INSTITUT FÜR
NACHRICHTENTECHNIK

Prof. Dr.-Ing. Gerhard Bauch

Master Thesis

March 2017

Information Optimum Design of Discrete LDPC Decoders for Irregular Codes

Author: Maximilian Stark
Supervisor: Prof. Dr.-Ing. Gerhard Bauch

Contents

Acronyms

AWGN additive white Gaussian noise

BI-AWGN binary-input AWGN

BEC binary erasure channel

LDPC low-density parity-check

LLR log-likelihood ratio

SPA sum-product algorithm

IB node Information Bottleneck node

FEC forward error correction

BER bit error rate

QAM quadrature amplitude modulation

PSK phase-shift keying

BPSK binary phase-shift keying

ASK amplitude-shift keying

WLAN wireless LAN

DVB-S2 digital video broadcasting – satellite 2

Chapter 1

Introduction

In the last decade, a small number of young entrepreneurs in the Silicon Valley started no more, no less than a digital revolution. By developing smartphone apps they facilitated people all over the world to connect easily and share nearly everything. To a certain extent, they rediscovered the Latin origin of the word "communicare" meaning "to share".

One major technical prerequisite allowing these new opportunities to spread around the world was the wide coverage of mobile Internet access. Providing wide coverage paired with high data rates was driving the communication technology sector in the past decades and enormous progress in the field of mobile broadband technologies was made. With the standardization and launch of Long Term Evolution (LTE) high-speed mobile data traffic became available for the mass market and laid the groundwork for many new, revolutionizing applications. However, since people have encountered these new possibilities, the demand for higher data rates at minimum latency grew further.

Due to several degrading physical effects, the mobile radio channel is one of the most challenging propagation environments. Anyway, engineers were able to develop and apply clever tricks to overcome the impairments of this transmission medium. In this context, channel coding plays and always played an important role because it helps to provide a guaranteed quality of service. Nevertheless, if existing, where is the theoretical upper limit of data rates for a certain channel, when a channel code is applied?

The seminal work "A Mathematical Theory of Communication" [Sha01] by Claude E. Shannon laid the foundation of modern coding theory. It answered the question on the maximum possible data rate from a theoretical perspective. Furthermore, it introduced the concept of mutual information which will be used throughout this work. Finally, Shannon determined the maximum achievable rate of a channel code which allows error-free transmis-

sion over a disturbed channel. Unfortunately, Shannon's seminal work did not cover construction tools for this code.

Anyway, this publication marked the beginning of a race in which researchers tried to find capacity achieving codes. This race should last for over 50 years until in 2001 the theoretical boundary was reached up to 0.0045 dB over E_b/N_0 [CFRU01]. The utilized code was a so-called *irregular* low-density parity-check (LDPC) code with a very large block length. Until this breakthrough, turbo codes were the best-known codes that came closest to the theoretical channel capacity [BGT93].

Although capacity approaching channel codes have been found in the past, their decoding requires complex algorithms that prohibit their practical implementation in consumer hardware. To achieve future demands on latency and data rates efficient encoding and decoding of error correcting codes are of crucial importance. Furthermore, reducing the power consumption of encoder and decoder is a current subject in communications engineering. This issue will gain integral importance in the future, especially, for mobile users and tiny sensor nodes with strict energy constraints.

An all new approach to design error correction units with low complexity are so-called *discrete* decoders. This idea was successfully applied to LDPC codes in [LB15] and [LSB16a]. For decades, it was a scientific consensus in the communication and signal processing community, that the most promising approach guaranteeing the best performance, would be to represent all received samples in the real or complex domain as precisely as possible. However, the real or complex domain are continuous, which can be impossibly represented in hardware accurately. Therefore, high precision floating-point data types or fixed point arithmetic with long bit words are often used in modern hardware in order to achieve close to optimal performance. To realize such high precision floating point operations, computationally expensive hardware, much processing time and a significant amount of memory are required.

In contrast to state-of-the-art implementations, discrete decoders avoid high precision processing but try to process and extract only *relevant information* needed for the decoding. This information theoretical approach exploits the Information Bottleneck method [TPB00]. The Information Bottleneck method is able to maintain relevant information about a variable of interest and simultaneously discretizes the event space of an observation coarsely.

A drawback of the so far existent discrete LDPC decoders is there limited applicability solely to *regular* LDPC codes. However, the full error correction potential of LDPC codes can only be exploited by irregular codes, which have different node degrees involved in the iterative decoding process. In

fact, most standardized LDPC codes are irregular codes, e.g in [IEE12] or [ETS14]. Hence, for practical applications, discrete decoders for irregular codes are of great interest. Different from state-of-the-art decoders, this generalization is not straightforward but requires a new design framework for discrete decoders. This work aims to extend the decoding principles from [KYK08] to irregular codes, based on the promising results from [LB15] for regular codes. In summary, the objectives of this work are to

1. broaden the applicability of discrete decoders to a wider class of codes
2. increase the practical relevance of the discrete LDPC decoders from [LB15]
3. describe and solve an information theoretical problem occurring in the construction of discrete LDPC decoders for irregular codes
4. evaluate the performance of the resulting decoders in numerical simulations for several irregular LDPC codes from communication standards
5. generalize the found solution to higher order modulation schemes, which are often paired with irregular LDPC codes in practice

To highlight the practical relevance, an irregular LDPC code from the wireless LAN (WLAN) standard IEEE 802.11n is taken for performance evaluation [IEE12]. Furthermore, another code from the digital video broadcasting – satellite 2 (DVB-S2) standard is investigated [ETS14]. The most important properties of the used codes are

- WLAN: code rate = 0.5, codeword length = 1296 bits
- DVB-S2: code rate = 0.5, codeword length = 64800 bits

In the next chapter, the Information Bottleneck method is introduced. Furthermore, a graphical framework called Information Bottleneck graph [LSB16a] is explained. Chapters 4-5 focus on LDPC codes and density evolution. Afterwards, Chapters 6-7 introduce discrete receiver design and reproduce results for regular LDPC codes from [LB15]. The following Chapter 8 presents and investigates *message alignment*, a newly developed technique needed to generalize discrete decoder design from regular to irregular LDPC codes. In Chapter 9 these findings are evaluated using simulations. Finally, before concluding the thesis, message alignment is applied to higher order modulation schemes in Chapter 10.

Chapter 2

Fundamentals of Information Theory and the Information Bottleneck Method

Asking people on the street how they would define information, probably, results in very versatile answers. Moreover, when trying to even measure information, it turns out that one deals with a quite fuzzy quantity. Claude Elwood Shannon is known as the father of information theory. He figured out how to describe information in a probabilistic manner. Shannon defined the self-information of an event $x \in \mathcal{X}$ with probability $\Pr(X = x)$ as

$$I(x) = \log_2 \frac{1}{\Pr(X = x)}. \tag{2.1}$$

If, as done in (2.1), the $\log_2$ is chosen, then the self-information is measured in the pseudo unit *bits*. A simple example can help to check the credibility. Imagine the weather forecast app on your smartphone tells you that it will not rain tomorrow with a probability of 99%. Given this probability, one can compute the self-information

$$I(X = \text{no rain}) = \log_2 \frac{1}{\Pr(X = \text{no rain})} = \log_2 \frac{1}{0.99} = 0.0145 \text{ bit} \tag{2.2}$$

$$I(X = \text{rain}) = \log_2 \frac{1}{\Pr(X = \text{rain})} = \log_2 \frac{1}{0.01} = 6.6438 \text{ bit}. \tag{2.3}$$

Waking up the next day and observing that it, against all odds, rains will be probably really surprising. One would argue that this unexpected observation is really informative, i.e. contains a lot of new information. This

is not the case, if does not rain, as expected. This little example illustrates that (2.1) is a proper proposal to quantify information.

The example also shows that the self-information does not depend on the event space $\mathcal{X}$ of a random variable X. It does not matter if X refers to weather conditions or any other probabilistic event. Instead, the self-information is only a function of X's statistics. This makes information theory a very general framework with applications in many different fields, like physics, medicine, biology, computer science, economics etc. [CT12].

The concept of self-information can be generalized to a measure of uncertainty, also called entropy. The entropy $H(X)$ of a discrete random variable X is defined as

$$H(X) = \sum_{x \in \mathcal{X}} p(x) \log_2 \frac{1}{p(x)}, \tag{2.4}$$

i.e. as expectation of the self-information for all events x in the event space $\mathcal{X}$.

Self-information and entropy are essential fundamentals of information theory. These quantities can be used to derive a variety of very advanced concepts as, for example, done in [CT12]. However, this chapter will only cover a small selection of mathematical concepts integral to understand the Information Bottleneck method, like, for example, mutual information, information divergences and rate distortion theory. An overview on different Information Bottleneck algorithms concludes the chapter.

In between, another helpful framework called information geometry is introduced.

2.1 Mutual Information

Two coupled random variables X and Y carry information about each other, the so-called mutual information $I(X;Y)$ [CT12]. At the same time, this quantity denotes the reduction in the uncertainty of one random variable due to the knowledge of the other. The resulting definition is as follows

$$
\begin{aligned}
I(X;Y) &= H(X) - H(X|Y) && (2.5)\\
&= -\sum_{x\in\mathcal{X}} p(x)\log_2 p(x) - \left(-\sum_{y\in\mathcal{Y}} p(y)\sum_{x\in\mathcal{X}} p(x|y)\log_2 p(x|y)\right) && (2.6)\\
&= -\sum_{x\in\mathcal{X}} p(x)\log_2 p(x) + \sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p(x|y)p(y)\log_2 p(x|y) && (2.7)\\
&= \sum_{x\in\mathcal{X}} p(x)\sum_{y\in\mathcal{Y}} p(y|x)\log_2\frac{p(x|y)}{p(x)} && (2.8)\\
&= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log_2\frac{p(x,y)}{p(x)p(y)}. && (2.9)
\end{aligned}
$$

Rewriting (2.9) as

$$I(X;Y) = H(Y) - H(Y|X), \tag{2.10}$$

reveals the mutual information as symmetric. In other words, what X says about Y is as informative as what Y says about X.

Figure 2.1 visualizes the relationship between $H(X)$, $H(Y)$, $H(X|Y)$, $H(Y|X)$ and $I(X;Y)$ using a Venn diagram. Here, the mutual information is the intersection of the blue and orange set. In most interesting cases, the intersection should be large, corresponding to a significant mutual information. For example for detection problems based on the output of a communication channel, the output random variable Y should be highly informative about the channel input X.

Investigating the extrema allows deepening the understanding of mutual information. In case of two statistically independent random variables X and Y, i.e. $p(x,y) = p(x)\cdot p(y)$, (2.9) simplifies to

$$
\begin{aligned}
I(X;Y) &= \sum_{x,y} p(x,y)\log_2\frac{p(x,y)}{p(x)p(y)} && (2.11)\\
&= \sum_{x,y} p(x)p(y)\underbrace{\log_2\frac{p(x)p(y)}{p(x)p(y)}}_{0} = 0. && (2.12)
\end{aligned}
$$

This result is reasonable, because if X and Y are independent, knowing one variable gives you no information about the other. The second interesting case is full correlation between the random variables X and Y, i.e. $p(x,y) = p(x,x) = p(x)$.

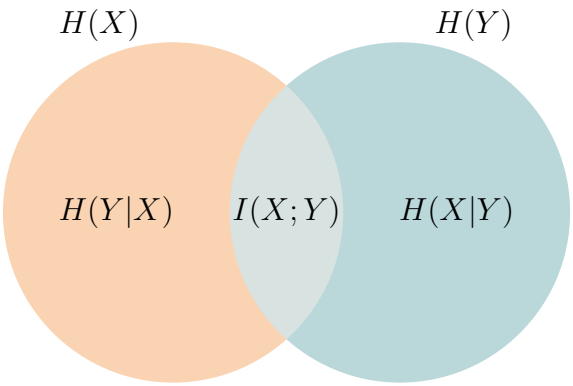


Figure 2.1: Venn diagram showing the connection between entropy, conditional entropy and mutual information

In this case the mutual information becomes

$$I(X;Y) = \sum_{x \in \mathcal{X}} \underbrace{p(x,x)}_{p(x)} \log_2 \frac{1}{p(x)} \tag{2.13}$$

$$= H(X). \tag{2.14}$$

2.2 Information Geometry

Information theory deals solely with probabilities and other fuzzy quantities. Thus, the problems are often hard to imagine and barely tangible. In mathematics, geometry can be helpful to understand the shape of functions and objects. Geometry also visualizes distances and helps to understand corresponding cost functions. Exploring information theory using geometric methods is the basic idea of information geometry [Ama16].

At first, a mathematical concept to describe a general kind of space, called manifold, is needed. An n-dimensional manifold M can be understood as a deformed n-dimensional Euclidean space, with some different properties, as described in [Ama16]. Admittedly, manifolds are a very abstract extension. However, understanding the properties of manifolds is not essential for this thesis.

Usually, coordinate systems are introduced to explore manifolds. Different coordinate systems can be used for the same manifold. Sometimes it is even required to use more than one coordinate system to reach all points in a manifold [Ama16]. Among others, the 2-dimensional Euclidean space can be described using an orthonormal coordinate system or a polar coordinate system.

Information theory deals with probability distributions, thus one has to investigate the manifold of probability distributions. The focus of this thesis lies on the class of discrete probability distributions.

Consider a discrete random variable X with event space $\mathcal{X} = \{0, 1, \cdots, n\}$ and probability mass function $p(x)$. For discrete random variables the probability mass function function is often represented as a point in the $n+1$-dimensional space using the vector notation

$$\mathbf{p} = [\Pr(X = 0), \Pr(X = 1), \cdots, \Pr(X = n)]^{\mathrm{T}}$$

or in matrix notation, if more than one random variable is involved. Validity of a probability mass function is only given, if

$$\sum_{x \in \mathcal{X}} p(x) = 1, \ \Pr(X = x) > 0 \, \forall x \in \mathcal{X}. \tag{2.15}$$

The set of all possible probability mass functions $\mathbf{p}$ forms an n-dimensional manifold. A possible coordinate system is

$$\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_n) = (\Pr(X = 1), \ldots, \Pr(X = n)),$$

where the parameter $\Pr(X = 0)$ is constrained according to

$$\Pr(X = 0) = 1 - \sum_{i=1}^{n} \xi_i.$$

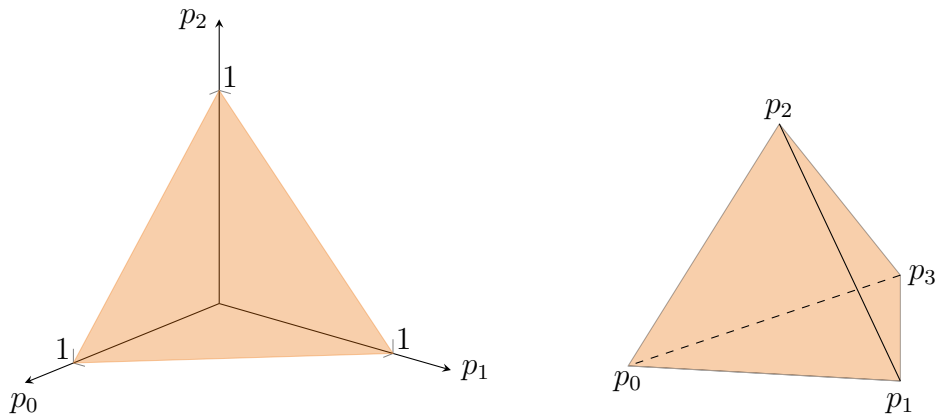
The resulting manifold is a simplex [Ama16]. Figure 2.2 shows two such probability simplices, denoted S_n. If $n = 2$, the corresponding manifold S_2 is the interior of a triangle, whereas S_3, i.e. $n = 3$, denotes the interior of the 3-simplex.

It is also possible to introduce another coordinate system $\boldsymbol{\theta}$, which will be very useful in the next section, namely

$$\theta_i = \log \frac{p_i}{p_0}, \ i = 1, \cdots, n.$$

2.3 Kullback-Leibler Divergence

When applying information geometry, probability distributions are interpreted as points inside a high-dimensional manifold. The distance between these points is determined using a so-called divergence $D\{p(x)||q(x)\}$ [CT12], [Ama16]. In contrast to a metric, which is used to measure distances in the



(a) (b)

Figure 2.2: In (a) the S_2 simplex is visualize. The right subfigure (b) shows a S_3 simplex.

Euclidean space, divergences have different, maybe on the first glance, curious properties. Most importantly, a divergence is usually not symmetric, raising questions on how to order the arguments. However, interpreting a divergence as distance measure between two probability distributions is valid and appropriate [CT12].

A common divergence known as Kullback-Leibler divergence D_{KL} or relative entropy is defined as [Ama16], [CT12]

$$D_{KL}\{p(x)||q(x)\} = \sum_x p(x) \log \frac{p(x)}{q(x)}. \tag{2.16}$$

In the context of source coding, the relative entropy determines the penalty, i.e. the number of additional bits needed to encode X if the distribution $q(x)$ is considered as the source distribution instead of the true distribution $p(x)$. The Kullback-Leibler divergence is always positive and only for $p(x) = q(x)$, $D_{KL}\{p(x)||q(x)\} = 0$.

2.4 Jensen-Shannon Divergence

In some applications, e.g. the Information Bottleneck method, a symmetric generalization of the Kullback-Leibler divergence $D_{KL}\{p(x)||q(x)\}$ is required, called the Jensen-Shannon divergence $JS\{p(x)||q(x)\}$. The Jensen-Shannon divergence with weights π_1 and π_2 is defined as [Slo02]

$$JS_\Pi\{p(x)||q(x)\} = \pi_1 D_{KL}\{p(x)||\bar{p}(x)\} + \pi_2 D_{KL}\{q(x)||\bar{p}(x)\}, \tag{2.17}$$

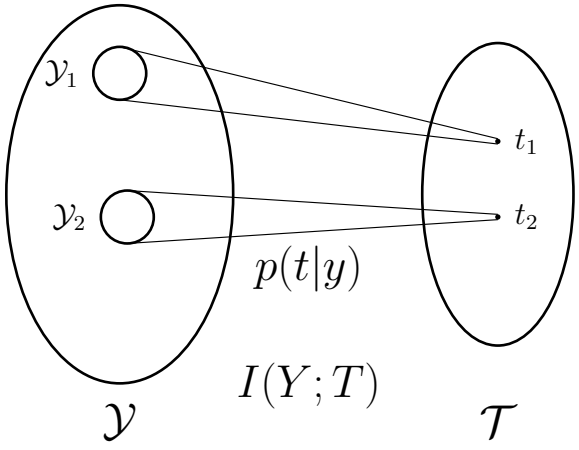


Figure 2.3: The general procedure of compression is shown. A subset of input sequences $\mathcal{Y}_1$, taken from a large set $\mathcal{Y}$ is mapped onto one single point t_1 in the reduced set $\mathcal{T}$. The mapping is denoted by $p(t|y)$ [Slo02].

where $\Pi = \{\pi_1, \pi_2\}$, $0 < \pi_1, \pi_2 < 1$, $\pi_1 + \pi_2 = 1$ and $\bar{p}(x) = \pi_1 p(x) + \pi_2 q(x)$.

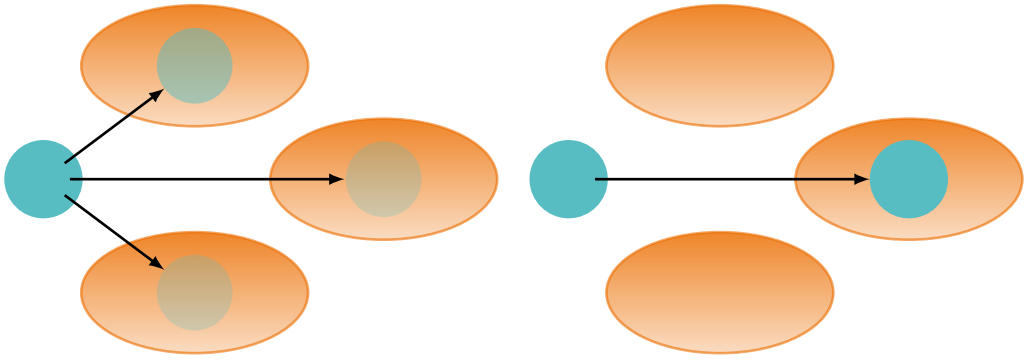
2.5 Rate Distortion Theory

The field of rate distortion theory deals with finding an optimum trade-off between distortion and compression, when quantization is applied.

Quantization is a well-known concept in signal processing if a continuous signal has to be represented by a limited number of N states. In most technical applications this number is a power of 2, i.e. $N = 2^n$ because then n is the number of bits required to store a sample. Sometimes, mainly in the field of source coding, this problem is also referred to as compression. Rate distortion theory allows determining optimal representatives, given a distortion measure d.

Compression or quantization is not solely referring to representing a continuous variable as discrete, it might be also required if a discrete variable has a large event space. Consider a discrete random variable Y with probability distribution $p(y)$ and finite event space $\mathcal{Y}$, with cardinality $|\mathcal{Y}|$. If $|\mathcal{Y}|$ is too large to be handled, e.g. by the system architecture, a compression variable T is introduced with a much smaller cardinality $|\mathcal{T}|$. The ratio between $|\mathcal{T}|$ and $|\mathcal{Y}|$ defines the compression rate, an important characteristic.

A compression can be visualized as a mapping from one large set to a smaller compression set, as shown in Figure 2.3. This mapping $p(t|y)$ can either be stochastic or deterministic as shown in Figure 2.4. Figure 2.4



(a) (b)

Figure 2.4: Compression can also be interpreted as clustering. Each orange ellipse is one different cluster. The mapping of $y \in \mathcal{Y}$ onto a cluster $t \in \mathcal{T}$ is described by the transition probability $p(t|y)$. The mapping can either be stochastic as shown in (a) or deterministic shown in (b).

visualizes also another notion of compression, namely clustering, because compression means to map different values $y \in \mathcal{Y}$ onto a cluster $t \in \mathcal{T}$. This can be one particular cluster t (deterministic mapping) or different clusters with a certain probability (stochastic mapping).

The term clustering will be used throughout this thesis because it emphasizes that, in contrast to typical quantization problems, determining a representative is not necessary. In later chapters, it will be introduced how to process only cluster indexes without the necessity of representatives.

However, no matter how the mapping from Y onto T is called, the question is how to accomplish it optimally. From an information theoretical perspective, it is straightforward to consider the mutual information $I(Y;T)$, usually termed *compression information*. The more compact Y is represented, the lower the compression information [Slo02]. In the extreme case of $|\mathcal{T}| = 1$, i.e. only one cluster exists, $I(Y;T) = 0$. Alternatively, when $|\mathcal{T}| = |\mathcal{Y}|$ no compression is achieved and there will be a one-to-one mapping resulting in $I(Y;T) = H(Y)$.

The above consideration highlights, considering the compression information alone is not sufficient to determine the costs of compression. Hence, in rate distortion theory the distortion measure $d(x,t)$ is introduced. This measure allows to determine the expected costs, respectively expected distortion [Slo02]

$$\mathbb{E}_{p(y)p(t|y)}\{d(x,t)\} = \sum_{x\in\mathcal{X},t\in\mathcal{T}} p(y)p(t|y)d(x,t). \tag{2.18}$$

In the next step, it is possible to set up the rate-distortion function $R(D)$ describing the trade-off between compression and expected distortion

$$R(D) = \min_{\{p(t|y):\mathbb{E}_{p(y)p(t|y)}\{d(x,t)\}\leq D\}} I(Y;T) \tag{2.19}$$

Consequently, $R(D)$ is the minimal compression information considering all $p(t|y)$ which satisfy the constraint. An implicit solution can be found [Slo02], [CT12]

$$p(t|y) = \frac{p(t)}{Z(y,\beta)} e^{-\beta d(x,t)}, \tag{2.20}$$

where $Z(y,\beta)$ is a normalization function. Furthermore, β is a Lagrange multiplier which can be determined by

$$\frac{\delta R}{\delta D} = -\beta. \tag{2.21}$$

It is interesting to investigate β in more detail. If $\beta \to \infty$ the focus is solely on minimizing the distortion, whereas if $\beta \to 0$ one is interested in compression only meaning that also the rate $R \to 0$.

The optimal solution described in (2.20) can always be found using the Blahut-Arimoto algorithm [CT12]. This iterative optimization algorithm makes use of the induced structure of the problem [Bla72]. As long as the optimization problem influenced by the chosen distortion function is convex, a global minimum can be found [Slo02].

Rate distortion theory is a broad field, but the most relevant concepts for this thesis are summarized in this section. It will be important to remember that mapping continuous variables to discrete ones always causes distortion, that a solution for a given convex distortion measure can be found using the Blahut-Arimoto algorithm and that the notation clustering, mapping and compression are closely related and will be often used synonymously.

2.6 Information Bottleneck Method

In the previous section rate distortion theory was introduced. In contrast to rate distortion theory, where an arbitrary and explicit distortion measure, e.g. the quadratic distance, has to be defined, the Information Bottleneck focuses on preserving the mutual information between the compression variable T and a so-called relevant variable X.

The Information Bottleneck method is a closed form information theoretical framework originating from the field of machine learning [TPB00]. It

is framed by the idea that the self-information of a random variable can be classified into relevant and irrelevant information with respect to an appropriately defined relevant random variable. The main goal of the method is to keep as much relevant information contained in an observation of a random variable Y as possible, when a lossy compression of this variable is applied.

As mentioned above the Information Bottleneck is an extension of rate distortion theory where finding the right distortion measure is not needed anymore. Instead, a compression only with respect to the loss of relevant information is performed. The most important and crucial parameters that have to be defined in the Information Bottleneck framework are the relevant variable X and the joint probability $p(x, y)$ between this relevant variable X and the observed variable Y. In some applications both tasks, i.e. finding X and determining $p(x, y)$ can be tough.

Following the argumentation from above, the mutual information between observed and relevant variable $I(X;Y)$ contains not only relevant but also irrelevant information. The overall Information Bottleneck setup is visualized in Figure 2.5. The key idea of the Information Bottleneck is to extract the relevant part by squeezing this information through a compact bottleneck.

Therefore, a third variable, the compression variable T is introduced. In the previous section it was explained, that compression can be seen as a classification problem, where a classifier has to be found mapping $y \in \mathcal{Y}$ to $t \in \mathcal{T}$ in an optimal way. Consequently, the classifier $p(t|y)$ results from minimizing the compression information $I(Y;T)$ because minimizing this information results in a compression of Y. The extreme case of compression namely $I(T;Y) = 0$ and $I(T;Y) = H(Y)$ were investigated in detail while discussing rate distortion theory.

Especially, $I(Y;T) = 0$, i.e. the representation of Y in T is as compact as possible, reveals that looking only for the minimum compression information does not guarantee that our resulting clustering contains any information about the relevant variable. This violates the central requirement of the Information Bottleneck approach. Thus, the second constraint is to maximize the kept relevant information between T and X. Virtually, the Information Bottleneck tries to minimize the compression information $I(Y;T)$ by mapping Y to a compact variable T which simultaneously maximizes the information $I(T;X)$ between T and the relevant variable X.

The resulting optimization problem is also called the Information Bottleneck variational principle and can be expressed mathematically by the minimization of the so-called Information Bottleneck functional $\mathcal{L}$ [Slo02]

$$\mathcal{L}[p(t|y)] = I(T;Y) - \beta I(T;X). \tag{2.22}$$

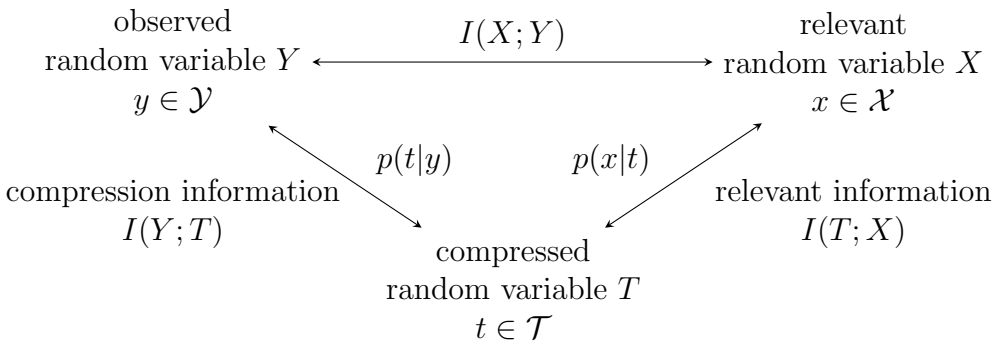


Figure 2.5: Information Bottleneck setup [LB15].

