foreach $\boldsymbol{c} \in S$ **do**
    foreach SNR *from 2 to 5* **do**
        Simulate generalized belief propagation decoding with strip length m using parity-check matrix $H' = (\boldsymbol{c}, \boldsymbol{c}^{(2)}, \dots \boldsymbol{c}^{n})^T$, where $\boldsymbol{c^{(n)}}$ denotes a cyclic shift of $\boldsymbol{c}$ by n positions, over $AWGN$ channel. Obtain corresponding FER and BER.
    Append rows which correspond to decoding results (FERs and BERs) of H' generated by $\boldsymbol{c}$ to RES
Split the odd and even rows of RES to RES_{FER} and RES_{BER}.
Normalize RES_{BER} and RES_{FER} by columns, because larger SNRs have smaller error rates.
Sort RES_{BER} and RES_{FER} by ascending row weight.
Sort S in the same order, corresponding to either RES_{BER} or RES_{FER} as necessary.
S now consists of good first rows for generating parity-check matrices, ranked by their error-correcting capabilities.

5 Results

Exhaustive search for best code representations was done for the best minimum distance QC $[24, 12]$ code from [BJKS02]. After choosing the best representations, decoding was simulated using Algorithm 4 with $s = 5$ decoders using strip lengths $m = 2, 4, 6$. Each generalized BP decoder performed a maximum of 50 iterations. Figure 7 and Figure 8 were obtained by simulating this decoding scheme for SNRs in the range 2-5. For simulations and plots, MatLab was used. They also contain ML decoding FER and BER and results from [HHLM07]. The best results were obtained with highest strip length $m = 6$. Decoder with strip length 6 had approximately the same performance as [HHLM07]. For some SNRs, slightly better error rates were achieved. This leads us to believe that this decoding scheme is promising, because in [HHLM07] three times more $s = 15$ decoders were used to decode the $[24, 12]$ Golay code with the same minimum distance. For decoders with strip length 6, FER and BER were respectively about 0.5dB and 0.2 worse than optimal decoding. It is possible that performance could be further improved by adding decoders or increasing strip length. The proposed decoding scheme could have lower average complexity, due to the fact that decoding is stopped once zero syndrome is obtained, whereas in their paper, every sub-decoder was run on every channel output. On the other hand, each generalized BP sub-decoder has a larger time complexity than regular BP decoders, therefore it is not apparent which algorithm is simpler.