The applied optimization technique is Lagrange optimization [TPB00]. Hence, β denotes a Lagrange multiplier which is connected to the information constraint on $I(T;X)$. The extreme case $\beta = 0$ indicates maximum compression and $\beta \to \infty$ means maximum preservation of the relevant information. Thus, the parameter β is called trade-off parameter in literature and plays a similar role as the Lagrange multiplier in the solution of the rate distortion setup [Slo02], [TPB00].

In later chapters, when the construction of discrete LDPC decoders is explained, choosing β in order to get a high compression is not desirable. Instead, the aim is to keep as much relevant information as possible, i.e. a large β is used. The desired level of compression can also be achieved by restricting the event space cardinality of the compression variable T, i.e. $|\mathcal{T}|$.

After having explained the Information Bottleneck setup, the next step is to find a solution for this optimization problem. In the previous section, it was shown that deriving a solution for the quite similar rate distortion problem was possible. Unfortunately, applying the result obtained in rate distortion theory is not feasible, because the dependency of $I(T;X)$ in $p(t|y)$ is non-linear, whereas the constraint in the rate distortion problem was linear [Slo02]. Furthermore, there is no distortion measure defined in advance, which was present in the optimal solution (2.20).

Tishby et al. [TPB00] were able to evade these problems and derived an implicit solution, namely

$$p(t|y) = \frac{p(t)}{Z(y,\beta)} e^{-\beta D_{KL}\{p(x|y)||p(x|t)\}}, \forall t \in \mathcal{T}, \forall x \in \mathcal{X}, \tag{2.23}$$

where $Z(y,\beta)$ is again a normalization function, as in (2.20).

It should be highlighted at this point that the Kullback-Leibler divergence $D_{KL}\{p(x|y)||p(x|t)\}$ in (2.23) emerges during the derivation and was not

assumed as distortion measure in advance. This is quite interesting since this very common information theoretical concept underlines its significance, by raising autonomously as the correct measure for this problem.

Although the Information Bottleneck is a very powerful framework and has an implicit solution, the applications in the field of communication technologies and signal processing are quite rare [LB15], [LSB16a], [LSB16b]. So far, the utility of the Information Bottleneck method was primarily illustrated in a variety of fields, e.g. gene analysis [JL10], bio-informatics [BSCG13], machine learning [TZ15], predictive inference [Sti14], word clustering [ST00].

2.7 Information Bottleneck Algorithms

In the previous section, the basic concepts of the Information Bottleneck method were introduced. Furthermore, an implicit solution of the Information Bottleneck functional (2.22) was derived resulting in (2.23). In this section, different algorithms able to construct optimal or approximated solutions to the Information Bottleneck problem are introduced.

One major challenge of the optimization problem is the convex/concave structure. Thus, it cannot be guaranteed that a global optimum is found. Hence, the presented algorithms need to be performed multiple times with different, randomly chosen, initial conditions. In the end, the best local minimum of $\mathcal{L}[p(t|y)]$ found is chosen. "Best" in this context refers to the amount of kept relevant information $I(T;X)$, which should be maximized according to the Information Bottleneck setup. Although maybe not a global optimum is found, this approach yields good results very often.

Before explaining the algorithms and their implementation in detail, the concept of deterministic mapping and its counterpart the stochastic mapping shall be recapped. As shown in Figure 2.4, if $p(t|y)$ is a deterministic mapping it means that each realization $y \in \mathcal{Y}$ is mapped into exactly one cluster that is represented by the assigned element $t \in \mathcal{T}$. This can also be called "hard" clustering. Whereas in case of a stochastic mapping, the clustering is "soft", i.e. $p(t|y)$ can take any value between 0 and 1 for several t as long as $\sum_{t\in\mathcal{T}} p(t|y) = 1$.

The role of the Lagrange multiplier β in the Information Bottleneck solution (2.23) was already discussed in the previous section. When being interested in only maintaining relevant information, i.e. $\beta \to \infty$, hard clustering algorithms turn out as beneficial approach [Slo02]. Hence, all three presented algorithms are hard clustering algorithms. Further algorithms are explained in [Slo02].

2.7.1 Sequential Information Bottleneck Algorithm

The sequential Information Bottleneck algorithm aims to minimize the Information Bottleneck functional (2.23) using a deterministic clustering. Starting from an initial clustering, the current event y is drawn from its cluster. Afterwards, the costs of putting a certain event $y \in \mathcal{Y}$ in one of the $|\mathcal{T}|$ clusters $t \in \mathcal{T}$ are determined. These so-called merger costs $\Delta\mathcal{L}$ can be computed as [Slo02]

$$\Delta\mathcal{L}(t_i, t_j) = p(\bar{t}) \cdot \bar{d}(t_i, t_j), \tag{2.24}$$

where

$$\bar{d}(t_i, t_j) = JS_{\Pi}\{p(y|t_i)||p(y|t_j)\} - \beta^{-1} JS_{\Pi}\{p(x|t_i)||p(x|t_j)\}. \tag{2.25}$$

and

$$p(\bar{t}) = p(t_i) + p(t_j). \tag{2.26}$$

In this work, maximum preservation of relevant information is desired. Hence, the limit $\beta \to \infty$ is of interest and thus the second term can be neglected. Only one event y is handled at a time, all other elements stay in their clusters. After the costs are computed for each cluster candidate, y is assigned to its new cluster t_{new}, which causes the smallest merger costs

$$t_{\text{new}} = \arg\min_{t} \Delta\mathcal{L}(t, t_j). \tag{2.27}$$

This procedure is performed sequentially for every event and repeated until a stable solution is found, i.e. no elements $y \in \mathcal{Y}$ are shifted between different clusters anymore. Using several different random initial cluster mappings tries to counteract the problem of local optimality. For input distributions with a large event space $|\mathcal{Y}|$ the sequential structure of the algorithm causes long computation times.

2.7.2 Modified Sequential Information Bottleneck Algorithm

The computation time of the sequential Information Bottleneck algorithm can be significantly reduced if the event space of the observed random variable Y can be uniquely sorted by the log-likelihood ratio (LLR) $L(y|x)$. A modified sequential Information Bottleneck algorithm was introduced in [LB15]. The main adaptation is that due to the mentioned order of the event space of Y, only cluster borders between two neighbored clusters have to be optimized. The theoretical background was elaborately investigated in [KY14].

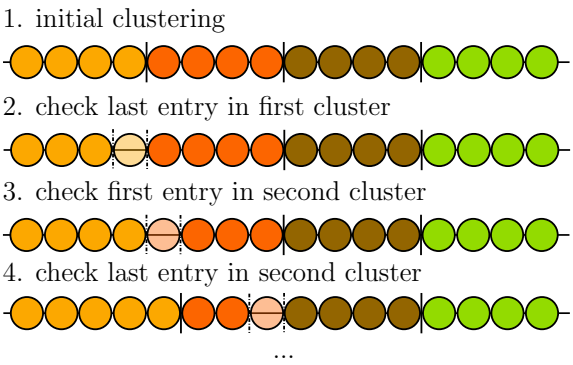


Figure 2.6: Visualization of the modified sequential Information Bottleneck algorithm.

Especially, for an interesting Information Bottleneck application, namely information optimum quantizer design, this algorithm is very useful.

At first, the initialization procedure is modified. Instead of initially choosing random clusters for each event, the cluster sizes are chosen randomly. Each cluster only consists of subsequent elements (cf. step 1. in Figure 2.6). The cumulative sum of the cluster sizes equals the sequence containing the boundaries of each of the $|\mathcal{T}|$ quantizer regions. The event space is now clustered according to these borders. Starting from this initial mapping the sequential Information Bottleneck algorithm from section 2.7.1 is used. However, instead of computing the merger costs for each event, only the first and the last element in a cluster are checked (cf. steps 2. and 3. in Figure 2.6). This implements the desired shifting of the quantizer region borders. Similarly to the sequential Information Bottleneck algorithm this procedure is repeated until the clustering does not change anymore (cf. step 4. in Figure 2.6). Obviously, the number of merger cost computations is drastically reduced from $|\mathcal{Y}|$ to $2|\mathcal{T}| - 1$ per iteration.

2.7.3 Symmetric Sequential Information Bottleneck Algorithm

If the ordered event space of Y is symmetric, in the sense that $L(x|Y = y) = -L(x|Y = |Y| - 1 - y)\ \forall y \in \mathcal{Y}$, the algorithm's complexity of the modified sequential Information Bottleneck algorithm can be halved. Optimization over the first $|\mathcal{T}|/2$ clusters only and using a flipped version of the obtained clustering for the second half of the event space results in a symmetric clustering. For a symmetric clustering, also $p(x|t)$ is symmetric in the sense that $p(x|T = t) = 1 - p(x|T = |T| - t - 1)\ \forall t \in \mathcal{T}$. Sometimes it might be possible

that the event space of Y contains duplicates, preventing a unique sort. In this case, an additional pre-clustering is applied to group such duplicates in order to be able to apply the modified variants of the sequential Information Bottleneck algorithm.

2.7.4 Prior Selection

The sequential Information Bottleneck algorithm starts with choosing initial cluster sizes. This equals the problem of prior selection in Bayesian statistics and is well known in machine learning [Bis07]. A really useful distribution is the Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\alpha})$. This family is commonly used as conjugated prior of multinomial variables and is basically a family of multivariate probability distributions parameterized by a vector $\boldsymbol{\alpha}$. Each realization drawn from a Dirichlet distribution is a valid probability mass function. However, the moments of this distribution, e.g. the mean, are only reached when considering infinitely many realizations. The idea developed during the implementation of the algorithms for this thesis is to use a Dirichlet distribution to create the initial cluster sizes for the modified and symmetric Information Bottleneck. Doing so one creates a new degree of freedom by putting a prior on the cluster size. In this thesis, the Dirichlet distribution was chosen such that each entry in $\boldsymbol{\alpha}$ is the same, i.e. $\alpha_i = \mathrm{const}, \forall i$. Hence, on average one samples a valid uniform distribution for $p(t)$, which is a so called non-informative prior.

Chapter 3

Information Bottleneck Graphs

In reality, when trying to infer a certain parameter given some observation one is facing two different kinds of problems, namely detection and estimation. If the relevant quantity X is continuous, i.e. $x \in \mathbb{R}$ the process of inferring x given the observation $y \in \mathcal{Y}$ is called estimation. In case of X being discrete with a finite event space cardinality $|\mathcal{X}|$ the inference task is called detection.

In the previous section, it was shown that the Information Bottleneck method is perfectly suited for classification problems. Moreover, it turns out that the Information Bottleneck method is able to simplify processing of inference problems [LSB16a].

Quantities, of interest for solving inference problems are, e.g,

maximum a-posteriori estimate $\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{y})$

marginal posterior $p(x_i|\mathbf{y})$

marginal distribution of Y $p(\mathbf{y})$.

Depending on the dimensionality of X obtaining the mentioned quantities can result in an exponentially increasing workload. Factor graphs are a graphical framework reducing the complexity by exploiting conditional independence in a systematic way [KFL01], [Loe04]. This framework is used in every field that involves estimation or detection and is used extensively in the development of decoding algorithms for error correcting codes [Ksc03].

The first section of this chapter will explain how to factorize a joint distribution and transform it into a factor graph.

The algorithm used to determine the three quantities mentioned above works on factor graphs and is called the sum-product algorithm or belief-propagation. Using this technique and combining it with the fundamental

purpose of the Information Bottleneck, i.e. extracting relevant information, results in a framework that allows to control and understand the flow of relevant information through a complex system. This framework, called Information Bottleneck graphs, was invented by Lewandowsky et al. [LSB16a] and can be understood as information preserving extension of factor graphs. Information Bottleneck graphs will be used throughout this thesis to explain discrete decoder design in a graphical manner. Hence, this framework will be introduced in detail in the second part of this chapter.

3.1 Factor Graphs

Let us assume a set of random variables $X_1, X_2, \ldots, X_{n-1}, X_n$ and the corresponding joint distribution $p(x_1, x_2, \ldots, x_{n-1}, x_n)$. According to the chain rule [CT12], every joint distribution can be factorized as

$$p(x_1, x_2, \ldots, x_{n-1}, x_n) = p(x_1|x_2, \ldots, x_{n-1}, x_n) \cdot p(x_2| \ldots, x_{n-1}, x_n) \ldots p(x_{n-1}|x_n)p(x_n). \tag{3.1}$$

Depending on the setting there might exist conditional independences, such that a factorization of four random variables could be

$$p(x_1, x_2, , x_3, x_4) = p_A(x_1) \cdot p_B(x_1, x_2) \cdot p_C(x_1, x_3, x_4). \tag{3.2}$$

Having obtained a simplified factorization of a joint distribution by exploiting setting specific conditional independences as in (3.2), one can construct a factor graph according to the following rules [Ksc03]:

1. Create a vertex for every variable X_i
2. Create a vertex for every factor p_k
3. Create an edge between vertex X_i and p_k if x_i is an argument of p_k

The corresponding factor graph to the factorization from (3.2) is shown in Figure 3.1.

3.1.1 Sum-Product Algorithm

The main motivation to introduce factor graphs was to use their capabilities, regarding the complexity reduction when computing marginal posteriors or maximum a-posteriori estimates. In the previous section the construction of

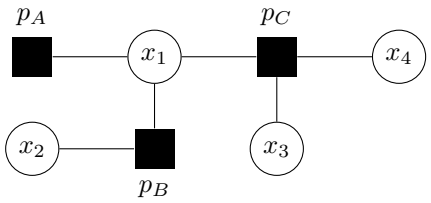


Figure 3.1: Factor graph corresponding to the factorization from (3.2).

factor graphs was explained. This section introduces the powerful class of messages passing algorithms.

Starting from a factorization of a probability distribution visualized in a graph as in Figure 3.1, the idea is to send messages which contain intermediate results created at one factor along an edge to the connected variable nodes. According to some update rules, the messages are propagated further to the connected factor nodes. This procedure is repeated until all messages are distributed through the whole graph. Depending on the desired quantity the algorithm is either called sum-product algorithm (SPA) to compute marginal posteriors or max-product algorithm to compute the maximum a-posteriori decision.

The outgoing message $\mu_{p_k \to X_j}(x_j)$ at a factor node p_k, or factor vertex, to a variable node x_j for the SPA is computed as

$$\mu_{p_k \to X_j}(x_j) = \sum_{\sim x_j} p_k(x_1, \ldots, x_j) \prod_{j \neq i} \mu_{p_k \to X_i}(x_i), \tag{3.3}$$

where $\mu_{p_k \to X_i}(x_i)$ denotes the incoming messages and $p_k(x_1, \ldots, x_j)$ is a so-called local function. Another more detailed example how to partition a global function in these local functions will be given in Chapter 6.

Furthermore, $\sim x_j$ means that the summation is about all x_i except x_j. The corresponding SPA update rule to determine the outgoing message from a variable node x_j to a factor node p_k is just

$$\mu_{X_j \to p_k}(x_j) = \prod_{k \neq l} \mu_{p_l \to X_j}(x_j), \tag{3.4}$$

The update rules for the max-product algorithm are quite similar, the only difference is that the sum operator in (3.3) needs to be replaced by the max operator [Bis07].

In order to determine the desired marginal distribution for a certain variable X_j, all incoming messages are multiplied.

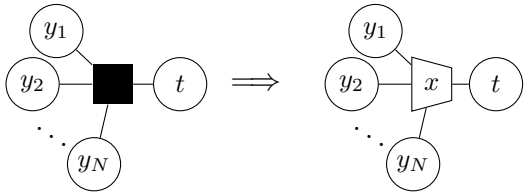


Figure 3.2: Replacing a factor vertex with an IB node transforms a general factor graph into an Information Bottleneck graph that focuses solely on transporting relevant message through the graph.

The sum-product algorithm only works exactly in cycle-free factor graphs. In most practically relevant applications related to error correction decoding these cycles exist, which requires an iterative execution of the sum-product algorithm. A more detailed discussion on how to adapt the sum-product algorithm for graphs with cycles is done in [Bis07]. Another common assumption is that the cycles are long enough, such that their influence can be neglected [Joh09]. This and further insides on iterative decoding will be given in the next chapters.

3.2 Information Bottleneck Graphs

On the one hand, factor graphs help to understand complex dependencies between different random variables in a graphical manner. On the other hand, exchanging messages between factors inside the graph allows determining really useful quantities, e.g. maximum a-posteriori estimates and marginal posterior distributions.

The messages exchanged while performing message passing algorithms are probability distributions. In the previous chapter, it was discussed that observations of an event contain not only relevant information. The idea of the Information Bottleneck method was to extract this relevant information and generate a compression variable T that is highly informative about the relevant variable X and simultaneously compresses the observation Y.

Exchanging only this compression variable T instead of the original Y during message passing would result in a significant reduction of overhead. This is the original idea of a so-called Information Bottleneck graph [LSB16a].

Information Bottleneck graphs use a new node symbol to illustrate compression mappings $p(t|\mathbf{y})$, which were designed with the Information Bottleneck method, in a factor graph. In Figure 3.2 it is shown how a factor vertex representing $p(t|\mathbf{y})$ is replaced by an Information Bottleneck node (IB node). This replacement transforms a general factor graph into an Information Bot-

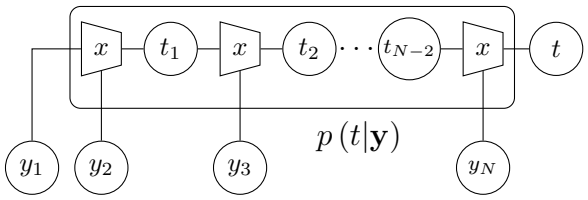


Figure 3.3: This Information Bottleneck graph basically visualizes a Markov chain like concatenation of compressions. The original operation from Figure 3.2 is split up into several simpler compression steps. Hence, a box is drawn around all these partial operations corresponding to the IB node in Figure 3.2.

tleneck graph that focuses solely on transporting relevant messages through the graph. The IB node is a trapezoid, where a vector of observations $\mathbf{y} = [y_1, y_2, \ldots, y_N]^{\mathrm{T}}$ is connected to all but the shortest sides and the compression variable is connected to the shortest side. The relevant variable is written in the center of this node.

The shape of this IB node was chosen wisely [LSB16a] since message passing in an Information Bottleneck graph is not exactly the sum-product algorithm with compressed messages. The trapezoid acts like an arrow indicating in which direction the relevant information is propagated through the graph. Hence, in more complex systems like in Figure 3.3 Information Bottleneck graphs visualize the flow of relevant information inside a system [LSB16a].

Figure 3.3 could be the corresponding Information Bottleneck graph obtained from a simple Markov chain [Bis07]. Observations gathered in the past are combined with a new evidence or observation. In state-of-the-art message passing moving only forward in time is referred to as filtering, whereas also allowing messages propagating backwards is denoted as smoothing [Bis07]. In the shown Information Bottleneck graph the focus is solely on transporting the relevant information obtained in the past to future steps, but not on backward propagation. This is also indicated by the shape of the IB node. Once the compression is applied, it is not possible to infer the original input sequence $\mathbf{y}$ uniquely.

Furthermore, there is another interesting relation between Figure 3.2 and Figure 3.3. The Information Bottleneck graph shown in Figure 3.2 indicates that all inputs $\mathbf{y}$ are compressed at a time. It is also possible to split up the compression in several partial compression, where the output of the previous IB nodes is one input together with a new observation y_i at the next IB node. Hence, one obtains a Markov chain like concatenation of simple compression operations. This approach is sometimes referred to as *opening* the node

[KYK08], [LSB16a], [LSB16b]. A box is drawn around all these IB nodes in Figure 3.3 to indicate that they perform the same task as the single node in Figure 3.2. The chosen notation is mathematically rigor, since

$$p(t|\mathbf{y}) = \sum_{\sim t} p(t_1, t_2, \ldots, t_{N-2}, t|\mathbf{y}). \tag{3.5}$$

This simple example illustrates the possibility of hierarchical modeling using Information Bottleneck graphs, meaning that a very complex systems can be split into smaller systems with lower complexity.

Chapter 4

Low-Density Parity-Check Codes

The first communication engineers designing digital communications systems were convinced that transmitting information in the presence of additive noise would always cause non-correctable transmission errors [Joh09]. In 1948, Claude Shannon's work "A Mathematical Theory of Communication" [Sha01], proved all these engineers to be wrong. Shannon claimed that by using forward error correction (FEC) codes it is possible to achieve error-free transmission as long as the rate of this code is below the capacity of a channel. This statement was formalized as

$$\mathrm{R} < \mathrm{C}, \tag{4.1}$$

where R denotes the code rate and C denotes the channel capacity. He found the capacity of a channel to be the maximum mutual information between channel input X and output Y

$$\mathrm{C} = \max_{f(X)} \mathrm{I}(X;Y), \tag{4.2}$$

where $f(X)$ is the probability density function of the channel input signal. His findings created an all new field of research, called information theory.

In (4.1) the code rate achieving error-free transmission is upper bounded, but it was not stated how to construct such a code. It was just a proof of existence. In another theorem, the so-called "Separation Theorem", Shannon stated that it is most beneficial to remove all the redundancy in a message first and then add redundant bits in a defined manner in a second step. The first step is called source compression, whereas the second step is referred to as channel coding. Consequently, having a message of K information bits the insertion of additional bits yields a longer vector of length N. This vector

is called the codeword. The ratio of information bits to codeword length defines the code rate R

$$R = \frac{K}{N}. \tag{4.3}$$

Obviously, for decoding a codeword at the receiver the insertion of redundant bits needs to follow predefined rules, which are known to the receiver. The process of adding redundant bits is called encoding. One strategy of encoding is to take one block of K information bits at a time and generate a block of N code bits. FEC codes which are generated this way are called block codes. Many different block codes exist, but from a mathematical perspective all of them, which are linear, can be represented using the simplified vector-matrix notation

$$\mathbf{c} = \mathbf{G}\mathbf{u}, \tag{4.4}$$

where $\mathbf{u}$ is a column vector of length K holding the information bits, $\mathbf{c}$ is a column vector length N holding all code bits and finally $\mathbf{G}$ denotes the $N \times K$ generator matrix. It is interesting to mention that a particular code can be generated using different generator matrices. A code is defined by the set of codewords, as long as this set is identical, it is still the same code with its same properties and only a different encoding rule. In order to explain how a block code works it is easiest to consider a special kind of block codes, namely systematic block codes. A code is called systematic if the information bits used for encoding are still a visible part of the codeword. Therefore, the generator matrix consists of two parts

$$\mathbf{G} = \begin{bmatrix} \mathbf{I_K} \\ \mathbf{P} \end{bmatrix}, \tag{4.5}$$

namely a $K \times K$ identity matrix $\mathbf{I_K}$ and a $(N - K) \times K$ matrix $\mathbf{P}$. The matrix $\mathbf{P}$ holds the equations used to calculate parity bits. The combination of these equations characterizes the code and defines how the redundant bits are constructed. These equations are also used at the receiver to distinguish between valid and non-valid codewords. A codeword is called valid if it satisfies all parity-check equations. Therefore, at the receiver

$$\mathbf{s} = \mathbf{H}\mathbf{c} = \mathbf{0} \tag{4.6}$$

needs to be fulfilled, where $\mathbf{s}$ is called syndrome. $\mathbf{H}$ is the parity-check matrix. For a systematic code, the parity-check matrix $\mathbf{H}$ can be written as [Joh09]

$$\mathbf{H} = \begin{bmatrix} -\mathbf{P}, & \mathbf{I_{N-K}} \end{bmatrix}. \tag{4.7}$$

Since all operations are performed in the Galois Field 2 (GF(2)), the minus sign in (4.7) equals the logical negation.

After Shannon's publication in 1948 many ideas about how to design codes came up and many of them were block codes. What they had in common was that the code design started with the generator matrix, admittedly an intuitive approach. By doing so many code properties could be set easily and the error correction ability is often known and fixed.

In his Ph.D. thesis from 1962 Gallager proposed a new class of codes low-density parity-check (LDPC) codes [Gal62]. The results of his Ph.D. thesis gained only minor practical relevance for a long time until they were resurrected by MacKay et al. [MN95]. One main reason was that at the time of their invention, LDPC codes could not be simulated accurately for large block lengths with existent computers. Furthermore, from a classical perspective, considering block codes being well structured and having known error correction performance, LDPC codes are the complete opposite. In contrast to classical codes, like Hamming codes, the design does not start at the transmitter by finding a generator matrix. LDPC code design starts with constructing a parity-check matrix. As the name implies this parity-check matrix is generally not well structured, but rather sparse.

There is no unique way to design a "good" LDPC code nor is there a possibility to make distinct statements about the error correction performance after the construction. In contrast to short Hamming codes, LDPC codes have to be quite long. Many different code construction techniques exist and are well explained in [Joh09] and [RL09], but will not be addressed further in this thesis.

Although the error correction performance cannot be predicted for one particular code, code ensembles can be used to yield probabilistic performance measures. One approach to determine the so-called ensemble threshold is density evolution. Density evolution and its discretized version are integral topics of this thesis and will be covered in more detailed in Chapter 5 and Chapter 6.

So far, channel coding using block codes was introduced in general. In the following, the briefly mentioned structure and properties of LDPC codes will be explained in more detail, but from a practical, applied perspective. Since the main work in this thesis focuses on discrete LDPC decoding the structure of this chapter is oriented towards a communication chain. Hence, the first section will focus on how to encode codewords for a given block of information bits. Since the parity-check matrix is of main importance for the decoding, a different, graphical notion of this matrix is brought up, so-called Tanner graphs. Accompanied by own remarks and descriptions most formulas in these two sections are derived according to [Joh09]. Finally,

the decoder of an LDPC code is explained. In addition, different decoding algorithms with varying complexity and accuracy are introduced briefly.

4.1 Encoding LDPC Codes

In the first part of this chapter it was shown how to derive the parity-check matrix $\mathbf{H}$ for a given generator matrix $\mathbf{G}$. Usually, LDPC code encoding addresses the inverse problem. For systematic codes, this does not cause big problems because analogous to (4.7) for a given parity-check matrix [Joh09]

$$\mathbf{H} = \begin{bmatrix} \mathbf{P} & \mathbf{I}_{N-K} \end{bmatrix} \tag{4.8}$$

a generator matrix satisfying

$$\mathbf{HG} = \mathbf{0}, \tag{4.9}$$

can be found as

$$\mathbf{G} = \begin{bmatrix} \mathbf{I}_{N-K} \\ -\mathbf{P} \end{bmatrix}. \tag{4.10}$$

The assumed orthogonality between generator and parity-check matrix in (4.9) follows directly from the generalization of the syndrome check equation in (4.6).

In general, randomly generated sparse parity-check matrices do not have the structure from (4.8) and therefore a systematic generator matrix cannot be found as easy as in (4.10). Hence, different techniques need to be used. The first approach is to transform an arbitrary parity-check matrix into the form from (4.8) by applying Gauß-Jordan elimination. A second option is to avoid calculating the generator matrix and encode using the parity-check matrix. This can be done by applying forward and backward substitution. Both techniques will be explained more precisely in the next sections.

4.1.1 Gauß-Jordan Elimination

The first intuitive approach is to transform the given parity-check matrix into the form from (4.8). This can be done by applying the Gauß-Jordan elimination as explained in [AORS11]. This algorithm is an extension of the well know Gauß elimination. When performing Gauß elimination the resulting matrix is called to be in row echelon form, which basically means that it has an upper triangular matrix at the left side:

$$\mathbf{A} = \left[\begin{array}{cccc|ccc} 1 & a_{1,2} & \dots & a_{1,K} & b_{1,1} & \dots & b_{1,N-K} \\ 0 & 1 & \dots & a_{2,K} & b_{2,1} & \ddots & \vdots \\ \vdots & 0 & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & b_{N-K,1} & \dots & b_{N-K,N-K} \end{array}\right] \tag{4.11}$$

This can be achieved by applying elementary row operations, namely interchanging and addition. As mentioned earlier one code can have different encoders, or generator matrices, as long as the set of codewords is the same. Elementary row operations do not change the generator matrix of a code. The next step is to transform the parity-check matrix into reduced row echelon form. A matrix is called to be in reduced row echelon form if the left part of (4.11) is basically the identity matrix, meaning that all coefficients are zero, except the diagonal which contains entries equal to one. Again, this can be achieved by performing elementary row operations. Finally, applying column permutations results in the desired, transformed parity-check matrix from (4.8).

Although all computations are done in advance, the generator matrix is not sparse resulting in a cumbersome vector-matrix multiplication for every sequence of information bits that has to be transmitted. The resulting complexity $\mathcal{O}(N^2)$ for matrix multiplication becomes problematic. On the one hand, in practically relevant scenarios LDPC codes have long codewords containing hundreds or thousands of bits. On the other hand, when analyzing the code using simulations large numbers of codewords have to be transmitted to yield sufficient error statistics [Joh09]. Hence, this approach is neither relevant for most practical implementations nor during the design evaluation.

4.1.2 Backward and Forward Substitution

Another approach is generating the codeword directly from parity-check matrix. Starting with (4.6), we can split up $\mathbf{c}$ into two parts

$$\mathbf{c} = \left[\mathbf{c}_K^{\mathrm{T}}, \mathbf{c}_{N-K}^{\mathrm{T}}\right]^{\mathrm{T}}, \tag{4.12}$$

where $\mathbf{c}_K$ holds the known information bits and $\mathbf{c}_{N-K}$ holds the desired, unknown parity bits. Splitting up $\mathbf{H}$ similarly and solving for $\mathbf{c}_{N-K}$ results in

$$\left[\ \mathbf{H}_K, \mathbf{H}_{N-K}\ \right] \left[\ \mathbf{c}_K^{\mathrm{T}}, \mathbf{c}_{N-K}^{\mathrm{T}}\ \right]^{\mathrm{T}} = \mathbf{0} \tag{4.13}$$

$$\mathbf{H}_{N-K}\mathbf{c}_{N-K} = \mathbf{H}_K\mathbf{c}_K \tag{4.14}$$

$$\mathbf{c}_{N-K} = \mathbf{H}_{N-K}^{-1}\mathbf{H}_K\mathbf{c}_K. \tag{4.15}$$

Due to the sparseness of the matrix in the first multiplication, this operation is computational less expensive. On the other hand, calculating the inverse of a large square matrix is $\mathcal{O}(N^3)$ complex. Hence, one rather stops at (4.14) and solves the system of equations. In order to solve

$$\mathbf{H}_{N-K}\mathbf{c}_{N-K} = \mathbf{H}_K\mathbf{c}_K, \tag{4.16}$$

$\mathbf{H}_{N-K}$ is first decomposed in an upper triangular matrix $\mathbf{U}$ and a lower triangular matrix $\mathbf{L}$. An efficient algorithm, called LU-decomposition, exists [AORS11]. The LU-decomposition in its general form is written as

$$\mathbf{P}\mathbf{H}_{N-K} = \mathbf{L}\mathbf{U}, \tag{4.17}$$

where $\mathbf{P}$ denotes a pivot matrix, permuting the rows in $\mathbf{H}_{N-K}$. Please note, that all computations need to be performed in the GF(2). Once the decomposition is calculated, the vector $\mathbf{z}$ which solves

$$\mathbf{L}\mathbf{z} = \mathbf{P}\mathbf{H}_K\mathbf{c}_K \tag{4.18}$$

is determined by forward substitution. Finally,

$$\mathbf{U}\mathbf{c}_{N-K} = \mathbf{z} \tag{4.19}$$

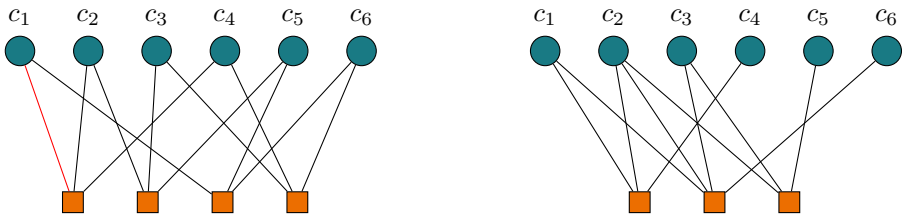
is solved by backward substitution to determine the missing parity bits.

The overall complexity reduction results from an efficiency backward and respectively forward substitution and the LU-decomposition with complexity $\frac{2}{3}\mathcal{O}(N^3)$ According to [Joh09], this approach allows encoding of LDPC codes in almost linear time. The only requirement is that $\mathbf{H}_{N-K}$ can be inverted in GF(2), otherwise column or row permutations have to be applied.

Since one part of this master thesis is verifying approaches for discrete decoder design by simulations and compare obtained results to state-of-the-art decoder, implementing an encoder to generate codewords is essential. Following the discussion above, the second encoding method using LU-decomposition was implemented.

4.2 Tanner Graphs

Block codes are typically investigated using tools from linear algebra. When dealing with LDPC codes the structure of the parity-check matrix is of crucial importance. However, investigating sparse matrices with only very few non-zero entries can be confusing. Therefore, a graphical representation of the parity-check matrix was introduced, namely Tanner graphs. Tanner graphs



(a) (b)

Figure 4.1: Tanner graph for a regular LDPC code. The colored edge corresponds to the colored one in matrix (4.20).

belong to the class of bipartite graphs. This kind of graphs is used to visualize the connections between two disjoint sets. The two sets of an LDPC code tanner graph are called variable or bit nodes and check nodes. Although this section deals with Tanner graphs, outlining relations to the matrix representation are necessary to understand the construction of Tanner graphs.

The given parity-check matrix

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \tag{4.20}$$

is considered. It should be mentioned that (4.20) defines a bad LDPC code because $\mathbf{H}$ is not sparse and also very small. However, the corresponding Tanner graph is shown in Figure 4.1a. The upper set represented by blue circles corresponds to variable nodes. The lower set of orange rectangles illustrates check nodes. Each row in (4.20) represents one check node equation. According to the vector-matrix multiplication from (4.6) each column of (4.20) is multiplied by one information bit and specifies in which parity-check equation a bit is involved. Consequently, each column represents a variable node and each row corresponds to a check node. Since a connection between a check node and variable node can only either exist or not, a one in the matrix indicates a connection, whereas zero means no connection. This relation is visualized by an edge in the graph. For illustration purposes, one edge is colored in red and the corresponding entry in the parity-check matrix (4.20) is colored in red too.

Having transferred the parity-check matrix representation into a Tanner graph allows analyzing the graph. In section 4.3 message passing algorithms will be introduced sending extrinsic information along the edges of a Tanner graph. It can be seen that so-called cycles exist in the graph. A cycle is a sequence of connected nodes which starts and ends at the same node

and does not touch other nodes in between more than once. The length of such cycles can influence the decoding performance significantly. Long cycles are preferred, whereas the shortest one, called girth [Joh09], bounds the performance.

Further important quantities are the node degree and the closely related node-degree distribution. In a first step, all possible node degrees are determined by counting the number of connections at a node. The parity-check matrix (4.20) has only one variable node degree d_v and one check node degree d_c

$$d_v = 2, \forall v \in V \tag{4.21}$$

$$d_c = 3, \forall c \in C \tag{4.22}$$

A code with one distinct check node degree and also only one particular variable node degree is called regular. Consequently, the code from (4.20) is regular.

Figure 4.1b shows a Tanner graph of a code which is different in terms of the so-called degree distribution. The degree distribution can be described in two different ways. One possibility is providing the fraction of nodes with a particular node-degree in vector notation. This approach yields the so-called node-degree distribution [Joh09]. To characterize a code, two vectors

$$\mathbf{d}_v = [d_{v,1}, d_{v,2}, \cdots, d_{v,\max}]^{\mathrm{T}}$$

and

$$\mathbf{d}_c = [d_{c,1}, d_{c,2}, \cdots, d_{c,\max}]^{\mathrm{T}},$$

holding the fraction of rows or columns having the same weight i, are defined. The smallest possible weight is $i = 1$ and the fraction of rows or columns with weight one are stored in the first entry of $\mathbf{d}_v$, i.e. $d_{v,1}$, or respectively in the first entry of $\mathbf{d}_c$, i.e. $d_{c,1}$. For the example shown in Figure 4.1b this yields

$$\mathbf{d}_v = \left[\frac{1}{2}, \frac{1}{3}, \frac{1}{6}\right]^{\mathrm{T}} \tag{4.23}$$

$$\mathbf{d}_c = \left[0, 0, \frac{2}{3}, \frac{1}{3}\right]^{\mathrm{T}}. \tag{4.24}$$

In contrast to the first example this code has different node degrees and is called to be irregular.

Another possibility of characterizing an LDPC code ensemble is from an edge perspective. Hence, this approach is called edge-degree distribution

[Joh09]. This distribution is better suited to understand density evolution in Chapter 5 and also for the generalization of discrete density evolution in Chapter 8. When using the edge-degree distribution the fraction of edges connected to variable nodes with degree i is denoted as λ_i. The fraction of edges connected to check nodes with degree i is denoted as ρ_i. It is possible to transform the node-degree into the edge-degree distribution by applying

$$d_{v,i} = \frac{\lambda_i / i}{\sum_j \lambda_j / j} \tag{4.25}$$

$$d_{c,i} = \frac{\rho_i / i}{\sum_j \rho_j / j} \tag{4.26}$$

In (4.3) it was defined how to compute the rate of a block code in general. For an LDPC code, this formula can be expressed in terms of the degree distribution as

$$R = \frac{K}{N} = 1 - \frac{\sum_i d_{v,i} i}{\sum_j d_{c,j} j} = 1 - \frac{\sum_j \rho_j / j}{\sum_i \lambda_i / i} \tag{4.27}$$

for a full rank parity-check matrix.

For a regular LDPC code (4.27) simplifies to

$$R = 1 - \frac{d_v}{d_c} \tag{4.28}$$

4.3 Decoding LDPC Codes

Digital transmission over noisy channel always suffers from bit errors. The task of a decoder is to detect and correct errors if possible. For block codes measures like the Hamming distance can be used to predict the detection and correction capabilities of a code. The optimum decoding strategy in terms of the word error rate is maximum a-posteriori decoding because it minimizes the expected posterior loss for detection problems. This decoding approach compares the received sequence to all possible codewords and chooses the most likely one. This can be formulated as

$$\hat{\mathbf{u}} = \arg\max_{\mathbf{u}} p(\mathbf{u}|\mathbf{y}). \tag{4.29}$$

In this notation $p(\mathbf{u}|\mathbf{y})$ is the a-posteriori probability for a given received vector $\mathbf{y}$, which considers the statistics of the channel and a-priori knowledge. For short codes this approach is optimum and the comparison is feasible. For

LDPC codes this approach would also be optimal but the number of codewords to compare with is huge, since for a code with K information bits, 2^K codewords exist. However, other techniques with much less complexity performing close to bitwise a-posteriori performance exist. Such techniques make use of the induced structure of the code, e.g. the sparseness of the parity-check matrix. As shown in the previous section parity-check matrices can be visualized using Tanner graphs. The idea is to send extrinsic information along the edges of the graph. This class of decoding algorithms is called message passing. The message exchange has to be repeated iteratively until all information needed to decode the received sequence is distributed throughout the entire graph. Therefore, message passing algorithms belong to the class of iterative decoding techniques. It turns out that if the girth is large, this iterative message exchange yields near optimum performance [Joh09].

The main complexity reduction compared to maximum a-posteriori decoding is achieved by message passing decoding, but the graph based receiver design offers another degree of freedom, that is, the complexity of the applied node operations. One bottleneck of iterative decoding is the limited speed when the number of iterations increases. Hence, two options exist, namely simplifying the node operations, resulting in a speed up due to complexity reduction or reducing the number of iterations. The second option is typically not a good idea because especially for signal-to-noise ratios close to the decoding threshold many iterations are required to obtain good error correction performance.

Hence, three state-of-the-art decoding algorithms performing different operations at the variable and check nodes are introduced. The bit flipping algorithm belongs to the class of hard decision decoding techniques, whereas the two other algorithms are soft decision decoding algorithms. The name hard decision indicates a high compression of the transmitted messages because they can take only binary states, namely 0 and 1. Soft decision techniques, however, exchange probabilities or more precise, scalar representatives called log-likelihood ratio (LLR). These LLRs can take a high dynamic range and need to be represented with high resolution in order to obtain the best decoding results. The different advantages and drawbacks of the three briefly mentioned message passing algorithms will be discussed in more detail in the next sections.

4.3.1 Bit Flipping Decoding

As mentioned above the messages sent along an edge in the Tanner graph are binary. Hence, this hard decision message passing algorithm causes only small overhead when messages are exchanged and also the operations at the

variable and check nodes are quite simple. The algorithm starts with an initialization of all variable nodes. The values used for this step are obtained from the detector directly and binary. In the first iteration these bits are passed along the edges to connected check nodes. The task of a check node is to generate extrinsic information. For generating extrinsic information about bit i at the jth check node, the incoming knowledge about bit i is ignored and the value that satisfies the jth parity-check equation, by taking the other incoming bits into account, is determined. The result is the outgoing bit on the ith edge of the jth check node. This procedure is repeated at all check nodes for all outgoing bits. In the next step, the situation has changed and the messages are sent from the check nodes to the variable nodes. The node operation at a variable node is similar to a repetition code. A majority decision is performed on all incoming bits and if the majority differs from the received bit then it is flipped. Now one iteration is done. This procedure is repeated until all parity-checks are satisfied or until the maximum number of iterations is reached. The drawback of this approach is the rather poor decoding performance, caused by the limited number of states, which cannot maintain any information about the certainty of the decision.

4.3.2 Belief Propagation Decoding

As mentioned above the missing presence of an uncertainty measure generally impairs the performance of hard decision algorithms. This is taken care of when performing belief propagation decoding. As the name implies, the idea is that a more or less save belief propagates through the graph. The belief is a probabilistic quantity and generated from a probability distribution. In the most general setting, the belief is just a probability distribution itself. When dealing with a discrete, binary event space a scalar representative can be computed. This value is called log-likelihood ratio (LLR). For a binary random variable X, the LLR $L(x)$ is defined as

$$L(x) = \log \frac{\Pr(X = +1)}{\Pr(X = -1)}. \tag{4.30}$$

It is interesting to notice that the sign of $L(x)$ provides the same information about x as obtained from hard decision and the magnitude $|L(x)|$ indicates the reliability of the belief. From a practical perspective, it can be seen that numerical instabilities can occur if the ratio inside log approaches 0 or if $\Pr(X = -1)$ is close to zero. Furthermore, on the first glance already quite small magnitudes like 10 express high certainties

$$\Pr(X = +1) = \frac{e^{L(x)}}{1 + e^{L(x)}} = \frac{e^{10}}{1 + e^{10}} = 0.99995 \tag{4.31}$$

In the next step, the node operations need to be adapted to handle this new measure of belief. However, the general procedure is quite similar to bit flipping decoding. The majority vote at the variable node for hard decision is replaced by a multiplication of beliefs. Multiplying beliefs corresponds to a summation of LLRs. Denoting the channel output LLR as $L_c(x_i)$, and extrinsic LLRs received from check nodes over edge j as $L_{\text{ext},j}(x_i)$, yields

$$L_i(\hat{x_i}) = L_c(x_i) + \sum_j L_{\text{ext},j}(x_i) \tag{4.32}$$

at the ith variable node for the final bit decision. Since only the extrinsic information should be sent back along the edge, the LLR received from this edge is excluded during the iterative decoding

$$L_{\text{ext},j}(x_i) = L_c(x_i) + \sum_{j \neq j'} L_{\text{ext},j'}(x_i). \tag{4.33}$$

The channel output LLR $L_c(x_i)$ is computed using conditional probabilities [Joh09]

$$\begin{aligned} L_c(x_i) &= \log \frac{\Pr(X_i = +1|y_i)}{\Pr(X_i = -1|y_i)} && (4.34)\\ &= \log \frac{\Pr(y_i|X_i = +1)\Pr(X_i = +1)}{\Pr(y_i|X_i = -1)\Pr(X_i = -1)} && (4.35)\\ &= \underbrace{\log \frac{\Pr(y_i|X_i = +1)}{\Pr(y_i|X_i = -1)}}_{\text{channel information}} + \underbrace{\log \frac{\Pr(X_i = +1)}{\Pr(X_i = -1)}}_{\text{a-priori information}} && (4.36) \end{aligned}$$

This formula serves as a starting point in later chapters when information optimum quantizer for the channel outputs are derived. Since all soft decision decoding algorithms are initiated with these LLRs, keeping as much relevant information as possible about the channel output is of crucial importance.

Deriving the operation applied to LLRs at the check nodes is a bit cumbersome. As for hard decision decoding the parity-check equations have to be solved. The elementary operation for this task is the (mod 2)-sum. Computing $L(x_i \oplus x_j)$ results in

$$L(x_i \oplus x_j) = \frac{e^{L(x_i)}e^{L(x_j)} + 1}{e^{L(x_i)} + e^{L(x_j)}} = L(x_i) \boxplus L(x_j), \tag{4.37}$$

where $L(x_i) \boxplus L(x_j)$ is called the box-plus operation [HOP96]. The overall equation for a general check node equals

$$L_{\text{ext},j}(x_i) = \sum_{j \neq j'} \boxplus L_{\text{ext},j'}(x_i). \tag{4.38}$$

Similar to the bit flipping decoding the belief propagation algorithms stops if all check equations are satisfied or the maximum number of iterations is reached.

4.3.3 Min-Sum Decoding

The box-plus operation performed at the check nodes is very computationally expensive and different ways to simplify this expression exist. One alternative is to use min-sum decoding instead of pure belief propagation. Rewriting (4.38) as

$$\sum_{j \neq j'} \boxplus L_{\text{ext},j'}(x_i) = \log \frac{1 + \prod_{j \neq j'} \tanh\left(L_{\text{ext},j'}(x_i)/2\right)}{1 - \prod_{j \neq j'} \tanh\left(L_{\text{ext},j'}(x_i)/2\right)} \tag{4.39}$$

indicates that the product term is dominated by the $L_{\text{ext},j'}(x_i)$ with the smallest magnitude $|L_{\text{ext},j'}(x_i)|$. Furthermore, the (mod 2)-sum performed for hard decision equals the multiplication of $L_{\text{ext},j'}(x_i)$ signs. Consequently, this approximation yields

$$L_{\text{ext},j}(x_i) \approx \prod_{j \neq j'} \text{sign}\left\{L_{\text{ext},j'}(x_i)\right\} \min\left\{|L_{\text{ext},j'}(x_i)|\right\} \tag{4.40}$$

The resulting complexity reduction follows directly from the simplified node operation. However, the exchanged messages need to be represented with high precision, so there is no complexity reduction during the message exchange.

Chapter 5

Density Evolution

In the previous chapter, LDPC codes were introduced. These codes are very powerful error correction codes, which can operate close to the theoretical optimum. But this is only true, if the degree distribution and other properties are set correctly. The error correcting behavior of an LDPC code is strongly non-linear, i.e once a channel parameter, e.g. the noise variance of an additive white Gaussian noise (AWGN) channel, reaches the threshold, the bit error rate (BER) drops dramatically. This region is called the waterfall region. Due to cycles inside the Tanner graph, the assumption that only extrinsic information is exchanged in every iteration, is violated if the number of iterations extends twice the girth. Hence, even for high signal-to-noise ratios (SNR), the BER does not fall below a certain value. This behavior is called error floor and the corresponding region is known as error floor region.

When constructing LDPC codes, one is interested in the threshold at which the code starts correcting all errors after a sufficient number of performed iterations. Ideally, one would like to determine this point in advance, in order to evaluate the suitability of an LDPC code for a certain application. Due to the often random construction and other factors the performance prediction for a specific code is not possible yet and still a big field of research [Joh09].

As mentioned earlier, an approach evading this impairment is to consider a probabilistic quantity the so-called code ensemble. A code ensemble represents the set of all possible realizations of a code with certain properties, like the degree distribution. A technique called density evolution allows evaluating the average threshold performance of a code ensemble. As the name implies, the idea is to track the evolution of probability distributions of the messages, which are passed around during the iterative decoding. In the end, the resulting probability distribution indicates if errors remain or not. Thus, one can adjust the noise power in a simulation accordingly un-

til the threshold is found. Passing around continuous probability density functions with an event space bounded by $\pm\infty$ can hardly be realized on a computer. Hence, discretized and simplified versions of density evolution were developed [KYK08]. In general, performing density evolution fast and with limited complexity but the same precision and validity is of interest. One alternative concept to approximate density evolution is designing extrinsic information transfer (EXIT) charts. When using EXIT charts only the mutual information of LLRs with underlying Gaussian densities are considered, instead of tracking the original, maybe not Gaussian, probability density function [Joh09].

The main focus of this thesis is on a different, more general approach firstly presented by Kurkoski, et. al. in 2008 [KYK08]. In their publication "Noise Thresholds for Discrete LDPC Decoding Mappings" [KYK08] they proposed an intelligent way of splitting up check and variable nodes and compressing the obtained intermediate results of the partial node operations without significantly reducing the amount of exchanged information.

A detailed explanation of this new, discrete density evolution scheme will follow in the next chapter. Beforehand, continuous density evolution will be explained together with all assumptions and requirements. Thereafter, density evolution for regular LDPC codes will be introduced. The following two sections explain how to lower the threshold by using irregular codes and how density evolution can be adopted for irregular LDPC codes.

5.1 Prerequisites

Density evolution is a powerful, generic technique to find the decoding threshold of LDPC codes. However, the concept itself is based upon several assumptions regarding the properties of iterative LDPC code ensembles. These assumptions will be listed, introduced and explained in the following. The main references for this section are [Joh09] and [RL09].

Symmetric distributions For density evolution symmetry of the transmission channel is assumed, meaning that the channel transition probability density function satisfies $f(y) = p(y|X = 1) = p(-y|X = -1)$. Consequently, the output LLRs are also symmetric.

All-zeros codeword For symmetric channel distribution it can be proven that the iterative decoding performance is independent of the actual transmitted codeword. Consequently, for simplicity, only the all-zeros codeword is used to evaluate the decoding performance.

Concentration theorem This theorem states that the a randomly chosen code from the ensemble will show a performance close to the ensemble average performance with high probability when the codeword length tends to infinity.

Cycle-free graphs For codeword length approaching infinity the performance achieved by an iterative decoder approaches the one obtained in a cycle-free graph. As mentioned above, as long as the number of iteration is smaller than two times the girths, exchanged message can be assumed as independent.

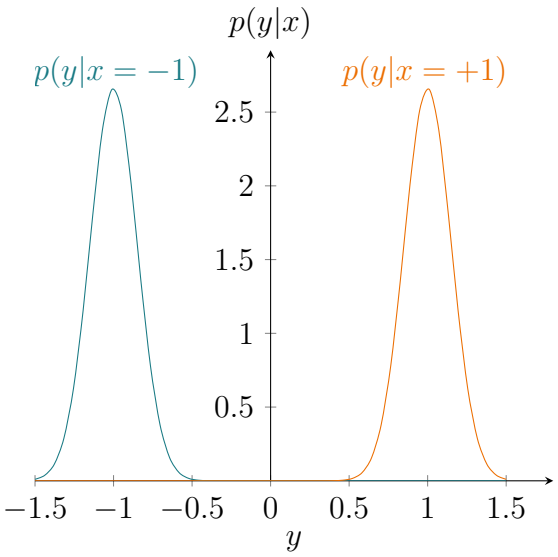
Density evolution is a general approach and works for any symmetric channel. In this thesis, the binary-input AWGN (BI-AWGN) channel will be used for simulation. However, the simpler binary erasure channel (BEC) is better suited for an easy understanding of density evolution. In Figure 5.1a the conditional probability density functions of a BI-AWGN channel output are shown. It can be seen that the channel output y is continuous, i.e. $y \in \mathbb{R}$. Figure 5.1b visualizes the BEC model. In contrast to the AWGN channel, the channel output of a BEC is discrete and can take only three states $y \in \{-1, e, +1\}$. The probability that a bit is erased is ϵ and $1 - \epsilon$ denotes the probability of a correct transmission.

Since only the all-zeros codeword needs and will be considered, what density evolution basically does, is a simultaneous, probabilistic, iterative decoding of all possible channel outputs of the transmitted all-zeros codeword. Repeating this procedure for different channel parameters, like noise variance, finds the threshold for an LDPC code ensemble which partitions the channel parameter space in one region where error free transmission is possible and another where errors remain.

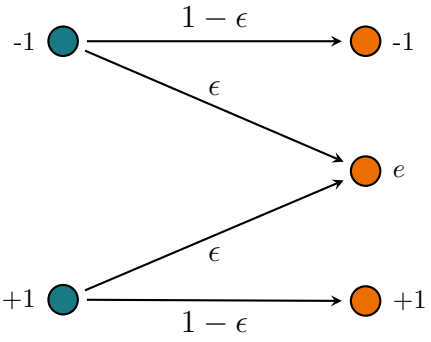
The last assumption regarding cycle-free graphs contains a common information theoretical prerequisite, namely that most of the obtained theoretical results are only true for infinitely long codewords as well as infinitely many iterations. Obviously, this assumption will be violated in practice. Nevertheless, it turns out the performance degradation is often acceptably small [Joh09].

5.2 Density Evolution for Regular LDPC Codes

At first finding the threshold of a regular LDPC code ensemble $\mathcal{T}(d_c, d_v)$ with check node degree d_c and variable node d_v is derived. Density evolution models the exchanged messages m as random variables. The probability density function of a message $m^{(v)}$, which is passed from a variable node v



(a) Conditional probabilities for a BI-AWGN channel output



(b) Binary erasure channel

Figure 5.1: In this figure two very common channel models are shown. The BI-AWGN channel will serve as channel model for simulations throughout the thesis. The binary erasure channel is better suited as a simplified introduction to density evolution.

to a check node at iteration l is denoted as $p_l^{(v)}$. Since only the all-zeros codeword $\mathbf{u} = [0, 0, \ldots, 0]^{\mathrm{T}}$ is sent, the sequence $\mathbf{x} = [+1, +1, \ldots, +1]^{\mathrm{T}}$ will be transmitted according to the defined bit mapping. Consequently, an error is made if any variable node a has negative LLRs $L^{\mathrm{total}}(\hat{x}_i)$ after the maximum number of iterations. In mathematical terms, no errors are made if

$$\lim_{l\to\infty} \int_{-\infty}^{0} p_l^{(v)}(\alpha) d\alpha = 0, \tag{5.1}$$

because $L^{\mathrm{total}}(\hat{x}_i)$ cannot be negative if (5.1) holds. $p_l^{(v)}(\alpha)$ depends on the channel parameters α. For a BEC α is the erasure probability ϵ and α is the standard deviation σ of the Gaussian channel noise for a BI-AWGN. Consequently, the threshold can be defined as [RL09]

$$\alpha^* = \sup \left\{ \alpha : \lim_{l\to\infty} \int_{-\infty}^{0} p_l^{(v)}(\tau) d\tau = 0 \right\}. \tag{5.2}$$

Developing $p_l^{(v)}$ and $p_l^{(c)}$ for the BEC is straightforward. The check-to-variable message $m^{(c)}$ will be e if any of the $d_c - 1$ incoming processed messages $m^{(v)}$ is e. Hence,

$$p_l^{(c)} = 1 - (1 - p_l^{(v)})^{d_c - 1}. \tag{5.3}$$

The message $m^{(v)}$ at iteration l will be an erasure e, if all incoming messages $m^{(c)}$ are erasures, which is true with probability $p_{l-1}^{(c)}$, as well as the initial value also being an erasure. This can be denoted as

$$p_l^{(c)} = \epsilon (p_{l-1}^{(c)})^{d_v - 1} \tag{5.4}$$

and by substitution rewritten as

$$p_l^{(c)} = \epsilon \left(1 - (1 - p_{l-1}^{(v)})^{d_c - 1}\right)^{d_v - 1}. \tag{5.5}$$

Therefore, when denoting the initial value $p_0^{(v)}$, with $p_0^{(v)} = \epsilon$, density evolution for $\mathcal{T}(d_c, d_v)$ and BEC equals

$$p_0^{(v)} = \epsilon, p_l^{(v)} = \epsilon \left(1 - (1 - p_{l-1}^{(v)})^{d_c - 1}\right)^{d_v - 1} \tag{5.6}$$

The general derivation of $p_l^{(v)}$ and $p_l^{(c)}$ requires some more mathematics. As derived in section 4.3 the outgoing message at a variable node equals

$$m^{(v)} = m_{\text{channel}} + \sum_{j=1}^{d_v - 1} m_j^{(c)}, \tag{5.7}$$

where m_0 denotes the channel message and $m_j^{(c)}$ denotes the incoming check messages. When assuming independence of the check node messages $m^{(c)}$ the resulting probability density function is the convolution

$$p_l^{(v)} = p_{\text{channel}} * \left[p_{l-1}^{(c)}\right]^{*(d_v - 1)}. \tag{5.8}$$

The $(d_v - 1)$ fold convolution can be applied, because the considered code ensemble is regular, i.e. all probability density functions $p_j^{(c)}, j > 0$ are identical. The computation of $p_l^{(v)}$ can be implemented by a fast Fourier transform

$$p_l^{(v)} = \mathcal{F}^{-1} \left\{ \mathcal{F} \left\{ p_{\text{channel}} \right\} \left[\mathcal{F} \left\{ p_{l-1}^{(c)} \right\} \right]^{d_v - 1} \right\} \tag{5.9}$$

Deriving a technique to compute $p_l^{(c)}$ is more complicated. In section 4.3 it was shown that determining the outgoing check node message involves the box-plus operation. Rewriting (4.39), yields

$$m_j^{(c)} = \sum_{j \neq j'} \boxplus m_{j'}^{(v)} = \log \frac{1 + \prod_{j \neq j'} \tanh\left(m_{j'}^{(v)}/2\right)}{1 - \prod_{j \neq j'} \tanh\left(m_{j'}^{(v)}/2\right)}. \tag{5.10}$$

The necessary steps to compute $p_l^{(c)}$, by transforming the incoming message probability density functions correctly according to (5.10), are quite long. A very detailed derivation is given in [RL09]. For simplicity only the fundamental ideas will be introduced. The first step is to determine the probability density function of the new auxiliary random variable

$$z_i = \phi\left(m_j^{(v)}\right) = -\log\tanh\left(\frac{m_j^{(v)}}{2}\right).$$

The resulting transformation will be denoted as $\Gamma\left[p_l^{(v)}\right]$. The second step starts with rewriting the product inside the log in (5.10) as sum. The summation of the independent, transformed random variables z_i involved, is again a convolution. Similarly, to (5.8) this can be written as

$$p(w) = p(z)^{*(d_c-1)} = \Gamma\left[p_l^{(v)}\right]^{*(d_c-1)}, \tag{5.11}$$

where w denotes another auxiliary random variable

$$w = \sum_{i=1}^{d_c-1} z_i.$$

Rewriting as sum requires an additional inversion ϕ^{-1} [RL09]. Hence, $p_l^{(c)}$ can be found compactly as

$$p_l^{(c)} = \Gamma^{-1}\left[\Gamma\left[p_l^{(v)}\right]^{*(d_c-1)}\right] \tag{5.12}$$

Substituting (5.12) in (5.8) yields the iterative general update equation for density evolution considering regular LDPC code ensembles

$$p_l^{(v)} = p_{\text{channel}} * \left[p_{l-1}^{(c)}\right]^{*(d_v-1)} \tag{5.13}$$

$$= p_{\text{channel}} * \left(\Gamma^{-1}\left[\Gamma\left[p_{l-1}^{(v)}\right]^{*(d_c-1)}\right]\right)^{*(d_v-1)}. \tag{5.14}$$

A more detailed understanding of (5.14) is not required, since the more important formulas for discrete density evolution are derived differently and involve less complex computations. Nevertheless, (5.14) is not only given for completeness but also needed as starting point for the generalization of discrete density evolution for irregular LDPC codes in Chapter 8.