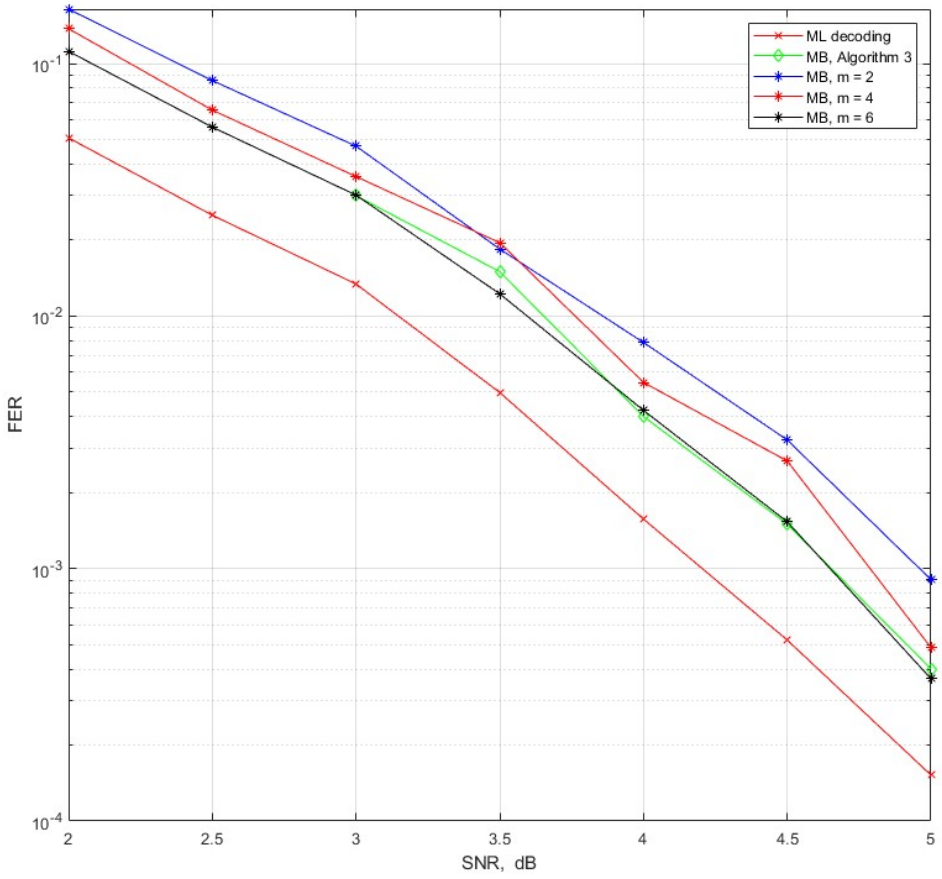

Figure 7. FER of multi-base decoding of QC $[24, 12]$ code

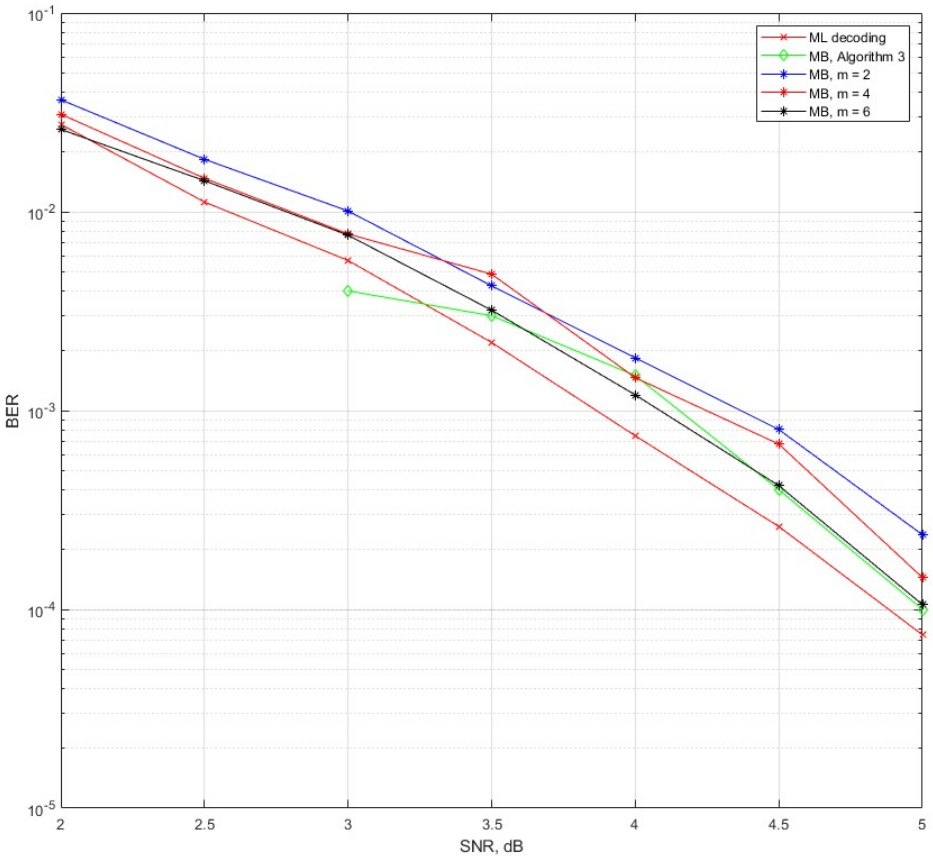

Figure 8. BER of multi-base decoding of QC $[24, 12]$ code

6 Conclusion

In this thesis, we tried to achieve low error rate, low complexity decoding for QC codes with the best minimum distance for given code length. We attempted to improve decoding with the following three techniques

1. Since the best QC codes are too dense for BP decoding, we tried to approach ML decoding by considering them as GLDPC codes and using the generalized BP algorithm.

2. Multi-base decoding - many decoders were used for the same code. Since each of them has relatively low complexity, adding them influences the complexity by some constant.

3. The performance of the BP algorithm depends on the structure of the parity-check matrix. We performed exhaustive search to find the best representations for codes such that error rates are minimal.

Using these techniques, near-ML performance was achieved with lower complexity.

6.1 Future work

Multi-base decoding of GLDPC codes is promising due to the degrees of freedom the schemes have. That is, there are many parameters to optimize, such as finding better representations, changing the number of decoders or changing the strip length for generalized BP decoding. The following topics should be considered for further research:

- Time complexity analysis for Algorithm 4.

- The techniques explored in this thesis could be tried for longer codes. It would be remarkable if results were similar, as optimal decoding is infeasible for longer codes.

- Different ways of combining sub-decoder outputs for MB decoding should be investigated. Methods from the fields of machine learning and neural networks could be used to combine different decoder outputs so as to reduce error rates.

References

[BCJR74] Lalit R. Bahl, John Cocke, Frederick Jelinek, and Josef Raviv. Optimal decoding of linear codes for minimizing symbol error rate (corresp.). *IEEE Trans. Inf. Theory*, 20(2):284–287, 1974.

[BJKS02] Irina E. Bocharova, Rolf Johannesson, Boris D. Kudryashov, and Per Ståhl. Tailbiting codes: Bounds and search results. *IEEE Trans. Inf. Theory*, 48(1):137–148, 2002.

[Gal62] Robert G. Gallager. Low-density parity-check codes. *IRE Trans. Inf. Theory*, 8(1):21–28, 1962.

[HHLM07] Thorsten Hehn, Johannes B. Huber, Stefan Laendner, and Olgica Milenkovic. Multiple-bases belief-propagation for decoding of short block codes. In *IEEE International Symposium on Information Theory, ISIT 2007, Nice, France, June 24-29, 2007*, pages 311–315. IEEE, 2007.

[Sha48] Claude E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(4):623–656, 1948.

[Tan81] Robert Michael Tanner. A recursive approach to low complexity codes. *IEEE Trans. Inf. Theory*, 27(5):533–547, 1981.

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Hans Kristjan Veri**

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Low-Complexity Iterative Decoding of Best Known Quasi-Cyclic Codes,

   supervised by Irina Bocharova, PhD, Boris Kudryashov, PhD, Vitaly Skachek, PhD.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Hans Kristjan Veri
09/05/2023