5.3 Threshold Improvement using Irregular LDPC Codes

Having obtained a possibility to determine the decoding threshold, moving this threshold closer to the theoretical limit is desirable. In the previous chapter, a more flexible class of LDPC codes was introduced which improves the decoding performance significantly, i.e. irregular LDPC codes. In a Tanner graph, irregular codes have varying variable and check node degrees. Variable nodes with a high node degree can generate high LLR values, indicating low uncertainty and can thereby help to decode bits related to nodes with a lower degree. When generalizing density evolution for irregular LDPC codes, the fraction of edges connected to a certain node degree is of interest. In the previous chapter, this representation was introduced as edge-degree distribution. An alternative mathematical notion is to express the edge-degree distribution as polynomials

$$\lambda(x) = \sum_{i=1}^{d_{v,\max}} \lambda_i x^{i-1} = \lambda_1 + \lambda_2 x + \lambda_3 x^2 + \cdots \tag{5.15}$$

$$\rho(x) = \sum_{i=2}^{d_{c,\max}} \rho_i x^{i-1} = \rho_2 x + \rho_3 x^2 + \cdots . \tag{5.16}$$

5.4 Density Evolution for Irregular LDPC Codes

In this section, the update equations from (5.12) and (5.14) are extended to irregular codes.

At first, (5.8) needs to be adapted, to account for all possible variable node degrees and their appearance probabilities

$$\begin{aligned} p_l^{(v)} &= p_{\text{channel}} * \sum_{i=1}^{d_{v,\max}} \lambda_i \cdot \left(p_{l-1}^{(c)}\right)^{*(i-1)} \\ &= p_{\text{channel}} * \lambda_* \left(p_{l-1}^{(c)}\right). \end{aligned} \tag{5.17}$$

According to (5.17), the weighting is done by computing the expectation over the message distributions. The second line uses a compact notation similar to (5.15) and (5.16), but with $(i-1)$-fold convolution instead of multiplication.

The same approach has to be applied to (5.12) resulting in

$$p_l^{(c)} = \Gamma^{-1}\left[\sum_{i=2}^{d_{c,\max}} \rho_i \cdot \Gamma\left[p_l^{(v)}\right]^{*(i-1)}\right] \tag{5.18}$$

$$= \Gamma^{-1}\left[\rho_*\left(\Gamma\left[p_l^{(v)}\right]\right)\right]. \tag{5.19}$$

Obviously, combining (5.17) and (5.19) yields the general update rule for irregular LDPC codes

$$p_l^{(v)} = p_{\text{channel}} * \lambda_*\left(\Gamma^{-1}\left[\rho_*\left(\Gamma\left[p_{l-1}^{(v)}\right]\right)\right]\right). \tag{5.20}$$

Density evolution itself only determines the threshold for a given pair of degree distributions, but gives no information on how to find good degree distributions. To find the degree distribution minimizing the threshold, an additional optimization algorithm can be used [RL09].

Nevertheless, finding optimal codes or how to construct them is beyond the scope of this thesis.

Chapter 6

Construction of Discrete Decoders

Previous chapters served as general introduction to the broad fields of information theory, error correction codes, graphical models to solve inference problems and LDPC code analysis. All these theoretical foundations are important because combining them lays the groundwork for this thesis. This chapter will successively introduce and explain all parts needed for constructing discrete decoders.

In state-of-the-art signal processing units, certain tasks can become a major bottleneck regarding time or energy consumption. It is the central aim of discrete decoder design to create decoders, which yield the same or better performance as existing techniques, by executing only simple operations. These simplified operations require only a small word length, i.e. the results can be represented using a small number of bits. LDPC decoders are perfectly suited as application due to a couple of reasons. Firstly, decoding belongs to the general class of detection problems which can be solved by the Information Bottleneck method as described in Chapter 2. Furthermore, since LDPC decoders use message passing algorithms and can be described by factor graphs, it is easy to describe them in Information Bottleneck graphs. Using this graphical extension of the Information Bottleneck method, the flow of information through a complex system can be understood easily, resulting in an intuitive and straightforward design strategy. Thirdly, the individual node operations like box-plus are computationally complex and it is desirable to simplify these operations. Consequently, LDPC decoders are a promising application.

The Information Bottleneck requires a joint probability distribution $p(x, y)$ of the relevant variable X and the observed random variable Y. At first, this joint probability distribution needs to be known to simplify the processing

chain with the Information Bottleneck method. A general discussion how to find these functions for density evolution was carried out in Chapter 5. As already outlined in the introduction the main focus of this thesis lies on a different density evolution scheme firstly presented by Kurkoski [KYK08], called discrete density evolution.

A detailed explanation of this technique will be the focus of this chapter. Beforehand, the design of information optimum quantizers is explained, since quantization is the mandatory first step when applying discrete decoders.

6.1 Relevant Information Preserving Quantizer Design

Converting a continuous signal into a discrete one is known as quantization [Kam13]. Finding optimal representatives of a quantizer is a well-known problem in rate distortion theory. A lot of work on how to find these representatives has been done and is summarized in [Kam13].

In contrast to state-of-the-art quantizers, the focus of information optimum quantization is explicitly on maintaining relevant information. Thus, it is possible to use the Information Bottleneck method to determine optimum clusterings of numerous channel output states. In state-of-the-art signal processing an analog, continuous signal is quantized such that only a limited number of distinct representation levels exist. These levels are still closely related to the original, possibly physical, quantity which the analog input signal expressed. Using the representation points, one obtains an intuition of the original input signal directly because, typically, the representatives are still elements of the input signal's domain.

Information optimum quantizers are much less intuitive to understand, because they map the values of the continuous, physical domain directly onto an abstract compression space spanned by the compression variable T. As explained in Chapter 2 the Information Bottleneck method returns two distributions. The first conditional distribution, often interpreted as a look-up table, describes the mapping from the observation onto the compression variable. The second conditional distribution $p(x|t)$, holds the meanings of distinct clusters $t \in \mathcal{T}$ for the relevant variable X. Therefore, the quantizer outputs only the cluster indices according to the first table $p(t|y)$ but does not indicate any value allowing conclusions about the original physical input value itself. To gain information about the relevant variable the appropriate $p(x|t)$ for a particular t has to be considered.

6.1.1 Channel Model

As explained in Chapter 2 the Information Bottleneck method requires the joint probability distribution $p(x, y)$ between the observed random variable Y and the relevant variable X. During the discussion of density evolution the BI-AWGN channel was introduced. This channel model will be used from now on. A given input bit $u = 0$ will be mapped onto a transmit symbol $x = +1$ and the input bit $u = 1$ will be mapped onto $x = -1$. Hence, the input alphabet $\mathcal{A}_{\text{in}}$ equals $\mathcal{A}_{\text{in}} = \{-1, +1\}$ and the output alphabet $\mathcal{A}_{\text{out}}$ is $\mathcal{A}_{\text{out}} = \mathbb{R}$. The conditional probability distribution, also called transition probability distribution [Kam13], of a BI-AWGN channel is given by

$$p(y|x = \pm 1) = \frac{1}{\sqrt{2\pi\sigma_N^2}} e^{-\frac{(y\mp 1)^2}{2\sigma_N^2}}, \tag{6.1}$$

where σ_N^2 denotes the noise variance.

6.1.2 Information Optimum Quantizer

Assuming a uniform prior on X, i.e. $\Pr(X = +1) = \Pr(X = -1) = 0.5$, an information optimum quantizer [LB15] can be generated easily by calculation of the joint distribution $p(x, y) = p(y|x)p(x)$ using (6.1).

The structure of $p(x, y)$ allows applying the symmetric sequential Information Bottleneck algorithm from Chapter 2. Figure 6.1 shows the resulting boundaries of the clusters obtained for a compression cardinality $|\mathcal{T}_{\text{c}}| = 16$, i.e. 4 bit quantization. It can be seen that the cluster borders are moving closer together as the noise variance σ_N^2 decreases. This is a reasonable result because if the variance decreases, the region where a wrong decision might occur also gets smaller. Hence, the quantizer focuses on this smaller region, because it is most relevant.

The quantizer is the first element in every digital receiver connecting the physical domain with a digital signal processor. The outputs of this device are forwarded to different signal processing units. As explained earlier, in information optimum design dealing with cluster representatives, which need to be represented as high precision floating point numbers, is not necessary anymore. Instead, only integers are forwarded and thus all subsequent units have to be simplified to work only on cluster indices.

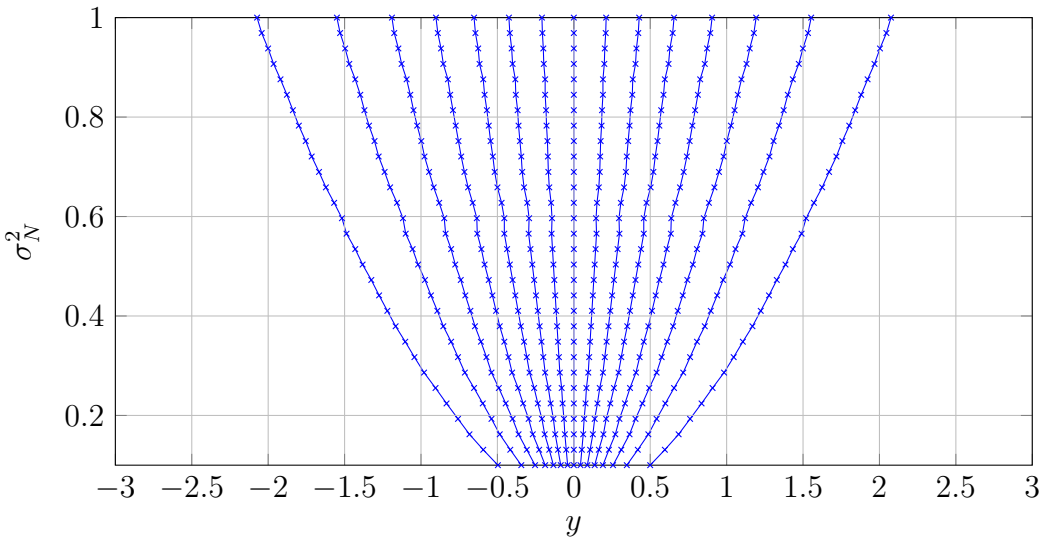


Figure 6.1: Cluster limits obtained by the symmetric sequential Information Bottleneck algorithm for different channel noise variances σ_N^2 and $|\mathcal{T}_c| = 16$.

6.2 Discrete Density Evolution

In the previous section, an information-optimal quantizer was derived. The next challenge when constructing a discrete receiver is to design an LDPC decoder, which solely works on a discrete output alphabet. Consequently, one major task is to replace the complex node operations introduced in Chapter 4 by more feasible and even simpler ones. Groundbreaking work in this field was done by Brian M. Kurkoski *et al.* in 2008. In their publication "Noise thresholds for discrete LDPC decoding mappings" [KYK08] the authors proposed a framework that combines density evolution and clustering algorithms.

The fundamental idea of this approach is to propagate compressed but highly informative messages along the edges of a Tanner graph. An important continuation of Kurkoski's findings was proposed by Lewandowsky in [LB15] introducing the Information Bottleneck method as clustering technique. As mentioned in Chapter 2 the Information Bottleneck method has exactly the required focal aim of preserving the mutual information about a relevant random variable when compressing the observation. Lewandowsky was able to show, that extending the ideas of Kurkoski *et al.* by applying the Information Bottleneck method allows constructing a discrete LDPC decoder with persuasive performance [LB15].

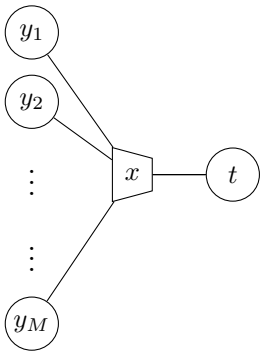


Figure 6.2: This Information Bottleneck graph visualizes the simultaneous processing of all M incoming messages at a variable or check node.

The overall design principle of discrete LDPC decoders can be easily visualized using Information Bottleneck graphs. Figure 6.2 visualizes the processing of M incoming discrete messages $\mathbf{y} = [y_1, \ldots, y_M]^{\mathrm{T}}$.

The number of processed messages M can take three different values. Firstly $M = d_c - 1$ for a check node, secondly $M = d_v$ for a variable node during the message passing and finally $M = d_v + 1$ when computing the output of a variable node considering all incoming messages as well as the channel message. The output is a realization of the compression variable T, i.e. $t \in \mathcal{T}$ and is determined by the mapping $p(t|\mathbf{y})$, which is designed using the Information Bottleneck method. The relevant variable x depends on the node type. For a variable node, x represents the underlying bit of a particular node. Whereas, if the clustering is designed for a check node, x represents the (mod 2)-sum of the possibly different bits $b_1, \ldots, b_m$.

To find the clustering as well as to mathematically define the relevant variable for both node types likewise, the vector $\mathbf{b} = [b_1, \ldots, b_m]^{\mathrm{T}}$ is introduced. The vector $\mathbf{b}$ contains all M bits represented by the messages in an input vector $\mathbf{y}$. Thus, the relevant variable for a variable node can be found as

$$x = b_1 = \text{const.} \tag{6.2}$$

and for a check node as

$$x = \sum_{l=1}^{M} \oplus b_l. \tag{6.3}$$

From a theoretical perspective, it would be possible to generate a clustering of $\mathbf{y}$ using the joint distribution $p(x, \mathbf{y})$ and the Information Bottleneck method. However, in practice, this is strictly not possible, since the vector $\mathbf{y}$ with M entries, all taken from the discrete alphabet $\mathcal{T}$, can take up to

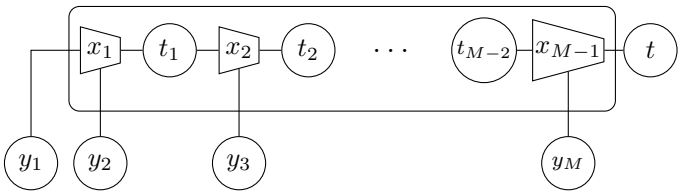


Figure 6.3: To reduce the amount of input combinations instead of compressing all incoming message simultaneously, the problem is decomposed in several partial node operations.

$|\mathcal{T}|^M$ distinct combinations. Thus, implementing the mapping $p(t|\mathbf{y})$ would have exponential space complexity in the number of input messages, which prohibits a direct, practical implementation.

Tackling this drawback by decomposition of the overall compression into several partial operations was also introduced by Kurkoski [KYK08]. The equations (6.2) and (6.3) can be rewritten as a serial concatenation of only two inputs, where one input depends on the result of the previous operation. This decomposition is described mathematically in great detail in [LSB16b]. However, once again it is easier to the understand the approach in a graphical manner, i.e. in terms of Information Bottleneck graphs. Figure 6.3 illustrates the Information Bottleneck graph for such a so-called *opened* node.

After being split into several partial operations, the original IB node, indicated by a black box in Figure 6.3, contains $M-2$ intermediate results and has still one output t. In contrast to the closed node, a relevant variable at each so-called *stage* x_l is introduced. According to (6.2) and (6.3), the intermediate relevant variable x_l for a variable node is denoted by [LSB16b]

$$x_l = b_1 = \text{const} \tag{6.4}$$

and x_l for a check node is

$$x_l = \sum_{m=1}^{l+1} \oplus b_m, \tag{6.5}$$

which is basically a running (mod 2)-sum. Especially (6.5) allows for an interesting interpretation because it points out that the intermediate results push forward only relevant information resulting in a so-called *flow* of relevant information.

In a next step the remaining joint distributions of the partial node operations have to be derived. These distributions are necessary to design message mappings for all intermediate stages using the Information Bottleneck method. At first a length two vector $\mathbf{y}_1 = [y_1, y_2]^{\mathrm{T}}$ for $l = 1$ is introduced.

For a variable node, independent of the decoder iteration, the first entry in this vector, i.e. y_1, holds the quantized channel output message for this particular variable node by definition [LB15]. The input vector for subsequent partial operations, i.e. $l > 1$, equals $\mathbf{y}_l = [t_{l-1}, y_{l+1}]^{\mathrm{T}}$. Consequently, the missing distribution follows as $p(x_l, \mathbf{y}_l)\ \forall l = 1, 2, \ldots, M-1$ and can be obtained for a check node and step $l = 1$ by marginalization

$$\begin{aligned} p(x_1, \mathbf{y}_1) &= \sum_{b_1} \sum_{b_2} p(x_1, b_1, b_2, \mathbf{y}_1) \\ &= \sum_{b_1} \sum_{b_2} p(x_1, b_1, b_2, y_1, y_2) \\ &= \sum_{b_1} \sum_{b_2} p(x_1|b_1, b_2) p(b_1, b_2, y_1, y_2) \\ &= \sum_{b_1} \sum_{b_2} \underbrace{p(x_1|b_1, b_2)}_{\delta(b_1 \oplus b_2 \oplus x_1)} p(y_1|b_1) p(y_2|b_2) \underbrace{p(b_1|b_2)}_{p(b_1)} p(b_2) \\ &= \sum_{\substack{(b_1,b_2): \\ b_1 \oplus b_2 = x_1}} p(y_1, b_1) p(y_2, b_2) \end{aligned} \tag{6.6}$$

and likewise for a variable node and step $l = 1$

$$\begin{aligned} p(x_1, \mathbf{y}_1) &= \sum_{b_1} \sum_{b_2} p(x_1, b_1, b_2, \mathbf{y}_1) \\ &= \sum_{b_1} \sum_{b_2} p(x_1, b_1, b_2, y_1, y_2) \\ &= \sum_{b_1} \sum_{b_2} p(x_1|b_1, b_2) p(b_1, b_2, y_1, y_2) \\ &= \sum_{b_1} \sum_{b_2} \underbrace{p(x_1|b_1, b_2)}_{\delta(b_1 - x)\delta(b_2 - x)} p(y_1|b_1) p(y_2|b_2) \underbrace{p(b_1|b_2)}_{\delta(b_1 - b_2)} p(b_2) \\ &= \sum_{\substack{(b_1,b_2): \\ b_1 = b_2 = x_1}} p(y_1|b_1) p(y_2, b_2). \end{aligned} \tag{6.7}$$

Similarly, this procedure can be applied for $l > 1$. Using (6.6) or (6.7) as input for the Information Bottleneck method, one obtains a clustering $p(t_l|\mathbf{y}_l)$ which maps a discrete vector $\mathbf{y}_l$ onto a discrete intermediate result t_l. As described earlier one input of the next step is a previous output, thus the output distribution of an intermediate stage $p(x_1, t_1)$ has to be forwarded during the design period. Exemplary, for $l = 1$

$$p(x_1, t_1) = \sum_{\mathbf{y}_1 \in \mathcal{Y}^{\mathrm{vec}}} p(t_1|\mathbf{y}_1) p(x_1, \mathbf{y}_1), \tag{6.8}$$

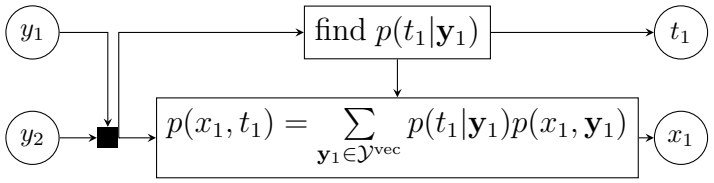


Figure 6.4: This figure shows the generation of the compression variable t_1 schematically, as well as the computation of the joint distribution $p(x_1, t_1)$.

where $\mathcal{Y}^{\text{vec}}$ denotes the set of all possible combinations of $\mathbf{y}_1$. Equation (6.8) only holds for the first partial node operation, but can be generalized for any $\mathbf{y}_l$ easily.

All steps performed at a particular intermediate IB node are schematically illustrated and summarized in Figure 6.4.

It was shown in [LSB16b] that the calculation of $p(x_1, \mathbf{y}_1)$ is basically an application of the sum-product algorithm to compute marginal posteriors. As explained in Chapter 3 all incoming messages at a variable node need to be multiplied to compute the marginals. However, this would result in $p(x_1, \mathbf{y})$, i.e. the joint probability also containing information obtained during later operations. As indicated in (6.6) and (6.7) as well as Figure 6.3 the relevant information is only passed forward. Thus, the problem is related to filtering without smoothing, also called forward propagation [Bis07].

Finally, it should be emphasized again, that the output distribution $p(x, t)$ of an outgoing message is always generated for one specific edge of the Tanner graph. If $p(x, t)$, i.e. the distribution of the transmitted message and the bit represented by this message, is exchanged between variable and check node iteratively, this process is referred to as discrete density evolution. Please note the connection to classical density evolution where the evolution of distributions was tracked as well.

The next section will summarize the major findings of this section and analyze the newly designed fully discrete LDPC decoder.

6.3 Generating a Discrete LDPC Decoder

In the previous section discrete density evolution was derived. It was shown how to find joint distribution serving as input for the Information Bottleneck method to obtain several distinct, deterministic clusterings or mappings which can be understood as look-up tables. In the first part of this section the resulting complexity reduction caused by decomposing the original setting is analyzed. As explained earlier, in the first iteration the cardinality of

Table 6.1: Number of entries in a look-up table without opening the node.

	$l = 1$	$l > 1$
variable node	$\lvert\mathcal{T}_c\rvert \cdot \lvert\mathcal{T}\rvert^{d_v-1}$	$\lvert\mathcal{T}_c\rvert \cdot \lvert\mathcal{T}\rvert^{d_v-1}$
check node	$\lvert\mathcal{T}_c\rvert^{d_c-1}$	$\lvert\mathcal{T}\rvert^{d_c-1}$

Table 6.2: Number of entries in a look-up table for one iteration, a regular (3,6)-LDPC code and $|\mathcal{T}_c| = |\mathcal{T}| = 16$.

	variable node	check node
closed node	$\lvert\mathcal{T}\rvert^{d_v} = 16^3 = 4096$	$\lvert\mathcal{T}\rvert^{d_c-1} = 16^5 = 1048576$
opened node	$(d_v - 1)\lvert\mathcal{T}\rvert^2 = 2 \cdot 16^2 = 512$	$(d_c - 2)\lvert\mathcal{T}\rvert^2 = 4 \cdot 16^2 = 1024$

a check node input message is $|\mathcal{T}_c|$. During the next iterations all incoming messages will have cardinality $|\mathcal{T}|$, which is also the cardinality of a check and variable node output. Hence, at a check node $|\mathcal{T}_c|^{d_c-1}$ input combinations exist during the first iteration and $|\mathcal{T}|^{d_c-1}$ input combinations for all other iterations. During iterative message passing, the variable nodes always process a channel message from the channel quantizer with cardinality $|\mathcal{T}_c|$ and $d_v - 1$ incoming messages from the check nodes with cardinality $|\mathcal{T}|$. This is summarized in Table 6.1.

For simplicity in the following example $|\mathcal{T}_c| = |\mathcal{T}|$. When opening the node, the increase in space complexity is only linear in the number of input messages. Thus for a check node $d_c - 2$ look-up tables are necessary each with $|\mathcal{T}|^2$ entries. Similarly, for a variable node $d_v - 1$ look-up tables exist also with $|\mathcal{T}|^2$ entries each. Thus, for $|\mathcal{T}| = 16$ the memory demand, i.e. the number of entries in a look-up table, for one iteration and a regular (3,6)-LDPC code is determined and summarized in Table 6.2. Obviously, the complexity reduction is enormous. Although, the overall complexity is reduced finding the look-up tables online during decoding is not practical. Hence, all tables are generated once for a so-called design-E_b/N_0. Afterwards, the decoder is used mismatched. The next chapter focuses on performance analyses of such a mismatched discrete decoder. However, it turned out that choosing the design-E_b/N_0 value close to the original threshold-E_b/N_0 found by density evolution is beneficial [KYK08], [LB15]. To find this threshold-E_b/N_0 the mutual information between all $d_v + 1$ messages at a variable node and the underlying code bit needs to be tracked for each iteration. This is sketched in Figure 6.5 exemplarily. Once a feasible design-E_b/N_0 was found

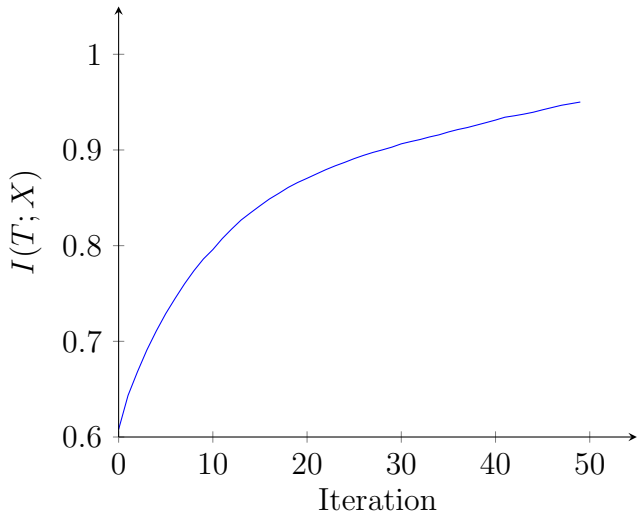


Figure 6.5: Exemplary sketch of increase in mutual information during discrete density evolution.

the generated look-up tables can be stored and used for decoding.

Decoding of LDPC codes using this new fully discrete technique will be the main focus of the remaining chapters.

Chapter 7

Information Optimum Decoding of Regular LDPC Codes

In [LB15] bit error simulation results for a regular LDPC code using a discrete LDPC decoder working only on integers were presented. At first, to verify the own implementation developed for this thesis, a regular LDPC code from the public MacKay-code-database [Mac] was selected. In this chapter, the obtained results will be compared with the performance of a belief propagation decoder and a min-sum decoder. Before analyzing the bit error rate performance, the simulation environment is described as well as the programming language.

7.1 Simulation Environment

7.1.1 Programming Language

Python 3.5 was chosen as programming language for this thesis. Furthermore, OpenCL, a programming language for heterogeneous programming was used to enable simulations on GPUs and CPUs resulting in a significantly increased simulation speed.

7.1.2 HPC Cluster Computing

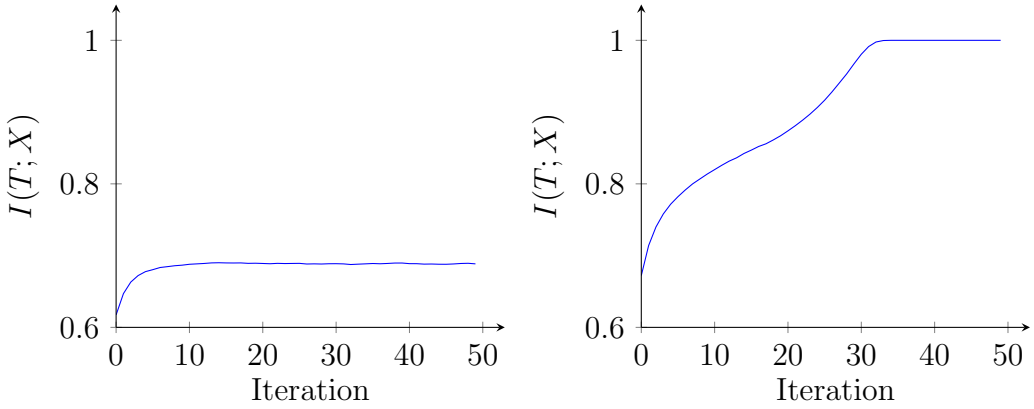
The bit error simulations ran on an HPC cluster provided by the TUHH. A cluster bundles a lot of powerful hardware. Processing time on this cluster can be requested using a workload manager. OpenCL allows working on

every computation node provided by the cluster, no matter if the node is a CPU or GPU. A list of available cluster hardware can be found at `https://www.tuhh.de/rzt/tuinfo/ausorg/hpc/hardware/`.

7.1.3 Implementation

Prior to investigating the performance of discrete LDPC decoders, all units forming a valid communication chain had to be implemented. The following Python modules can be found on the attached CD

- Sequential Information Bottleneck algorithm
- Modified sequential Information Bottleneck algorithm
- Symmetric sequential Information Bottleneck algorithm
- Linearized input symmetric sequential Information Bottleneck algorithm
- Discrete density evolution
- Min-sum LDPC decoder
- Belief propagation decoder
- LDPC encoder
- Discrete LDPC decoder
- Information optimum quantizer
- BI-AWGN channel
- PyOpenCL implementations of all decoder types
- Environments for decoder generation and bit error rate simulation
- etc.



(a) $E_b/N_0 = 0.85$ dB (b) $E_b/N_0 = 1.35$ dB

Figure 7.1: Evolution of mutual information for different design-E_b/N_0 values and a regular (3,6) LDPC code.

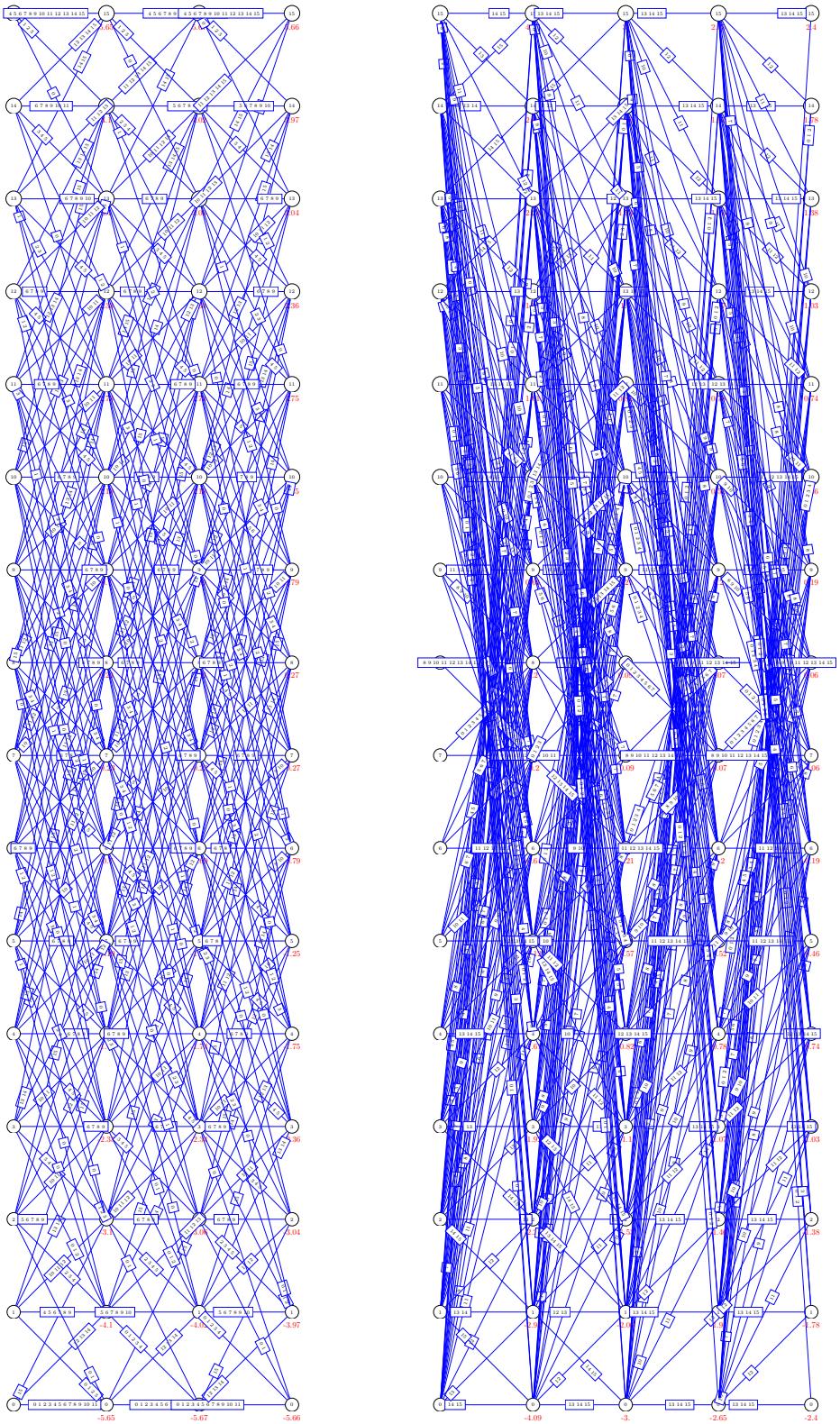
7.2 Analysis of Generated Discrete Decoders for Regular LDPC codes

The code used from the MacKay database is labeled 8000.4000.3.483. The codeword length is $N = 8000$ and the code rate $R = 0.5$. The variable node degree is $d_v = 3$ and the check node degree equals $d_c = 6$.

The first step is to generate a decoder. As explained in the previous chapter, the most important design parameter is the so-called design-E_b/N_0. The decoder is only generated for this specific E_b/N_0 and will not be updated during the simulation. When simulating bit error rates for different E_b/N_0-values only the channel quantizer is updated. Thus, all look-up tables required for decoding only need to be generated once. Furthermore, this can be done offline which is a major advantage.

One possibility to analyze a generated decoder is the mutual information curve as shown in Figure 7.1. If the E_b/N_0 used for the decoder generation is chosen too low, the mutual information between the variable node decision variables and the bit they represent saturates at a certain level and does not increase even for a large number of iterations (cf. Figure 7.1a). However, as soon as the design-E_b/N_0 passes the threshold-E_b/N_0, the mutual information approaches 1 for a huge number of iterations. This behavior impacts the BER curves and will be investigated in the next section.

In Appendix A more mutual information curves are shown for different design-E_b/N_0-values.



(a) Variable node iteration = 0 (b) Check node iteration = 0

Figure 7.2: Trellis diagram displaying the 16 possible output mappings corresponding to an input vector $\mathbf{y}_l$ at each partial node operation for both node types and $|\mathcal{T}| = 16$.

Another possibility to analyze a discrete LDPC decoder is to investigate the resulting message mappings. As mentioned in earlier chapters, two input messages combined in the vector $\mathbf{y}_l$ are mapped to a compression value t_l. This can be visualized using a trellis diagram as shown in Figure 7.2. These plots are generated automatically using a Python script which was developed during this thesis. Each trellis segment represents the transition from one current partial node operation to the next one. For example, the input vector $\mathbf{y}_0 = [15, 1]^{\mathrm{T}}$ would result in an output $t_0 = 14$ which is automatically the first input of the second partial node operation. It can be seen that due to the construction of the mappings, namely by using the symmetric sequential Information Bottleneck algorithm, the trellis itself is also symmetric. The states in the upper half correspond to positive LLRs, i.e. $\log\left(\frac{\Pr(X=+1|t)}{\Pr(X=-1|t)}\right) > 0, \forall t > |\mathcal{T}|/2$. Consequently, the states in the lower half correspond to negative LLRs. This observation is quite useful when performing hard decisions, because $t > |\mathcal{T}|/2$ corresponds to a transmit bit $u = 0$ and $t < |\mathcal{T}|/2$ corresponds to $u = 1$. The LLRs expressed by a cluster are also included in Figure 7.2 as red labels below the states.

7.3 Results for Regular LDPC codes

In the previous section different generated decoders were analyzed. The next step is to perform bit error rate simulations. As explained earlier a regular code with rate $R = 0.5$ was chosen.

The number of clusters, i.e. $|\mathcal{T}|$, was set to $|\mathcal{T}| = 16$, i.e. only 4 bits are needed to express all possible states. In general, it is possible to choose a channel quantizer $|\mathcal{T}_c|$ cardinality different from $|\mathcal{T}|$. Nevertheless, the simulation results were generated using $|\mathcal{T}_c| = |\mathcal{T}| = 16$, i.e. the quantizer was also limited to 4 bit, which is a quite coarse quantization, for example, compared to double precision with 64 bits. Using $|\mathcal{T}_c| = |\mathcal{T}|$ offers the advantage that storing messages from the channel and the messages passed during decoding requires the same amount of bits in the hardware.

Despite the short bit width, the results shown in Figure 7.3 are quite impressive in comparison to the included performance of a belief propagation decoder working with double precision. Although only 16 different integers were processed by the discrete decoder, the performance degradation in terms of bit error rate is only 0.1dB. The operations performed during discrete decoding are only simple look-ups in the priorly generated look-up tables, whereas the belief propagation reference simulation processed double precision channel output values and also performed double precision box-plus operations. The green curve in Figure 7.3 shows a fair reference because the

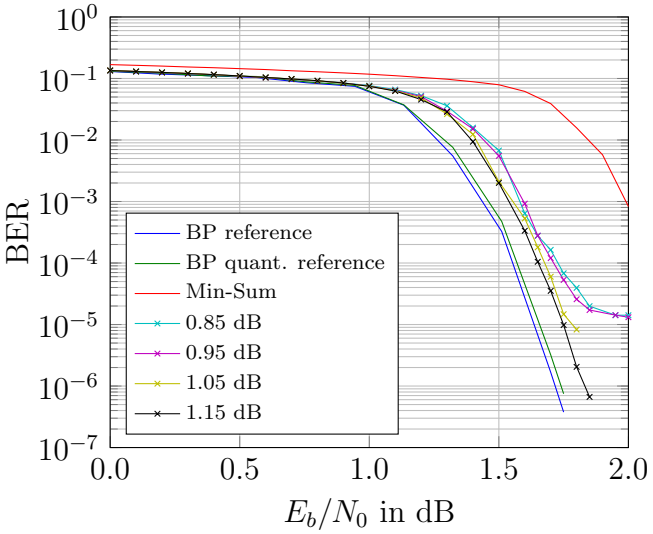


Figure 7.3: BER curves of different discrete decoders compared to state of the art decoders like belief propagation, min-sum decoder or belief propagation with quantized input values. The curve label corresponds to the design-E_b/N_0 of the discrete decoders.

belief propagation decoder only obtained 16 distinct input LLRs from the channel output quantizer. As a consequence, both decoders, discrete and quantized belief propagation, suffered from the same information loss due to channel output quantization. However, even quantized LLRs processed by belief propagation are double precision floating points and the operations during message passing are still computationally expensive box-plus operations. The alternative state-of-the-art decoding technique to achieve simplified node operations is min-sum decoding. Although the min-sum decoder can exchange double precision LLRs, the performance is very poor compared to the discrete, information based decoding technique.

Some further insides about the discrete decoders can be gained from the simulation results shown in Figure 7.3. It can be seen that the evolution of mutual information, observed during the decoder generation, as visualized in Figure 7.1, has a direct influence on the error floor of the discrete decoder. If the design-E_b/N_0 for the decoder generation is set to low, the bit error rate does not decrease further even for high signal-to-noise ratios. As shown in Figure 7.1a this corresponds to a saturated mutual information evolution. The decoder underestimates the certainty of the received samples and thus high reliabilities cannot propagate through the graph during message passing. If the design-E_b/N_0 is chosen high enough, i.e. the mutual

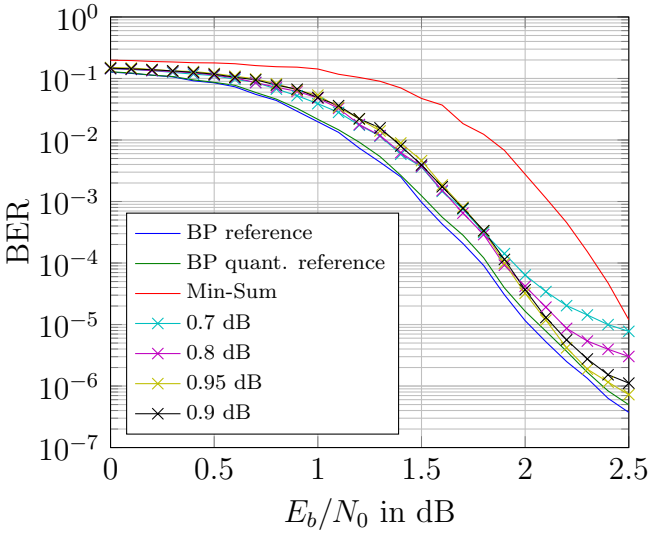


Figure 7.4: Applying discrete decoders to irregular codes results in an early error floor even if high E_b/N_0-values are used for the generator construction.

information slowly approaches 1, no uncertainty remains and the early error floor can be overcome. As shown in Figure 7.3 the error floor of the discrete decoders decreases with increasing E_b/N_0 used for generation, until it is not ascertainable in the investigated bit error rate region.

7.4 Simple Approach to Irregular LDPC Codes

In previous chapters, it was outlined that irregular LDPC codes approach the theoretical channel capacity much closer than regular codes can [CFRU01]. Hence, applying discrete decoders also to this kind of codes is highly desirable.

For higher practical relevance of the results and a possible comparison to other scientific publications it is beneficial to use codes from an official standard. As a first approach the implementation of the previously described discrete decoder generation was applied to an irregular LDPC code from the IEEE 802.11 WLAN standard. The code is quite short with a length of 1296 bits. The selected rate is $R = 0.5$ with the following node-degree distributions

$$\mathbf{d}_c = \left[0, 0, 0, 0, 0, 0, \frac{10}{12}, \frac{2}{12}\right]^{\mathrm{T}} \tag{7.1}$$

$$\mathbf{d}_v = \left[0, \frac{11}{24}, \frac{9}{24}, \frac{1}{24}, 0, 0, 0, 0, 0, 0, \frac{3}{24}\right]^{\mathrm{T}}. \tag{7.2}$$

From an implementation perspective no major changes need to be performed to apply the present decoder generation technique to irregular codes. It is sufficient to generate look-up tables for the highest node degree and take the appropriate intermediate results as output for lower node degrees. All distributions are weighted accordingly with respect to the edge-degree distribution of the irregular code, as described in (5.17) and (5.19), during discrete density evolution.

The resulting decoders are used for bit error simulations. The corresponding bit error rate curves are shown in Figure 7.4 for several decoders generated for different E_b/N_0-values. Obviously, the chosen code is not very strong, i.e. the waterfall region is not very steep, which is caused by the short length. However, even for this quite weak code the discrete decoder is not able to follow the reference curve into low bit error regions. Although the E_b/N_0-value used for generation is quite high, an error floor cannot be avoided. Of course, it would be possible to increase the E_b/N_0-value for decoder construction further and further but this as a negative influence on the performance in higher bit error regions. As visualized in Figure 7.4 the design-E_b/N_0 for one particular decoder moves to higher E_b/N_0's when the E_b/N_0-value used for decoder generation is increased. Consequently, there seems to exist a crucial problem which occurs when applying discrete decoders generated with the available discrete density evolution scheme to irregular codes. In the next chapters, this problem is analyzed deeply and solutions will be developed and evaluated to overcome the discussed problems.

Chapter 8

Message Alignment

Discrete decoders build a new powerful class of LDPC decoders combining low complexity and high performance. However, the previous chapter already outlined that discrete decoders constructed using the available discrete density evolution technique are not as versatile as other techniques like belief propagation or min-sum decoding.

This chapter will investigate why the implementation used for regular LDPC codes cannot be applied to irregular codes directly. The findings will not be limited to irregular LDPC codes or decoding in general, but will also help to understand the flow of relevant information in distributed tasks.

Since the addressed problem is very sophisticated, it is beneficial to start with a simple example in the first section. Afterwards, the problem will be formalized and a new technique called *message alignment* will be derived.

8.1 Consensus of Meanings

The Information Bottleneck solely focuses on maintaining relevant information about a certain random quantity. To understand the problem which occurs when trying to apply the discrete evolution scheme to irregular codes it is important to recap two quantities from previous chapters, namely belief and uncertainty. The mutual information helps to reduce the uncertainty of a variable X when knowing Y. The belief itself is a probability mass function that allows guessing the value of X when knowing Y. This belief is often expressed using LLRs if X is binary. The belief is important for detection and estimation tasks. When compressing the observation in order to keep all relevant information the belief is not anymore expressed by $L(x|y) = \frac{\Pr(X=+1|y)}{\Pr(X=-1|y)}$ but $L(x|t) = \frac{\Pr(X=+1|t)}{\Pr(X=-1|t)}$, i.e. each cluster has a certain meaning with respect to X. The conditional probability $p(x|t)$ expresses a certain meaning for X

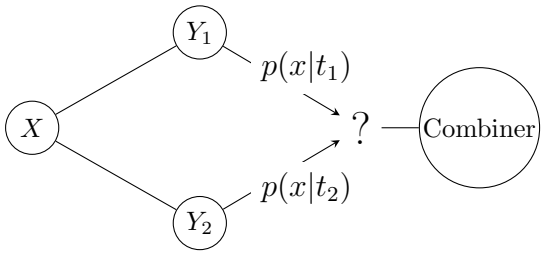
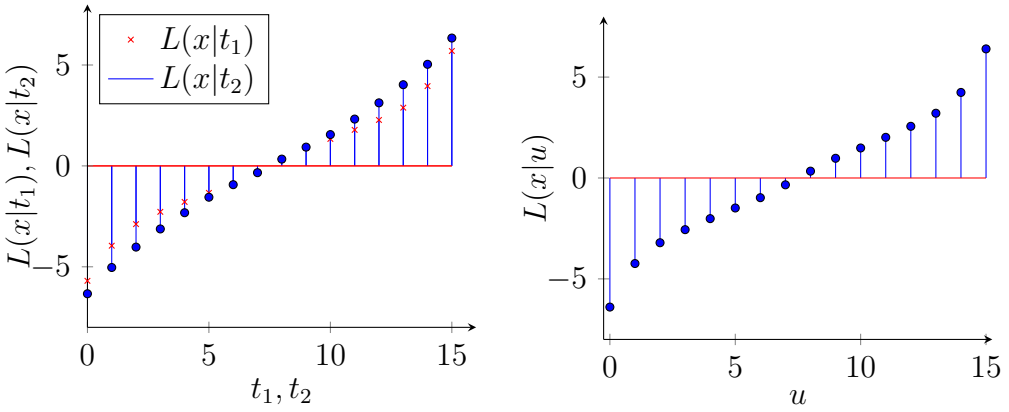


Figure 8.1: Two distributed nodes measure the same quantity and forward their compressed belief, where due to limited number of clusters, a cluster with the same cluster index can have different meanings.

for a given t. However, the meaning of a cluster can change depending on the input distribution $p(x, y)$. Assume a setting as shown in Figure 8.1. Two nodes gather independent observations Y_0 and Y_1 of a relevant variable X, in a possibly distributed system.

For instance, consider two students who have written the same exam. Afterwards, they are asked by an evaluation team of the university to grade the exam with the following options $\mathcal{O} = \{\text{easy}, \ldots, \text{not sure}, \ldots, \text{brutal}\}$. Each of these options is only a subset of the possible student's feelings which are in reality more versatile. Hence, the limited amount of answers equals a compression. For computer based analysis the evaluation team enumerates the possible outcomes in $\mathcal{O}$, i.e. $\mathcal{T} = \{0, 1, \ldots, |\mathcal{T}| - 1\}$. In the previously used notation the enumerated events correspond to realizations of the compression variable T. Furthermore, there exists a relation between a cluster $t \in \mathcal{T}$ and a corresponding meaning $o \in \mathcal{O}$. To simplify the notation $p(x|o) = p(x|t)$, but t corresponds to the enumerated outcomes in $\mathcal{T}$. The relevant variable X in this example is the *true* difficulty of the exam which can either be hard or easy, i.e. X is binary. The validity of the poll is limited if the interviewed students are from two different experience levels, e.g. experienced and beginner. If the old student, who has written many exams, says it was hard it will have a different meaning or reliability as if a student writes its first exam and finds it hard.

The evaluation team does not know which student is more experienced and gives more reliable results, they just know that there are two groups of students $\mathcal{S} = \{\text{beginner}, \text{experienced}\}$ and a student is experienced with $\Pr(S = \text{experienced})$ and a beginner with $\Pr(S = \text{beginner})$. Moreover, the students are asked to tell the evaluation team the corresponding reliabilities for each cluster in terms of LLRs and the corresponding conditional probabilities denoted as $p(x|t_{\text{beginner}})$ or $p(x|t_{\text{experienced}})$. The crucial task of the evaluation team is to ascertain the real difficulty of the exam without



(a) (b)

Figure 8.2: The left figure shows the different meanings of a cluster for the two considered nodes from Figure 8.1 exemplary. The right graph shows the naive combination of $p(x|t_1)$ and $p(x|t_2)$ resulting in a different belief.

knowing who out of the test set gave the feedback.

The random variable U represents the outcome of the poll, i.e. the combined clusters, with the event space $\mathcal{U} = \mathcal{T}$ and its corresponding probability mass function $p(u)$. A particular outcome of the poll $u \in \mathcal{U}$ can either be caused by an experienced student or a beginner. Thus the distribution $p(u)$ can be written as

$$p(u) = \sum_{s \in \mathcal{S}} p(t_s)p(s). \tag{8.1}$$

Just as $p(u)$, also the joint distribution $p(x, u)$ is found by averaging the two distributions $p(x, t_{\text{beginner}})$ and $p(x, t_{\text{experienced}})$. Applying Bayes' rule yields

$$p(x|U = u) = \frac{\sum_{s \in \mathcal{S}} p(x, T_s = u)p(s)}{\sum_{x \in \mathcal{X}} \sum_{s \in \mathcal{S}} p(x, T_s = u)p(s)} = \frac{p(x, U = u)}{\Pr(U = u)}, \tag{8.2}$$

to obtain an averaged cluster meaning.

Figure 8.2a shows the conditional probabilities for two different students and the resulting distribution $p(x|u)$ the team will use for their decisions in Figure 8.2b as LLRs.

For notational convenience the lengthy but intuitive labels t_{beginner} and $t_{\text{experienced}}$ are replaced by t_1 and t_2 in the following.

Obviously, no matter if experienced student or beginner, in both cases the initially intended meaning of their grading, i.e. expressed by $L(x_1|t_1)$

and $L(x_2|t_2)$, will be changed significantly. The major occurring problem is that the evaluation team has no information about the experience level of the particular student who gave a particular test evaluation. The problem could also appear if the two nodes in Figure 8.1 are not students but sensor nodes measuring the same quantity or nodes in a Tanner graph with varying degree.

Considering an irregular LDPC code the different node degrees can generate different reliabilities with respect to the same underlying bit. In this context, the evaluation team is the other node type, e.g. a check node, which receives the probability distribution from, e.g., a variable node during density evolution. Due to the random structure of the graph it is not possible to track the degrees of all sending nodes for each receiving node. Hence, it is only possible to describe the composition in a probabilistic manner using the edge-degree distribution.

In the previous chapter discrete density evolution for regular codes was applied to irregular codes directly, with the only difference that (5.17) and (5.19) were used to determine an output distribution which is forwarded to another node input. Comparing (5.17), or respectively (5.19), with (8.2) underlines similarities between the simple example from above and density evolution. The random relation between cluster indexes and meanings causes a significant change in meaning when averaging over all possible degrees weighted by the edge-degree in density evolution. Therefore, it is not possible to propagate the intended reliabilities through a randomly structured graph without *aligned* meanings of messages. Especially variable nodes with high degrees suffer from this problem because there typically high LLRs, are attenuated causing an early error floor as seen in the previous chapter. Thus, the central aim is to pair the newly developed message alignment with discrete density evolution in order to construct discrete LDPC decoders for irregular codes.

The labeling of the clusters is arbitrary and can be changed easily. In contrast to the original observation, where each $y \in \mathcal{Y}$ had at least a physical meaning this is not true for the cluster labels. Hence, it is possible to relabel the clusters in order to align the resulting meaning of the independent compressions. From an information geometrical perspective, the involved conditional distributions $p(x|t_1)$ and $p(x|t_2)$, which are carrying the information about the relevant variable X, are points on an $S_{|\mathcal{X}|-1}$ simplex. In case of a binary variable X, as for the considered binary LDPC code, the resulting S_1 simplex is just a line. Consequently, $p(x|t_1)$ and $p(x|t_2)$ are two points on that line spaced by a certain distance. The further the two points diverge the larger the resulting information loss or Kullback-Leibler divergence. This affects the misinterpretation in the belief of the relevant random variable X.

Of course, when considering a simple shape like an S_1 simplex it would be sufficient to measure the distance between the points or its scalar counterparts, i.e. LLRs, using the euclidean distance. However, for relevant random variable with non-binary event space this is not necessarily true. Instead, the Kullback-Leibler divergence is chosen as more generic distortion measure.

The next sections will focus on the mathematical formulation of the previously described problem as well as developing iterative algorithms to minimize the information loss caused by inappropriate averaging in the context of discrete density evolution.

8.2 Mathematical Problem Formulation

Summarizing the previous sections, one concludes that combining clusters, denoting diverging meanings or *beliefs*, with respect to the relevant variable results in an undesirable loss in inference performance. The idea developed during this thesis is to introduce a relabeling of the clusters, which can also be seen as deterministic transformation

$$p(z_1|t_1) = \delta(z_1 - z_1^*(t_1)) \tag{8.3}$$

where T_1 is the original compression variable and Z_1 is the newly introduced aligned compression variable.

For simplicity only two different clustering results shall be combined. A second application of the Information Bottleneck method creates another clustering T_2 with respect to the same relevant variable X. For the sake of fairness, it is valid to argue that also this clustering should be aligned against T_1, but this more complicate case will be investigated in later sections.

Consequently, for a given $t_1 \in \mathcal{T}$, solely the deterministic mapping $p(z_1|t_1)$ needs to be found such that

$$p(x|Z_1 = z_1) \approx p(x|Z_2 = z_2), \text{ if } z_1 = z_2 \tag{8.4}$$

The distortion measure used to align the message meaning as good as possible is the Kullback-Leibler divergence. The Kullback-Leibler divergence is not symmetric, raising the question, in which order to place the arguments. In [M+05] common divergences in information theory were investigated in detail. It turns out, that the reference distribution should be placed as second argument to force the first distribution to lock on the dominant mode of the second distribution. This is desirable for the message alignment and thus the sought mapping $p(z_1|t_1)$ should be designed such that

$$\forall(t_1, z_1): \quad D_{KL}\left\{p(x|T_1 = t_1)||p(x|Z_2 = z_1)\right\} \longrightarrow \min. \tag{8.5}$$

It is not possible to derive a closed form solution due to the structure of the setting. This is a known and common problem [M+05]. Thus, it is recommended in [M+05] and [WLW09] to develop iterative algorithms finding stationary points.

A first design approach to find a suitable $p(z_1|t_1)$ for a given $t_1 \in \mathcal{T}$ is

$$z_1^*(t_1) = \arg\min_{z_1} D_{KL}\left\{p(x|T_1 = t_1)||p(x|Z_2 = z_1)\right\}, \forall t_1. \tag{8.6}$$

In Chapter 9 it will be shown that already this simple algorithm yields significant performance gains.

8.3 Quantifying the Mismatch Loss in Discrete Density Evolution.

Message alignment can be interpreted as a Kullback-Leibler divergence minimization problem. However, minimizing this expression in closed form is often not possible. The previously introduced iterative algorithm is limited to two distinct posterior distributions $p(x|t_1)$ and $p(x|t_2)$, which should be aligned, i.e. $p(x|z_1) \approx p(x|t_2)$ for $t_2 = z_1$. When considering irregular LDPC codes, usually, more than two distinct node degrees exist. In this more complex setting, it is not obvious which belief to use as reference. Hence, another concept has to be defined, extending the original minimization, namely the so called mismatch loss. This quantity measures the remaining information mismatch after the alignment procedure.

After the message alignment, the resulting joint distribution $p(x, u)$ is obtained, which is used for density evolution at the other node type as

$$p(x, U = u) = \sum_{s\in\mathcal{S}} p(x|Z_s = u)\Pr(Z_s = u)p(s), \tag{8.7}$$

where $p(s)$ is the corresponding probability mass function of the random variable S. The random variable S represents the possible node degrees and their corresponding probabilities are determined by the edge-degree distribution as in the previous sections. By marginalization, $p(u)$ can be found as

$$\begin{aligned} p(u) &= \sum_{x\in\mathcal{X}}\sum_{s\in\mathcal{S}} p(x|Z_s = u)\Pr(Z_s = u)p(s) && (8.8)\\ &= \sum_{s\in\mathcal{S}} \Pr(Z_s = u)p(s)\sum_{x\in\mathcal{X}} p(x|Z_s = u) && (8.9)\\ &= \sum_{s\in\mathcal{S}} \Pr(Z_s = u)p(s). && (8.10) \end{aligned}$$

The resulting conditional probability equals

$$p(x|U=u) = \frac{\sum\limits_{s\in\mathcal{S}} p(x|Z_s=u)\Pr(Z_s=u)p(s)}{\Pr(U=u)}, \tag{8.11}$$

which expresses the meaning of the joint clusters with respect to the relevant variable X after the alignment step. Please note that $p(x|z_s)$ already depends on the corresponding transformation mappings $p(z_s|t_s)$ for the particular node degrees.

The joint distribution $p(x,u)$ is forwarded as input to the other node type since it corresponds to the joint distribution of the represented bits x and the aligned messages z_s.

The costs caused by the remaining mismatch can be determined using the Kullback-Leibler divergence $D_{KL}\{p(x|z_s)||p(x|u)\}$ for $z_s = u$. Please note that for simplicity from now on the "*" in (8.6) is dropped, i.e. Z_s represents the optimally transformed version of T_s. For a certain node degree s which outputs the aligned messages z_s the "local" expected costs are

$$\mathcal{L}_{\text{local}} = \sum_{z_s\in\mathcal{Z}_s} p(z_s) D_{KL}\{p(x|z_s)||p(x|U=z_s)\}, \tag{8.12}$$

where "local" indicates that the costs belong to one specific node degree s. These costs describe the change in initially intended meaning due to the remaining mismatch. If more than two nodes are involved the local costs are weighted to obtain the global mismatch loss. The corresponding weighting factors for each local clustering loss are given by $p(s)$, i.e. the edge-degree distribution. Consequently, the overall expected mismatch loss is

$$\mathcal{L}_{\text{global}} = \sum_{s\in\mathcal{S}} p(s) \sum_{z_s\in\mathcal{Z}_s} p(z_s) D_{KL}\{p(x|z_s)||p(x|U=z_s)\}, \tag{8.13}$$

where $\mathcal{L}_{\text{global}}$ is a function of all determined deterministic mappings $\delta(z_s - z_s^*(t_s)), \forall s$.

Only when considering this cost function it is possible to make reliable statements about different investigated alignment techniques.

Applying the initial minimization strategy would result in finding optimal mappings $p(z_s|t_s)$ for every node, except the one chosen as reference node z_{ref} which was seen as most useful and reliable.

One of the main advantages of the Information Bottleneck method was the absence of a predefined distortion measure. For this alignment strategy a certain measure to select a reference node is required. The left bar chart in

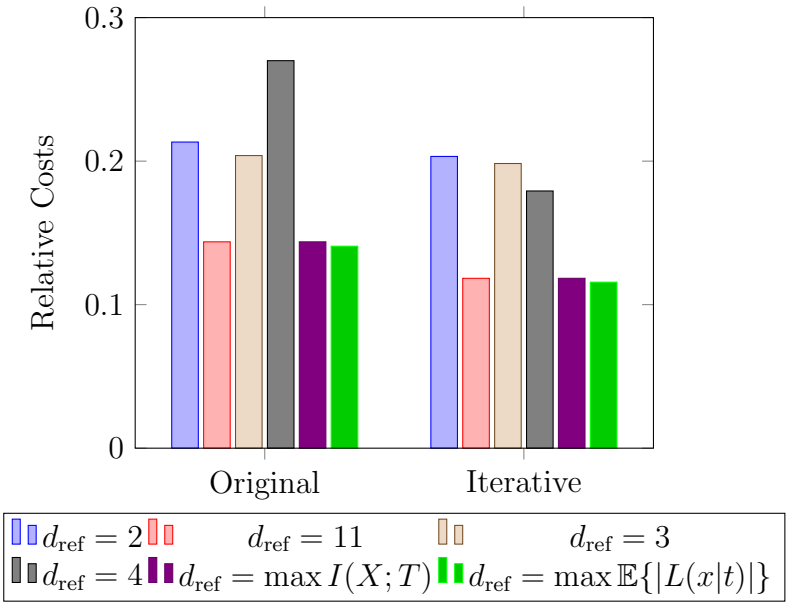


Figure 8.3: Comparison of relative costs for different initial reference nodes for two different optimization techniques. Lower relative costs translate directly to less message meaning divergence. Furthermore, d_{ref} denotes the node degree of the reference node.

Figure 8.3 shows a comparison of all possible reference nodes for an irregular LDPC code from the WLAN standard which will be used in later chapters and their corresponding relative cost reduction compared to the case without message alignment. Using (8.13) the relative costs can be found as

$$\text{relative costs} = \frac{\mathcal{L}_{\text{global, aligned}}}{\mathcal{L}_{\text{global, not aligned}}}. \tag{8.14}$$

Consequently, in Figure 8.3 a relative cost of 1 would correspond to no gain compared to the case without message alignment. The shown result is an example for the first decoder iteration and a check node output messages. It turns out that no matter which node is chosen as reference the costs are reduced at least by a factor of 4. The best performance is achieved if the node-degree with the highest average LLR magnitude is chosen. A more detailed comparison with the second best option, namely choosing the node with the highest mutual information, reveals that there is no big difference between these two approaches. From a practical perspective, it is preferable to choose the node with the highest mutual information. Since when considering the communication overhead generated by the alignment, the variable node with the highest degree also contains the highest amount of mutual

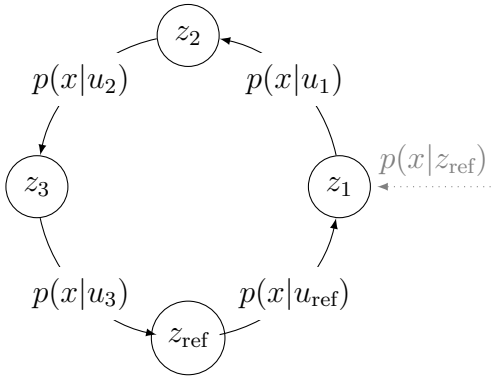


Figure 8.4: Finding a consensus on the best message alignment iteratively.

information. Hence, the reference node is fixed in advance and does not need to be determined for each iteration by exchanging average LLR magnitudes between the nodes.

The optimization criterion defined in (8.5) assumes only two nodes with different cluster meanings. In this section, this approach was extended by defining and evaluating the mismatch loss. However, in contrast to (8.6) where the messages were always aligned with respect to the same reference node, the combined meaning $p(x|u)$ is used to compute the mismatch loss. This is beneficial because, if a matching is always performed against the initial cluster meanings of the reference node, the already changed distribution after the previous partial alignments is not considered. Neglecting this would result in an alignment that has been optimized for the static reference $p(x|z_{\text{ref}})$ and a certain node $p(x|z_s)$ but is not necessarily optimal for the intermediate matching result $p(x|u)$ and $p(x|z_s)$, since in general $p(x|z_{\text{ref}}) \neq p(x|u)$.

Trying to minimize (8.13) directly is not trivial. A common technique to minimize such problems is to use iterative algorithms. The invented algorithm is schematically shown in Figure 8.4. A reference node is chosen according to the same criterion as in the previous algorithm. The corresponding cluster meanings are denoted as $p(x|z_{\text{ref}})$. The algorithms used to align two nodes is applied to determine the resulting message alignment between reference node and another randomly chosen node. The combined conditional probability function $p(x|u_1)$ is computed using the normalized weights of the two nodes, defined by the edge-degree distribution. Instead of using again $p(x|z_{\text{ref}})$ as input distribution in the next alignment step, $p(x|u_1)$ is used. This approach is beneficial because the knowledge that the original message meanings of the reference node $p(x|z_{\text{ref}})$ are already adapted is not ignored anymore. This pairwise aligning and combining is repeated several

times until all nodes are included in the alignment procedure.

The last step in the first iteration of one matching cycle is to match the reference node against the current result, i.e all previous pairwise alignments are considered.

The advantage of this second iterative technique is apparent from the right bar chart in Figure 8.3. The iterative approach reduces the relative costs even further. Especially the node degree suited worst as reference node in the original setting reduces its costs significantly. However, choosing the node degree with the highest mutual information or the highest average LLR magnitudes still results in the largest gains. Selecting the node degree with the highest mutual information together with the iterative algorithm results in 10-times lower global alignment costs compared to the case without alignment. This is an obvious performance gain.

Furthermore, the adjustment of the originally developed minimization algorithm leads to 20% less relative costs compared to the original technique.

In the next chapter, these algorithms will be used to generate discrete decoders for irregular LDPC codes. The effect of reduced costs is directly related to the BER performance, which will be presented in the next chapter.

8.4 Message Alignment in Information Bottleneck Graphs

In the previous section, message alignment was presented as new technique to enlarge the number of possible applications of information optimum design. It turned out that the optimal relabeling of cluster indices helps to harmonize the different meanings of originally similar clusters. Another important step is to include this technique in the framework of Information Bottleneck graphs, which made it possible to visualize the flow of relevant information in a complex system. In this section, a new shape representing the message alignment procedure will be introduced. Figure 8.5 shows this new shape.

In the well-known factor graph notation a factor is a filled square. The newly created message alignment node contains a circle inside this square with two important characteristics, namely the color and the label. The label $x|.$ indicates that the alignment is for compression variables carrying information or meanings about the relevant variable X. The color of the circle is a second important property because it helps to understand which variables participate in the alignment procedure. As explained in the previous section it is possible that more than two variables are aligned using an iterative algorithm. In a graph, it would be too confusing to draw solid, black lines

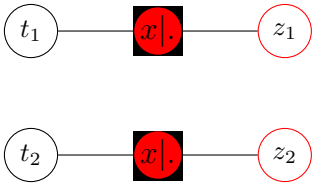


Figure 8.5: The new message alignment node is characterized by its label and the color. Every variable node that is involved in the alignment is colored the same way as the message alignment node itself. The label indicates the relevant variable.

indicating links between factor and variable nodes, if many variables are involved in the alignment. Hence, instead of a solid connection using a line, the link is visualized using colors. If variables are found by message alignment their nodes get the same color as the circle inside the message alignment node. In the next section, this new IB node will be used to visualize a discrete decoder for irregular LDPC codes.

8.5 Discrete Decoder Design for Irregular LDPC Codes

In the previous sections of this chapter, a new technique called message alignment was introduced as well as a new IB node. In this section the major findings will be recapped and applied to a simple toy example. As stated earlier, the main motivation of this thesis and the development of message alignment is to enable discrete decoding of irregular LDPC codes. A simple irregular LDPC code ensemble is given by the following edge-degree distribution

$$\lambda(x) = \frac{2}{3}x^2 + \frac{1}{6}x^3 + \frac{1}{6}x^5 \tag{8.15}$$

$$\rho(x) = \frac{1}{6}x^3 + \frac{5}{6}x^4, \tag{8.16}$$

i.e. distinct check node degrees are $d_c = 4$ and $d_c = 5$ and the existing variable node degrees are $d_v = 3$, $d_v = 4$ and $d_v = 6$.

In Figure 8.6 one iteration of the discrete decoder is illustrated using an Information Bottleneck graph. Since the focus in this Information Bottleneck graph is on message alignment, the IB nodes are closed although they will be opened in practice as described in Chapter 6. Furthermore, the notation from (6.4) and (6.5) in Chapter 6 is used. Thus, a relevant variable x_l

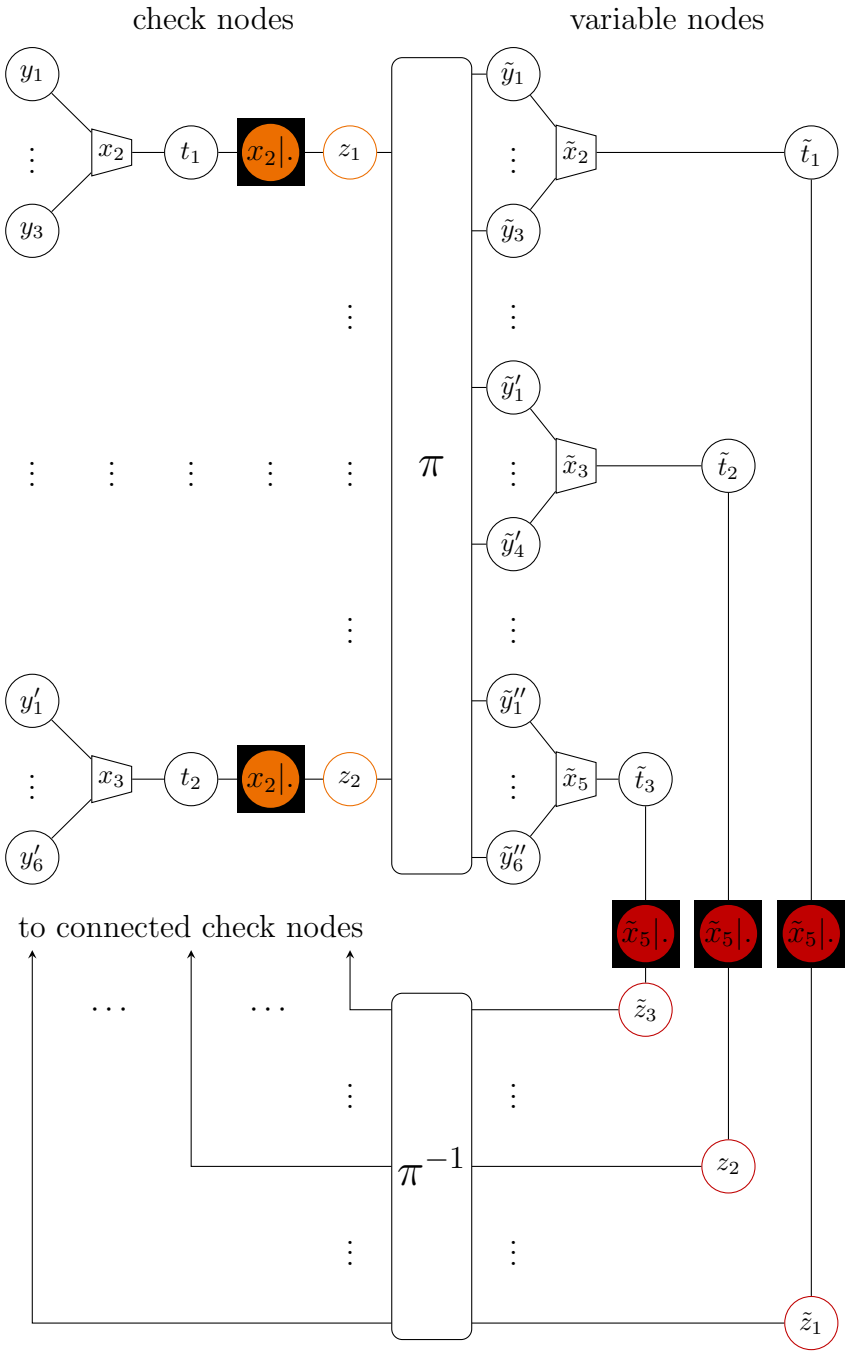


Figure 8.6: Discrete message passing in discrete irregular LDPC decoders using message alignment. One decoder iteration is shown starting at the check nodes.

means that the discrete message t is an output of a check node with degree $d_c = l + 2$ or a variable node with degree $d_v = l + 1$. The mapping which determines this outgoing message is designed for one particular edge. Thus, for each edge there exists a look-up table, which is similar if the node degree of the sending node is the same. This structure offers high parallelism of the decoding algorithm.

The illustrated discrete message passing iteration in Figure 8.6 starts at the check nodes in the top left corner. According to (8.16) two check node degrees exist. In the previous section, it was highlighted that the initial reference node in message alignment should be the one with the highest mutual information $I(X;T)$ or the highest average LLRs. For check nodes, the node with the lowest degree is most informative. Thus, the variable node with degree $d_c = 4$ is defined as initial reference. In Figure 8.6 this is indicated by the label $x_2|$. in the message alignment nodes on the left side (cf. message alignment node with filled, orange circle). The two connected variable nodes z_1 and z_2 are aligned as the color of the node implies. Since only two distinct node degrees exist, algorithm (8.6) is used to obtain the mapping $p(z_2|t_2)$ and $p(z_1|t_1)$ is just an identity mapping. The joint distribution $p(x, u)$, defined in (8.7), is forwarded to the input of the variables nodes during discrete density evolution. Too many pseudo-random connections between variable and check nodes to be tracked exist, hence the Tanner graph is modeled stochastically which is illustrated by an interleaver π in Figure 8.6. Please note that in the actual decoding realizations of Z_1 and Z_2 are sent to the variable nodes. The distribution $p(x, u)$ is found by substituting the appropriate values from (8.16) in (8.7)

$$p(x, U = u) = \frac{1}{6}p(x|Z_1 = u)p(Z_1 = u) + \frac{5}{6}p(x|Z_2 = u)p(Z_2 = u). \quad (8.17)$$

Please note that the event space $\mathcal{U}$ equals the original compression space $\mathcal{T}$. In a next step the variable nodes design independent mappings, i.e. $p(\tilde{t}_1|\tilde{\mathbf{y}})$, $p(\tilde{t}_2|\tilde{\mathbf{y}}')$ and $p(\tilde{t}_3|\tilde{\mathbf{y}}'')$, using the distributions received from the check nodes and the channel. Since three different variable node degrees exist, the iterative message alignment algorithm illustrated in Figure 8.4 is applied. At first the node with degree $d_v = 6$ is chosen as initial reference because the variable node with the highest degree is most informative. Consequently, after the alignment between reference and variable node with degree $d_v = 4$ the first intermediate result $p(x|u_1)$ is determined. By considering the weights defined in the edge-degree distribution and normalizing them, $p(x|u_1)$ equals

$$p(x|u_1) = \frac{p(x, u_1)}{\sum\limits_{x \in \mathcal{X}} p(x, u_1)}, \quad (8.18)$$

where

$$p(x, U_1 = u_1) = \frac{\frac{1}{6}p(x, Z_3 = u_1) + \frac{1}{6}p(x, Z_2 = u_1)}{\frac{1}{6} + \frac{1}{6}}. \tag{8.19}$$

In the next step $p(x|u_1)$ and $p(x|t_1)$, i.e. the variable node with the lowest degree, are aligned. The aligned output variable is denoted by U_2. According to (8.6) and when putting $p(x|u_1)$ as reference, the minimization problem follows as:

$$z_1^*(t_1) = \arg\min_{z_1} D_{KL}\left\{p(x|T_1 = t_1)||p(x|U_1 = z_1)\right\}, \forall t_1. \tag{8.20}$$

Similar to the first step, the intermediate distribution $p(x|u_2)$ equals

$$p(x|u_2) = \frac{p(x, u_2)}{\sum\limits_{x \in \mathcal{X}} p(x, u_2)}, \tag{8.21}$$

where

$$p(x, U_2 = u_2) = \frac{\frac{1}{6}p(x, Z_3 = u_2) + \frac{1}{6}p(x, Z_2 = u_2) + \frac{2}{3}p(x, Z_1 = u_2)}{\frac{1}{6} + \frac{1}{6} + \frac{2}{3}}. \tag{8.22}$$

Finally, it is possible align the mappings of the initial reference node and the latest intermediate result. At first the mapping $p(z_3|t_3)$ is found

$$z_3^*(t_3) = \arg\min_{z_3} D_{KL}\left\{p(x|T_3 = t_3)||p(x|U_2 = z_2)\right\}, \forall t_3. \tag{8.23}$$

Afterwards, intermediate result and newly designed mapping $p(z_3|t_3)$ are combined to obtain

$$p(x|\tilde{u}) = \frac{p(x, \tilde{u})}{\sum\limits_{x \in \mathcal{X}} p(x, \tilde{u})}, \tag{8.24}$$

where

$$p(x, \tilde{U} = \tilde{u}) = \frac{\frac{1}{6}p(x, Z_3 = \tilde{u}) + \frac{1}{6}p(x, Z_2 = \tilde{u}) + \frac{2}{3}p(x, Z_1 = \tilde{u})}{\frac{1}{6} + \frac{1}{6} + \frac{2}{3}}. \tag{8.25}$$

As shown in Figure 8.6 $p(x, \tilde{u})$ is passed back to connected check nodes over the random graph π^{-1} during discrete density evolution, whereas during decoding $\tilde{z}_1, \tilde{z}_2$ and $\tilde{z}_3$ are sent.

Chapter 9

Information Optimum Decoding of Irregular LDPC Codes

In previous chapters the superior performance of irregular LDPC codes was outlined. Furthermore, a new strategy to harmonize meanings of clusters received from different node degrees, which represent beliefs with respect to the same relevant variable, was developed.

In this chapter, the new technique, called *message alignment*, will be applied when constructing discrete decoders for irregular codes. The investigated irregular codes are selected from two different standards. In the first part, BER performance analysis is realized for a short irregular code from the IEEE 802.11 WLAN standard. Afterwards, BER performance analysis is performed for a much longer and thus stronger irregular code from the DVB-S2 standard. Analogously to Chapter 7, belief propagation decoding using continuous respectively quantized input LLRs as well as min-sum decoding are considered as reference systems.

9.1 Results and Discussion for irregular IEEE 802.11 WLAN Code

The WLAN standard is widely used if people want to establish a wireless Internet connection. The standard considers different LDPC codes for error correction with different codeword length [IEE12]. The selected code has length $N = 1296$ and the node-degree distribution from (7.1).

Figure 9.1 shows the resulting BER curves for reference decoders and two discrete decoders. Both discrete decoders are generated for one particular

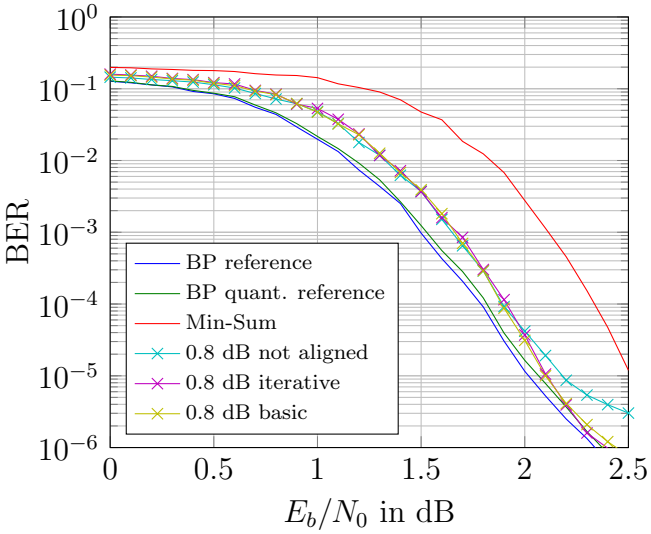


Figure 9.1: BER performance comparison using the reference decoders and discrete decoders generated for a design-$E_b/N_0 = 0.8$ dB and $|\mathcal{T}| = 16$.

design-$E_b/N_0 = 0.8$ dB. During the simulation, only the channel quantizer adapts to the current E_b/N_0. The magenta curve in Figure 9.1 corresponds to the discrete decoder which uses message alignment during the construction. Obviously, the undesirable error floor disappears, when message alignment is applied. Consequently, it can be concluded that the message alignment strategy is a promising approach. As reference, the cyan curve displays BER results for a discrete decoder without the alignment strategy. Moreover, Figure 9.1 shows the improved performance due to the iterative message alignment (magenta curve), which was developed as an extension of the firstly proposed alignment technique (yellow curve). Performing the alignment pairwise and iteratively resulted in lower global costs, introduced as a quality measure in Chapter 8. Hence, the correlation between reduced global costs and improved error correction performance is evidently demonstrated. Simulation results for different design-E_b/N_0 emphasizing this observation are included in Appendix B for different examples. The performance degradation of the discrete decoder is only around 0.1 dB. However, it is possible to close the gap even further by increasing the number of clusters. As shown in Figure 9.2, when using 5 bit quantization, the discrete LDPC decoder does not suffer from any practically relevant loss in comparison to the belief propagation reference. Results for a variety of discrete decoders generated for different design-E_b/N_0 values and 32 clusters, i.e. 5 bits, are also included in Appendix B.

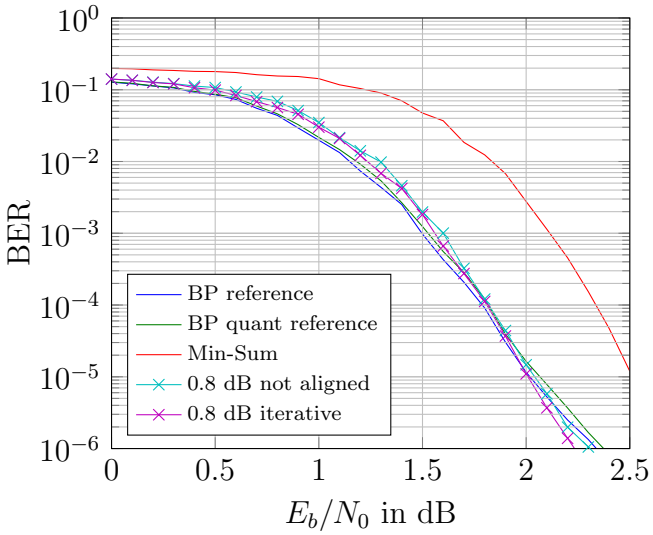


Figure 9.2: BER performance comparison using the reference decoders and discrete decoders generated for a design-$E_b/N_0 = 0.8$ dB and $|\mathcal{T}| = 32$.

9.2 Results and Discussion for DVB-S2 Code

The results obtained for the WLAN code indicated that message alignment is an expedient approach for the application of discrete decoders to irregular LDPC codes. However, for a couple of reasons it is deeply interesting to include an irregular LDCP code from the DVB-S2 standard in the investigation.

Firstly, the chosen code belongs to the common class of repeat-accumulate codes, which are easy to encode. Secondly, the code is longer, stronger and thus has a steep waterfall region and a low error floor. Being able to follow this steep curve would again prove that message alignment is able to suppress the early error floor encountered during the simple implementation in Chapter 7. Thirdly, applying the newly developed technique to another code verifies the validity of earlier results.

Figure 9.3 shows BER simulation results for the DVB-S2 code. The BER curves for the reference systems are again compared with the performance of a discrete decoder generated for a design-$E_b/N_0 = 0.8$ dB. In contrast to the results obtained without message alignment (light blue), it is possible to follow the belief propagation decoder curve with a gap of only 0.2 dB (magenta), when using the new technique. Furthermore, the performance improvement obtained by applying iterative message alignment outperforms the basic approach (yellow) remarkably.

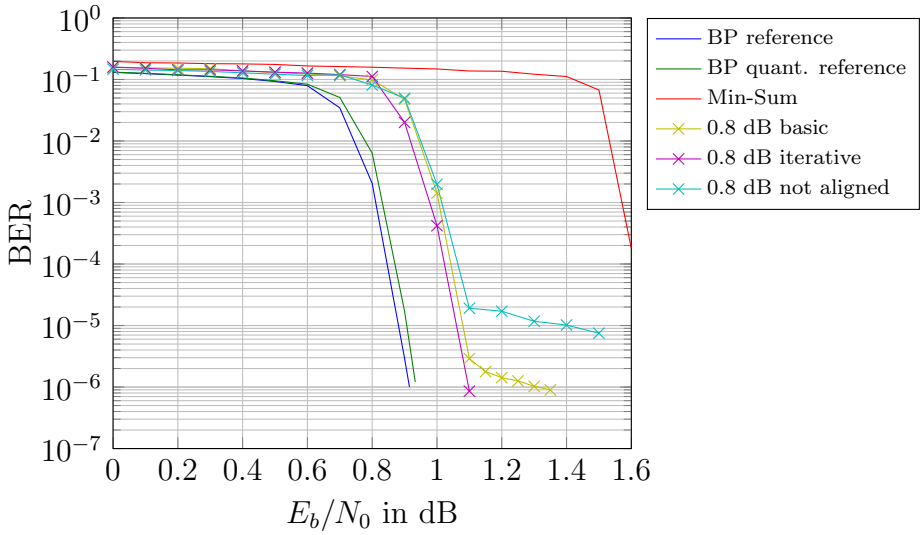


Figure 9.3: BER performance comparison using the reference decoders and discrete decoders generated for a design-$E_b/N_0 = 0.8$ dB and $|\mathcal{T}| = 16$.

The chosen code from the DVB-S2 standard [ETS14] has a codeword length of $N = 64800$ and the following node-degree distribution

$$\mathbf{d}_c = \left[0, 0, 0, 0, 0, 0, \frac{1}{32400}, \frac{32399}{32400}\right]^{\mathrm{T}} \tag{9.1}$$

$$\mathbf{d}_v = \left[1/64800, \frac{32999}{64800}, \frac{19440}{64800}, 0, 0, 0, 0, \frac{12960}{64800}\right]^{\mathrm{T}}. \tag{9.2}$$

Further BER simulations strengthening the observations are included in Appendix C. Like for the WLAN code, it is possible to close the gap by increasing $|\mathcal{T}|$. Figure 9.4 shows BER curves for the reference systems and a discrete decoder generated for a design-$E_b/N_0 = 0.8$ dB and 32 clusters. The remaining gap between fair belief propagation reference and discrete decoder after spending one more bit, i.e. 5 bits in total, is below 0.1 dB. Further simulation results for 32 clusters but different design-E_b/N_0 values are included in Appendix C.

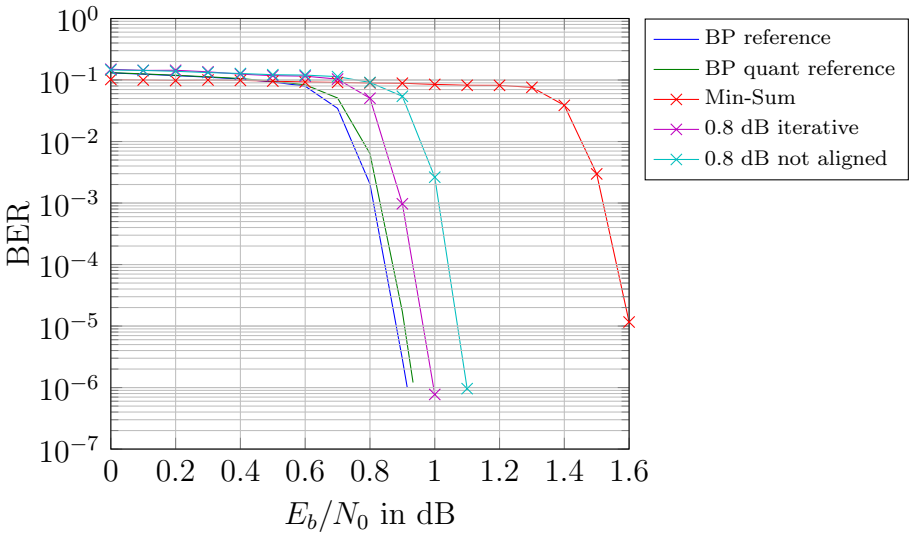


Figure 9.4: BER performance comparison using the reference decoders and discrete decoders generated for a design-$E_b/N_0 = 0.8$ dB and $|\mathcal{T}| = 32$.

Chapter 10

Performance Evaluation for Irregular LDPC Codes with Higher Order Modulation

Higher order modulation schemes are often used to increase the spectral efficiency [Kam13]. According to the standard, also the previously investigated irregular LDPC codes are paired with such modulation schemes [IEE12], [ETS14]. However, so far only information optimum quantizers for binary phase-shift keying (BPSK) modulations were proposed in literature [KYK08], [LB15], [LSB16a]. It turns out, that generalizing quantizer design poses quite similar challenges as the ones encountered during irregular LDPC decoder design.

In this chapter, the newly developed message alignment technique is applied to quantizer design for higher order modulation, to outline its universality. Therefore, in the first part quadrature amplitude modulation (QAM), which is used together with the WLAN code, is introduced. Afterwards, the quantizer design is explained and simulation results are given to compare the performance. In the second part, phase-shift keying (PSK) is investigated. 8-PSK is used in the DVB-S2 standard. Again, the quantizer design is explained and simulation results are given.

10.1 Quantizer Design for 64-QAM Modulation

QAM schemes spread constellation points in the complex plane according to a grid-like pattern. Depending on the number of symbols M, each symbol carries $\log_2 M$ bits. Thus, with each channel access the number bits transmitted

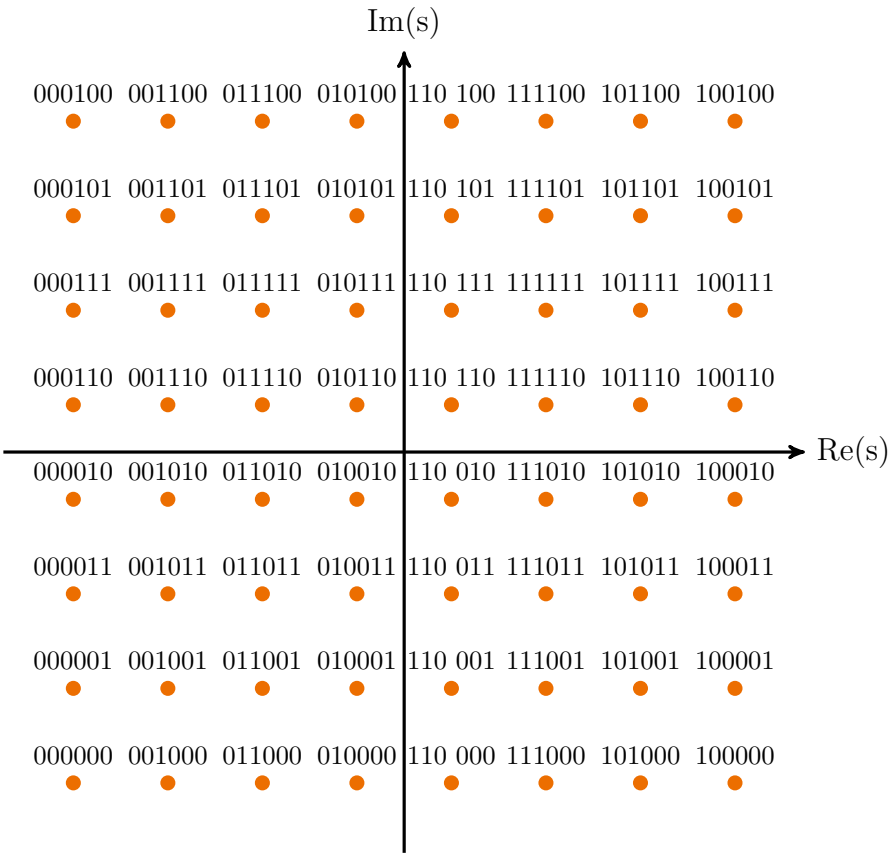


Figure 10.1: Constellation mapping of the investigated 64-QAM.

increases with M, resulting in a higher spectral efficiency. Simultaneously, the bit error probability increases due to the decreased distance, since more symbols need to be packed into the plane.

The considered 64-QAM is shown in Figure 10.1. Due to the arrangement, the real and imaginary part can be treated independently, resulting in two $\sqrt{M}-$amplitude-shift keying (ASK) modulations [Kam13]. Often, the bit mapping is chosen such that the bits for a certain amplitude are identical. Furthermore, the bit mappings of each $\sqrt{M}-$ASK modulation are identical allowing to reuse the quantizer designed for the real part also for the imaginary part. Consequently, one quantizer designed to discretize the received samples $\mathrm{Re}(\tilde{y})$ can also be applied to $\mathrm{Im}(\tilde{y})$.

Information optimum quantizer design always starts with a joint distribution between the relevant quantity, i.e. the transmitted symbol S, and the observation $\tilde{Y}$. For a $\sqrt{M}-$ASK modulation the corresponding joint

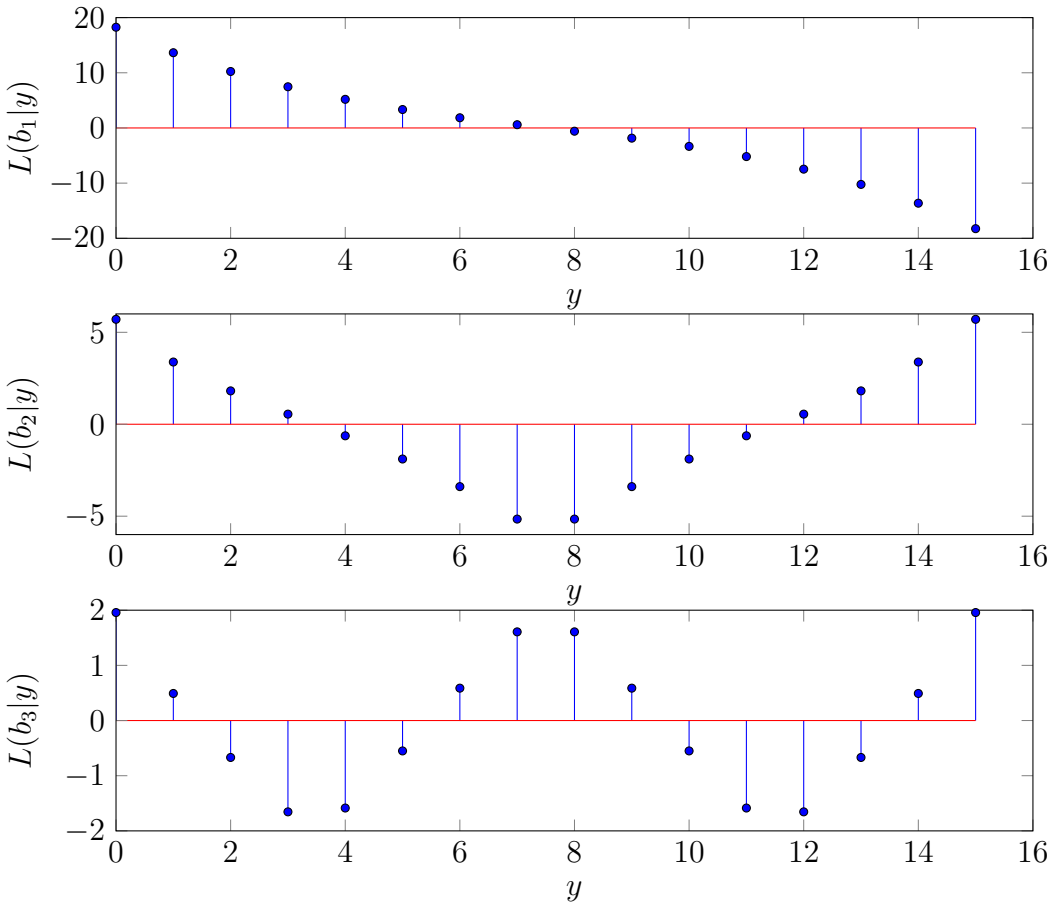


Figure 10.2: Meanings of the different clusters for the three bits of an 8-ASK constellation. Obviously, the meaning expressed by a cluster varies significantly for different bits.

distribution is given by

$$p(\tilde{y}, s) = p(\tilde{y}|s)p(s) = \frac{1}{\sqrt{2\pi\sigma_N^2/2}} \exp\left(-\frac{1}{2\sigma_N^2/2}(\tilde{y}-s)^2\right)\frac{1}{\sqrt{M}}. \quad (10.1)$$

In the first step of the design procedure $p(\tilde{y}, s)$ is used as Information Bottleneck input to determine $p(y, s)$, where the random variable Y describes the quantized version of $\tilde{Y}$.

The bit labeling defines the relation between a symbol and the $\log_2 M$ bits and can be written as conditional distribution $p(b_0, b_1, \cdots b_M|s)$. Using this relation the joint distribution for one arbitrary bit and the compressed observation can be derived as

$$p(b_M, y) = \sum_{s \in \mathcal{S}} p(y|s)p(b_M|s)p(s). \tag{10.2}$$

Figure 10.2 shows the meaning of each cluster for all three different bits of an 8-ASK modulation. The meanings expressed by a particular cluster differ significantly for different bits. However, the decoder can cope only with one particular distribution. Hence, it is necessary to align these meanings using message alignment. Due to its superior performance, the iterative message alignment algorithm is used for the quantizer generation.

10.2 Results and Discussion for IEEE 802.11 WLAN Code with 64-QAM

Figure 10.3 shows the bit error rate simulations for the same reference decoders as in previous chapters and the new discrete decoder. The belief propagation algorithm which takes only $|\mathcal{T}|$ distinct LLRs obtains its inputs from the newly designed 64-QAM quantizer. The loss caused by the information optimum quantizer for 64-QAM is only 0.2 dB which is slightly higher compared to results seen in previous chapters. However, in contrast to the binary case, i.e. BPSK, the relevant variable can take 8 values. Thus, the ratio $|\mathcal{T}|/|\mathcal{X}|$ is decreased by a factor of four and the performance degradation is still quite small. Consequently, applying message alignment to quantizer design for higher order modulation schemes turns out as a promising approach. The additional penalty which the discrete decoder suffers is still only 0.2 dB. Thus, in combination with the loss caused by the quantizer, the overall gap between continuous belief propagation and discrete decoding is 0.4 dB. Min-sum decoding is still performing worse than the discrete decoder although neither the input is quantized nor the word length is limited during processing.

10.3 Quantizer Design for 8-PSK Modulation

QAM is a very common modulation scheme, e.g., if the spectral efficiency shall be increased [Kam13]. However, due to the required fast amplitudes changes amplifiers need to fulfill demanding linearity constraints. If cheap, low-power amplifiers are used this linear region is often quite small. Hence, another modulation scheme is used which encodes information only in the phase instead of the amplitude. These schemes are called phase-shift keying (PSK). In PSK all constellation points are placed on a unit circle. Often,

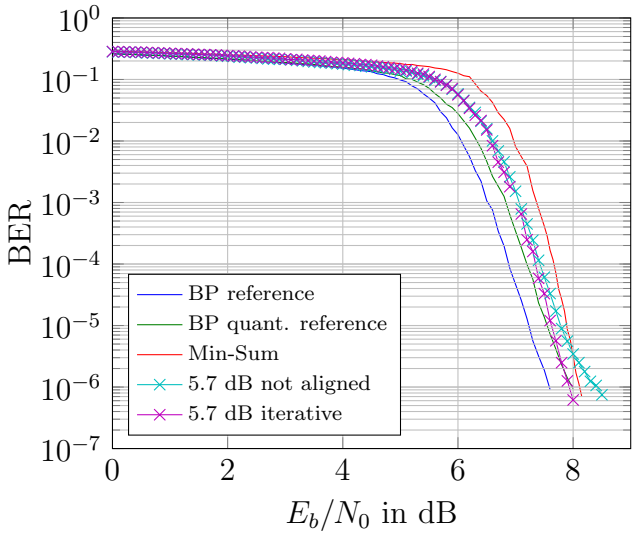


Figure 10.3: BER curve comparison using reference decoders and discrete decoders generated for a design-$E_b/N_0 = 5.7$ dB and $|\mathcal{T}| = 16$ for 64-QAM modulation and an irregular code form the WLAN standard.

communication satellites cannot effort expensive, energy demanding amplifiers. Thus, in the DVB-S2 standard LDPC decoders are paired with 8-PSK modulation. The corresponding constellation diagram is shown in Figure 10.4.

At first, to simplify the processing, the complex plane is transformed from Cartesian into polar coordinates using a bivariate transformation [Stü11]

$$p(\theta, r|s) = \frac{r}{2\pi\sigma_N^2} \exp\left\{-\frac{1}{2\sigma_N^2}\left(r^2 - 2\sqrt{2}r\cos(\theta - s) + 2\right)\right\}. \tag{10.3}$$

For computation of the bit error rate of PSK modulation only the phase is considered. However, to provide LLRs for soft decision decoding also the magnitude needs to be considered. Using the chain rule, the mutual information between all three quantities can be written as

$$I(S; \Theta, R) = I(S; \Theta) + I(S; R|\Theta). \tag{10.4}$$

Figure 10.5 visualizes the normalized loss of mutual information, i.e. $\frac{I(S;R|\Theta)}{I(S;R,\Theta)}$, when considering solely the phase. Interestingly, only up to 4% mutual information are lost and this amount decreases for higher E_b/N_0-values. Thus, the developed simple low complexity quantizer quantizes only the phase. The appropriate marginal distribution $p(\theta|s)$ can be found as

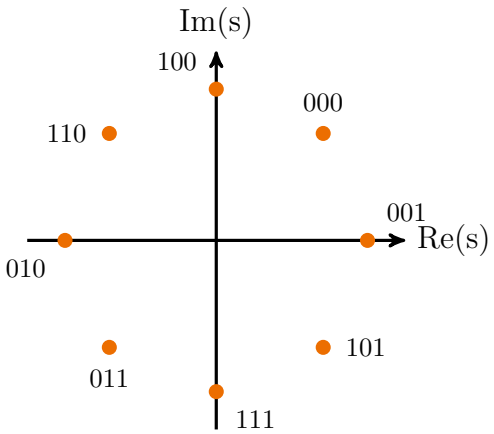


Figure 10.4: Constellation mapping of the investigated 8 PSK modulation.

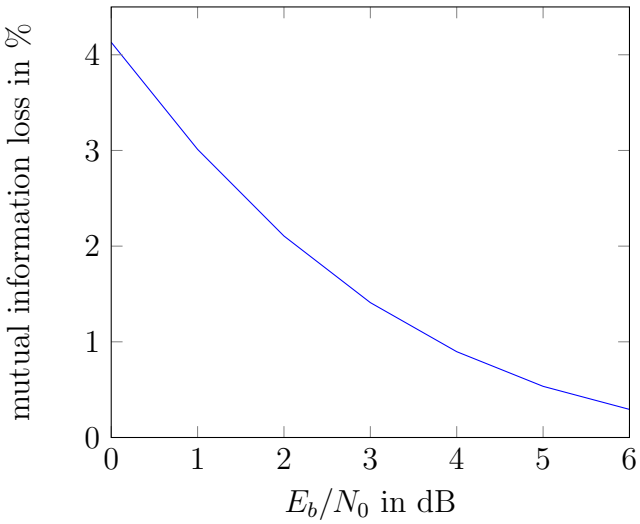


Figure 10.5: Relative loss of mutual information when considering only the phase.

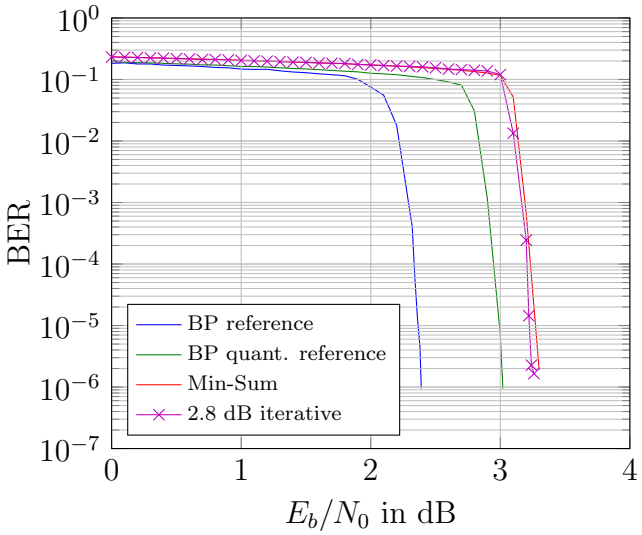


Figure 10.6: BER curve comparison using reference decoders and discrete decoders generated for a design-$E_b/N_0 = 2.8$ dB and $|\mathcal{T}| = 16$ for 8-PSK modulation and an irregular code form the DVB-S2 standard.

$$p(\theta|s) = \frac{1}{\pi} \exp\left\{-\frac{1}{\sigma_N^2} \sin^2(\theta - s)\right\} \cdot \tag{10.5}$$

$$\int_0^\infty x \exp\left\{(x - \sqrt{\frac{1}{\sigma_N^2}} \cos(\theta - s))^2\right\} dx. \tag{10.6}$$

Using this distribution allows generating an information optimum phase quantizer. The desired distributions for each bit can be obtained using the mapping, as shown in (10.2). Similarly to QAM quantization, the meanings expressed by the same cluster for different bits differs significantly. Thus, message alignment is used to harmonize the cluster meanings.

10.4 Results and Discussion for DVB-S2 Code with 8-PSK

Figure 10.6 and 10.7 display BER simulation results for the newly designed quantizer for $|\mathcal{T}| = 16$ and $|\mathcal{T}| = 32$ compared to reference systems.

The performance degradation of the discrete decoder is still only 0.2 dB compared to belief propagation with quantized LLRs. However, the performance loss caused by the quantizer is more severe than for 64-QAM or

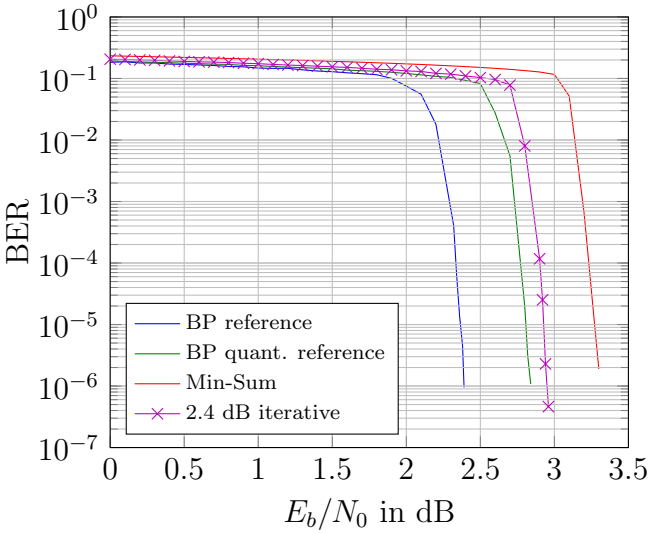


Figure 10.7: BER curve comparison using reference decoders and discrete decoders generated for a design-$E_b/N_0 = 2.4$ dB and $|\mathcal{T}| = 32$ for 8-PSK modulation and an irregular code form the DVB-S2 standard.

BPSK. For $|\mathcal{T}| = 16$ the quantizer adds a penalty of 0.6 dB and 0.4 dB for $|\mathcal{T}| = 32$. Nevertheless, in both cases the min-sum decoder is still outperformed, although with a smaller margin. Furthermore, it should be noted again that only the phase is considered which is a significant complexity reduction. Hence, one can summarize that considering only the phase together with message alignment for quantizer design in combination with discrete decoders allows decoding irregular LDPC codes with 8-PSK modulation better than min-sum decoding but in a less complex manner. Moreover, it is observed that increasing the number of phase quantization levels to $|\mathcal{T}| = 32$ allows to considerably outperform the min-sum decoder and decreases the gap between discrete decoding and belief propagation decoding.

Chapter 11

Conclusion

The central aim of this thesis was to broaden the applicability of discrete decoders to irregular LDPC decoders. In the first part existing discrete decoders for regular LDPC decoders were implemented and evaluated. Python was chosen as programming language making it necessary to implement existing Information Bottleneck algorithms as well as reference systems and receiver units.

Using bit error rate simulation, the competitive performance of discrete decoders was verified using a regular LDPC code from a public database.

It was shown that a straightforward application of existing design approach is not possible for irregular codes. It turned out, that due to the random structure of the graph, in combination with the significantly shrank cardinality of the exchanged variable, unresolved conflicts in intended meanings with respect to the relevant variable occurred. Since the belief expressed by equivalent cluster indices can differ, an intermediate step aligning the meaning of the message was introduced. Basically, message alignment can be interpreted as Kullback-Leibler divergence minimization problem. The Kullback-Leibler divergence helps to figure out which clusters generated by different Information Bottlenecks express similar meanings. According to this measure the labeling of the clusters is adapted to harmonize their meanings.

At first, an iterative algorithm was derived which aligned two different node degrees. However, most irregular codes are composed of more than two distinct degrees. Thus, as an extension of the first algorithm an iterative pairwise alignment technique was developed aiming to minimize the overall mismatch costs. It could be shown that this second approach results in an even further increased performance.

Extensive bit error simulations were conducted to compare the performance between the newly introduced discrete decoding construction technique and existing reference systems. Belief propagation decoders with and

without quantized channel LLRs as well as min-sum decoders served as reference systems.

In a first simulation an irregular code from the WLAN standard was chosen. The performance degradation of a discrete decoder was only 0.1-0.2 dB over the continuous belief propagation decoder, although the size of the message alphabet was limited to only 16 unsigned integers. Furthermore, the min-sum decoder was considerably outperformed. To validate the promising results of this first simulation a second code from DVB-S2 was chosen. This code was longer and also stronger. However, again the gap between continuous belief propagation and discrete decoder was only 0.2 dB. In contrast, the min-sum decoder lost around 0.7 dB.

This successful integration of message alignment in the irregular LDPC decoder construction motivated to identify new applications. It was discovered, that quantizers for higher order modulation schemes suffer from the same problems as irregular LDPC codes in terms of an existing message meaning conflict. Due to the nature of higher order modulation, distinct bits in a symbol are protected differently. The resulting distinct reliabilities cannot be recovered from the cluster indices directly. Thus, message alignment is needed. To increase the practical relevance of the findings of this thesis, appropriate modulation schemes for the investigated codes defined in the standards were considered. The WLAN standard pairs 64-QAM modulation and the investigated irregular LDPC code, whereas DVB-S2 combines 8-PSK modulation and irregular codes.

Bit error rate simulation validated that the application of message alignment to information optimum quantization under higher order modulation is beneficial. Especially the combination between WLAN code and 64-QAM modulation showed only minor performance degradation. During the quantizer design for 8-PSK, only the phase was quantized because it was shown that neglecting the magnitude causes a loss in mutual information of only less than 4 %. Although considering the phase of the received signal greatly reduces the receiver complexity, the achieved bit error rate performance was still very promising.

In conclusion, it can be summarized that message alignment increases the number of applications of the Information Bottleneck method in system design significantly. Especially the possibilities for distributed detection problems are an interesting topic for further work. Furthermore, the influence of the neglected magnitude when designing quantizers for 8-PSK could be investigated in more detail.

Irregular LDPC codes gain increasing importance for current and future communication systems. Despite the findings of this thesis are mostly theoretical, they could help to adapt the decoding to hardware constraints with

almost no practically relevant performance degradation. In addition, the provided generalization of information optimum quantizer design to higher order modulation schemes allows to include discrete decoders in already existing standards as well as future high-speed communication systems. Thus the findings of this thesis increase the practical applicability of discrete, information optimum LDPC decoders significantly.

Appendix

Appendix A

Additional Simulation Results for Regular LDPC Code from MacKay Database

The regular LDPC code was chosen from the MacKay database [Mac] and is labeled 8000.4000.3.483. The codeword length is $N = 8000$ and the node degrees are $d_v = 3$ and $d_c = 6$. This Appendix contains mutual information evolution curves for different E_b/N_0-values.

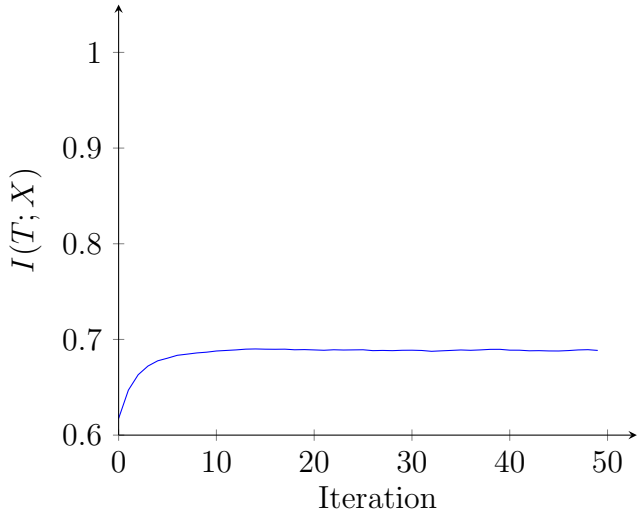


Figure A.1: Evolution of mutual information for design-$E_b/N_0 = 0.85$ dB

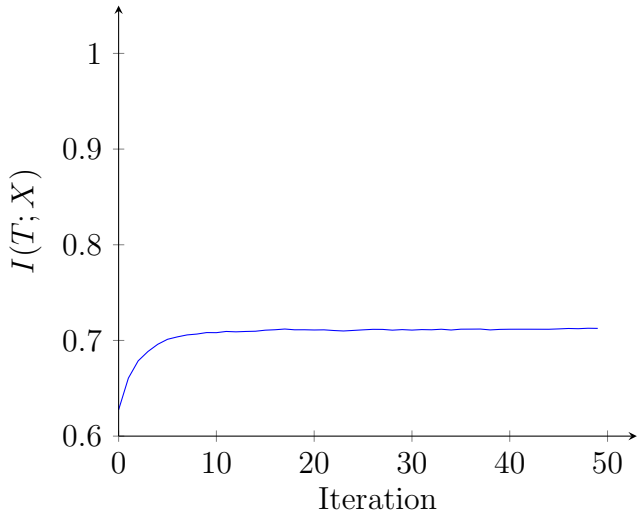


Figure A.2: Evolution of mutual information for design-$E_b/N_0 = 0.95$ dB

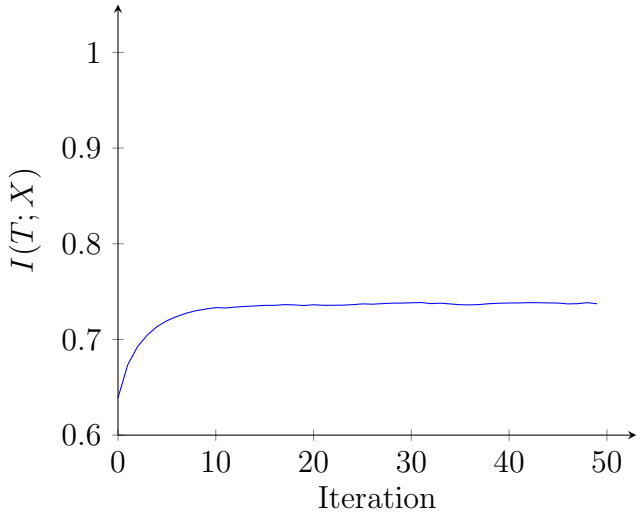


Figure A.3: Evolution of mutual information for design-$E_b/N_0 = 1.05$ dB

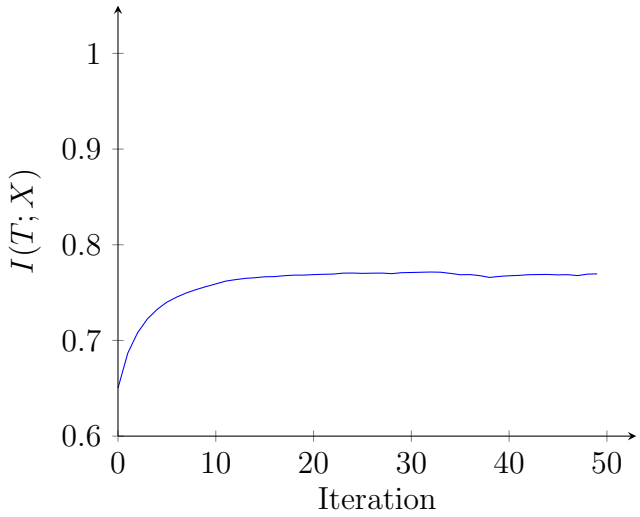


Figure A.4: Evolution of mutual information for design-$E_b/N_0 = 1.15$ dB

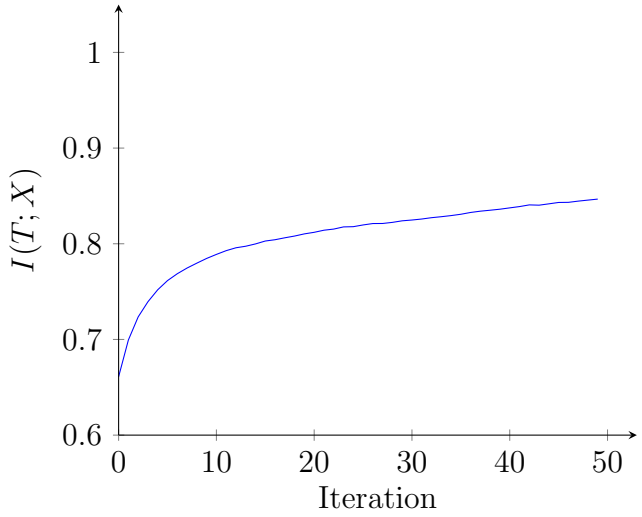


Figure A.5: Evolution of mutual information for design-$E_b/N_0 = 1.25$ dB

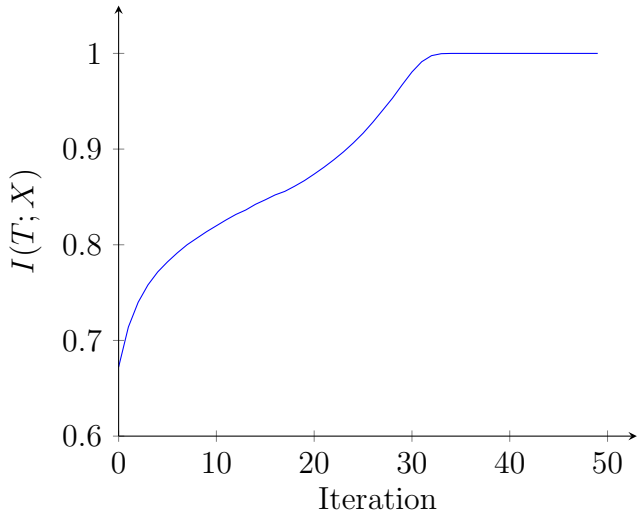


Figure A.6: Evolution of mutual information for design-$E_b/N_0 = 1.35$ dB

Appendix B

Additional Simulation Results for Irregular LDPC Code from IEEE 802.11 Standard

The chosen code from the WLAN standard has length $N = 1296$, the node-degree distribution is already defined in Chapter 9. This Appendix contains BER simulation results for discrete decoders using only 16 or 32 distinct clusters as indicated in the corresponding caption.

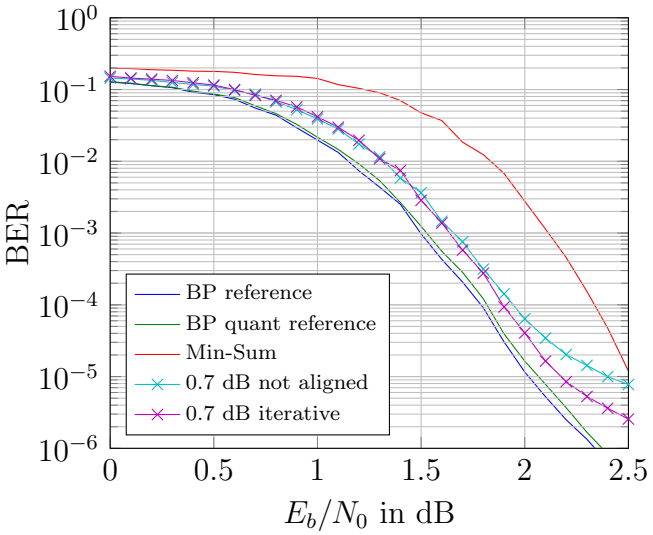


Figure B.1: Simulation results for design-$E_b/N_0 = 0.7$ dB, $|\mathcal{T}| = 16$

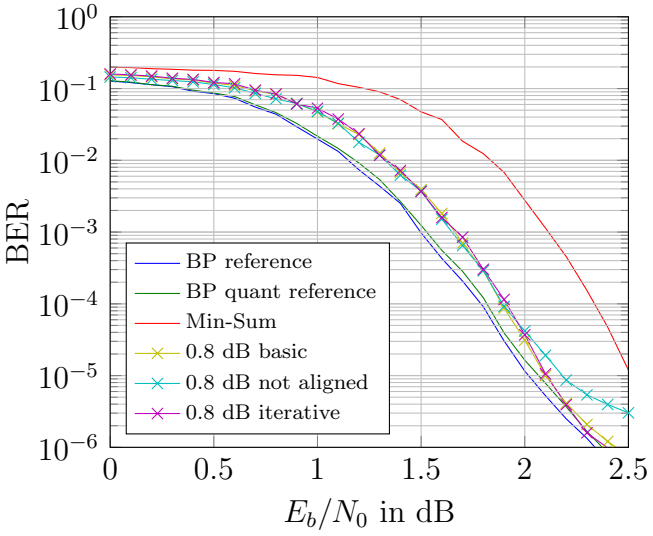


Figure B.2: Simulation results for design-$E_b/N_0 = 0.8$ dB, $|\mathcal{T}| = 16$

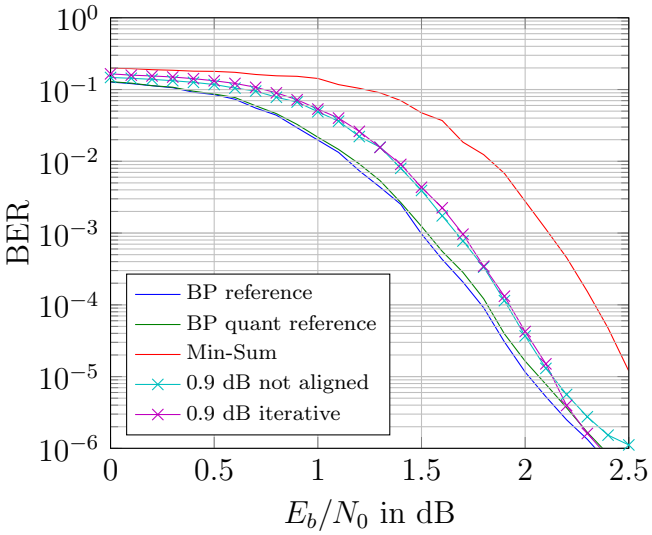


Figure B.3: Simulation results for design-$E_b/N_0 = 0.9$ dB, $|\mathcal{T}| = 16$

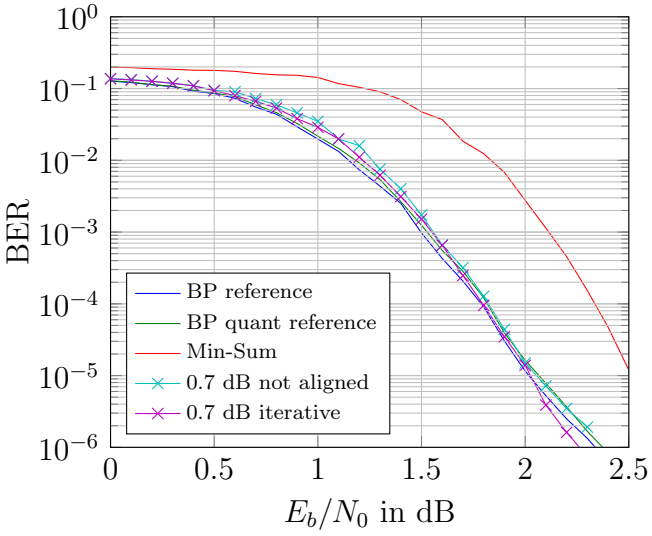


Figure B.4: Simulation results for design-$E_b/N_0 = 0.7$ dB, $|\mathcal{T}| = 32$

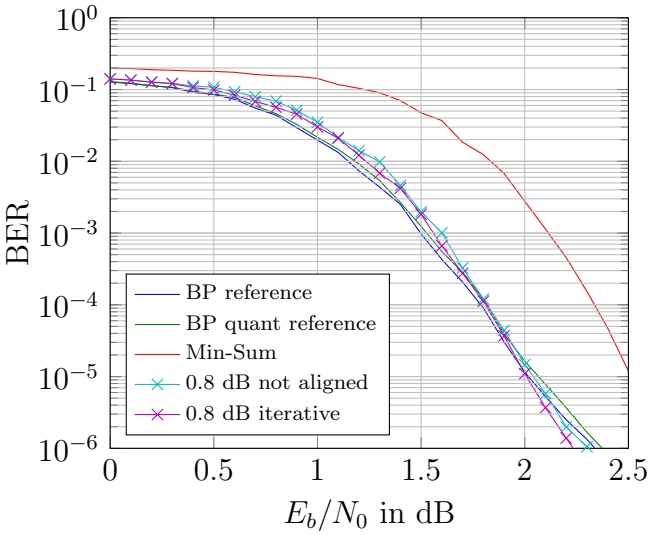


Figure B.5: Simulation results for design-$E_b/N_0 = 0.8$ dB, $|\mathcal{T}| = 32$

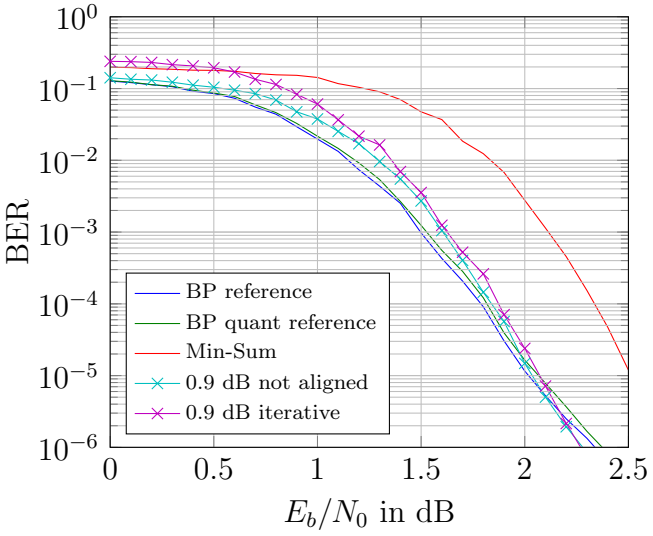


Figure B.6: Simulation results for design-$E_b/N_0 = 0.9$ dB, $|\mathcal{T}| = 32$

Appendix C

Additional Simulation Results for Irregular LDPC Code from DVB-S2 Standard

The chosen code from the DVB-S2 standard has length $N = 64800$, the node-degree distribution was already defined in Chapter 9. This Appendix contains BER simulation results for discrete decoders using only 16 or 32 distinct clusters as indicated in the corresponding caption.

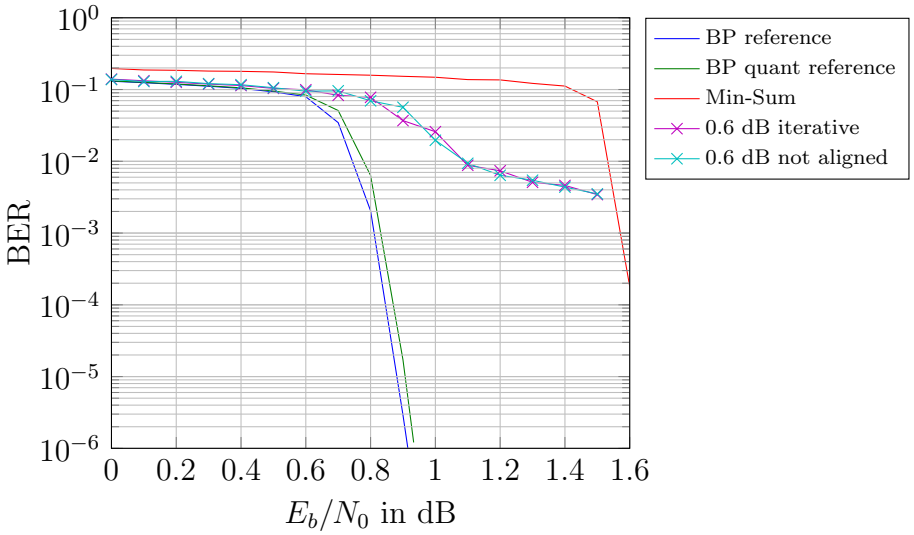


Figure C.1: Simulation results for design-$E_b/N_0 = 0.6$ dB, $|\mathcal{T}| = 16$

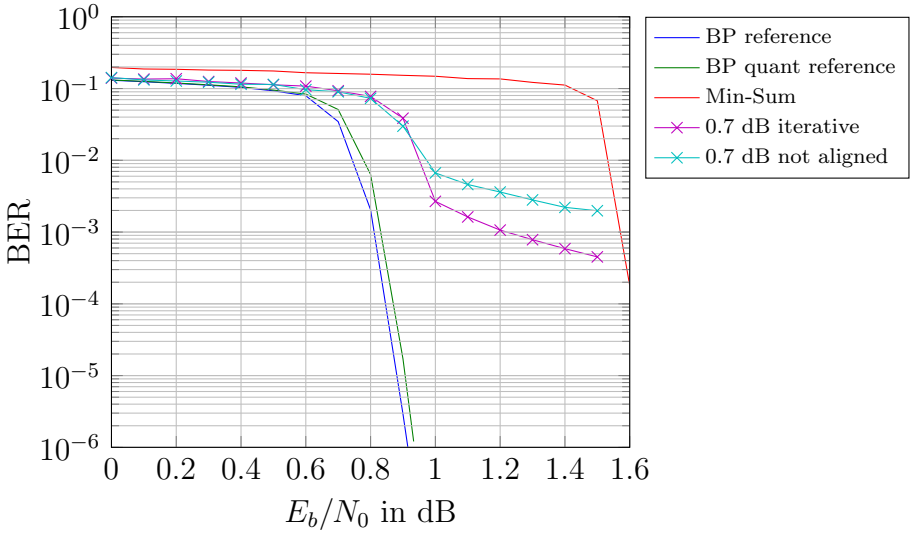


Figure C.2: Simulation results for design-$E_b/N_0 = 0.7$ dB, $|\mathcal{T}| = 16$

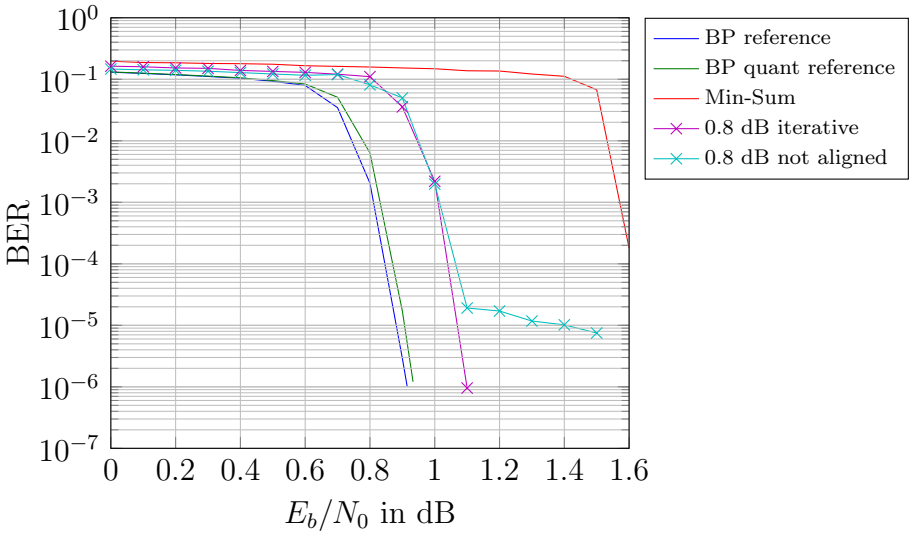


Figure C.3: Simulation results for design-$E_b/N_0 = 0.8$ dB, $|\mathcal{T}| = 16$

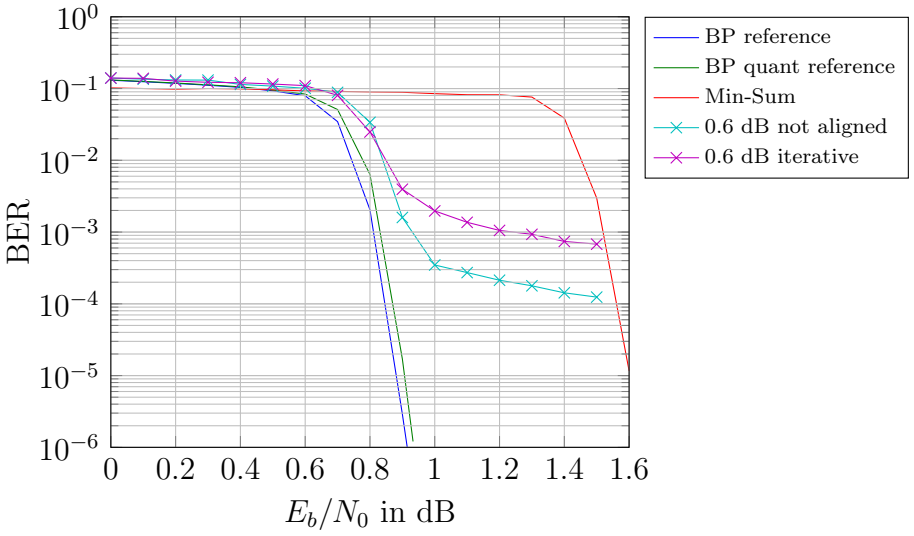


Figure C.4: Simulation results for design-$E_b/N_0 = 0.6$ dB, $|\mathcal{T}| = 32$

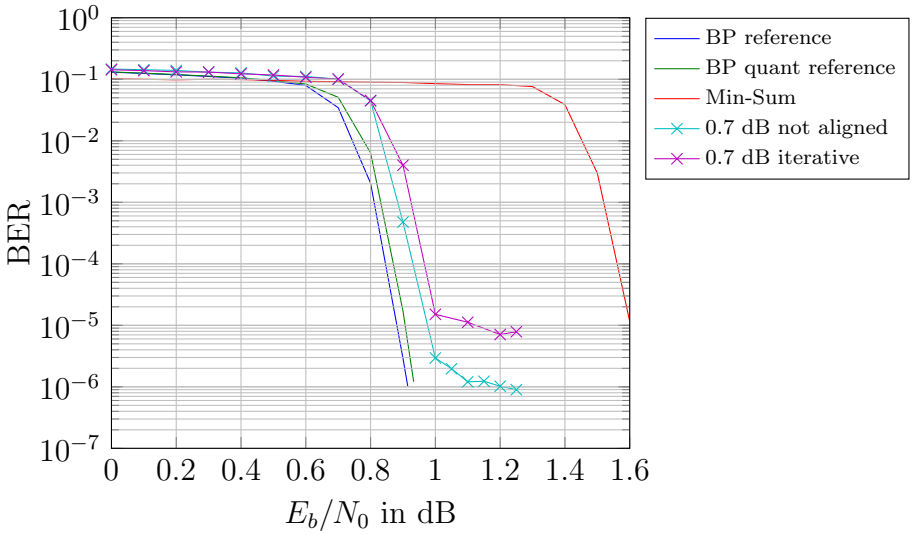


Figure C.5: Simulation results for design-$E_b/N_0 = 0.7$ dB, $|\mathcal{T}| = 32$

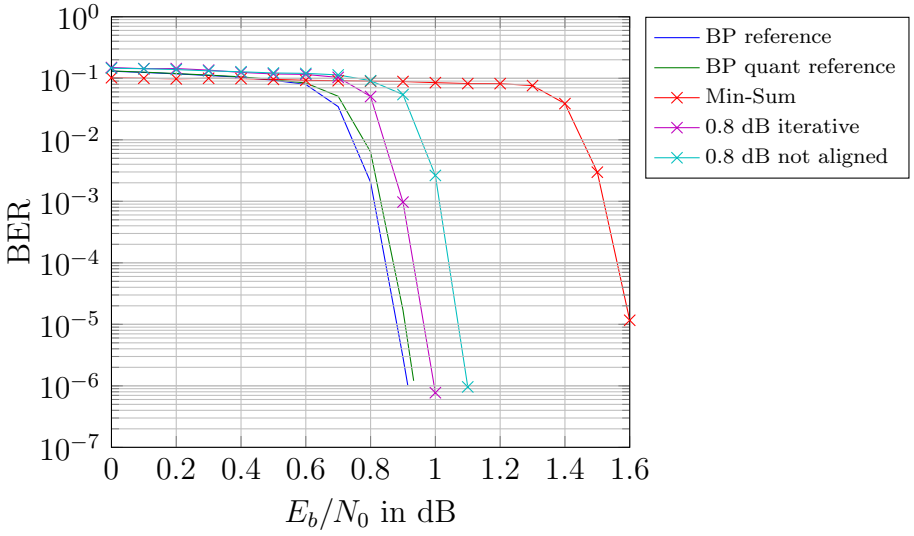


Figure C.6: Simulation results for design-$E_b/N_0 = 0.8$ dB, $|\mathcal{T}| = 32$

Bibliography

[Ama16] S.-i. Amari, *Information geometry and its applications*. Springer, 2016, vol. 194.

[AORS11] R. Ansorge, H. J. Oberle, K. Rothe, and T. Sonar, *Mathematik für Ingenieure: Differential-und Integralrechnung, Differentialgleichungen, Integraltransformationen, Funktionen einer komplexen Variablen*. John Wiley & Sons, 2011, vol. 2.

[BGT93] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Communications, 1993. ICC'93 Geneva. Technical Program, Conference Record, IEEE International Conference on*, vol. 2. IEEE, 1993, pp. 1064–1070.

[Bis07] C. Bishop, "Pattern recognition and machine learning (information science and statistics), 1st edn. 2006. corr. 2nd printing edn," 2007.

[Bla72] R. Blahut, "Computation of channel capacity and rate-distortion functions," *IEEE transactions on Information Theory*, vol. 18, no. 4, pp. 460–473, 1972.

[BSCG13] S. K. Buddha, K. So, J. M. Carmena, and M. C. Gastpar, "Function identification in neuron populations via information bottleneck," *Entropy*, vol. 15, no. 5, pp. 1587–1608, 2013.

[CFRU01] S.-Y. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 db of the shannon limit," *IEEE Communications letters*, vol. 5, no. 2, pp. 58–60, 2001.

[CT12] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[ETS14] ETSI, "Second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications," 2014.

[Gal62] R. Gallager, "Low-density parity-check codes," *IRE Transactions on information theory*, vol. 8, no. 1, pp. 21–28, 1962.

[HOP96] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on information theory*, vol. 42, no. 2, pp. 429–445, 1996.

[IEE12] IEEE, "IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area network - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications." *IEEE Std. 802.11-2012*, p. 1602, 2012.

[JL10] B. Jin and X. Lu, "Identifying informative subsets of the gene ontology with information bottleneck methods," *Bioinformatics*, vol. 26, no. 19, pp. 2445–2451, 2010.

[Joh09] S. J. Johnson, *Iterative error correction: turbo, low-density parity-check and repeat-accumulate codes*. Cambridge University Press, 2009.

[Kam13] K.-D. Kammeyer, *Nachrichtenübertragung*. Springer-Verlag, 2013.

[KFL01] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 498–519, 2001.

[Ksc03] F. R. Kschischang, "Codes defined on graphs," *IEEE Communications Magazine*, vol. 41, no. 8, pp. 118–125, 2003.

[KY14] B. M. Kurkoski and H. Yagi, "Quantization of binary-input discrete memoryless channels," *IEEE Transactions on Information Theory*, vol. 60, no. 8, pp. 4544–4552, 2014.

[KYK08] B. M. Kurkoski, K. Yamaguchi, and K. Kobayashi, "Noise thresholds for discrete ldpc decoding mappings," in *IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference*. IEEE, 2008, pp. 1–5.

[LB15] J. Lewandowsky and G. Bauch, "Trellis based node operations for ldpc decoders from the information bottleneck method," in *Signal Processing and Communication Systems (ICSPCS), 2015 9th International Conference on*. IEEE, 2015, pp. 1–10.

[Loe04] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Magazine*, vol. 21, no. 1, pp. 28–41, 2004.

[LSB16a] J. Lewandowsky, M. Stark, and G. Bauch, "Information bottleneck graphs for receiver design," in *Information Theory (ISIT), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 2888–2892.

[LSB16b] J. Lewandowsky, M. Stark, and G. Bauch, "Optimum message mapping ldpc decoders derived from the sum-product algorithm," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–6.

[M+05] T. Minka *et al.*, "Divergence measures and message passing," Technical report, Microsoft Research, Tech. Rep., 2005.

[Mac] D. J. C. MacKay, "Encyclopedia of sparse graph codes." [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html

[MN95] D. J. MacKay and R. M. Neal, "Good codes based on very sparse matrices," in *IMA International Conference on Cryptography and Coding*. Springer, 1995, pp. 100–111.

[RL09] W. Ryan and S. Lin, *Channel codes: classical and modern*. Cambridge University Press, 2009.

[Sha01] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[Slo02] N. Slonim, "The information bottleneck: Theory and applications," Ph.D. dissertation, Hebrew University of Jerusalem, 2002.

[ST00] N. Slonim and N. Tishby, "Document clustering using word clusters via the information bottleneck method," in *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2000, pp. 208–215.

[Sti14] S. Still, "Information bottleneck approach to predictive inference," *Entropy*, vol. 16, no. 2, pp. 968–989, 2014.

[Stü11] G. L. Stüber, *Principles of mobile communication*. Springer Science & Business Media, 2011.

[TPB00] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," *arXiv preprint physics/0004057*, 2000.

[TZ15] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *Information Theory Workshop (ITW), 2015 IEEE*. IEEE, 2015, pp. 1–5.

[WLW09] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," *Proceedings of the IEEE*, vol. 97, no. 2, pp. 427–450, 2009.

List of Figures