Unequal Error Protection Modulation to Support Low-Density Parity-Check Code Design

Alexandra Filip

a.filip@jacobs-university.de

A report submitted as a requirement for the degree of

Master of Science

February, 2012

Jacobs University Bremen
Supervisor: Prof.Dr.-Ing.Werner Henkel

Abstract

We propose a linear programming approach to optimize the variable node degree distribution for a Low-Density Parity-Check code that will provide UEP inside a modulation symbol. Based on a rate maximization criterion, the optimization of the degree distribution is designed for a higher order hierarchical constellation and requires splitting the variable node degree distribution into sub-degree distributions corresponding to the different modulation classes. Once the optimized distributions are available, we continue with obtaining the parity-check matrix of the code and with assigning the different bits inside a codeword to the channels with different noise variances. The noise variances are priorly obtained using the different bit-error probabilities of the bits inside a symbol. As a result, we opt for an irregular LDPC code and we add irregularities to the channel by using different modulation classes originating especially from different distances in a hierarchical modulation signal set. Our aim is to assess if the unequal protection modulation supports in any way the resulting variable node degree distribution. Furthermore, we evaluate the performance of the obtained codes for different constellation parameters by comparing the bit-error ratio performance curves.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Low-Density Parity-Check (LDPC) codes, first introduced by Robert Gallager in his PhD thesis [1], have been shown to perform very close to capacity, as described by the Shannon capacity formula. These results have motivated further research on LDPC codes for different channels and noise scenarios.

For this research work, we have focused on designing irregular LDPC codes due to their improved performance, as presented in the literature, in comparison to the regular ones. We introduced non-uniformities by using a non-uniform (hierarchical) modulation which results in different error probabilities for the different sets of bits inside a symbol. This unequal error protection (UEP) modulation allowed us to protect the bits inside a modulation symbol differently.

We therefore aim to design one LDPC code to provide UEP inside a modulation symbol for a higher order constellation (HOC). For the case of a 16-QAM hierarchical constellation, we obtain two equivalent channels which protect the bits differently such that the most significant bits will be assigned to the first channel while the less significat bits will be sent on second channel. The level of UEP between the two modulation classes can be easily varied with the help of the constellation parameter which is defined as the ratio between the intersymbol distances describing the constellation.

Further on, by keeping the check node degree distribution fixed, we aim at optimizing the variable node degree distribution after dividing it into sub-degree distributions corresponding to the different modulation classes. The optimization is done using density evolution and a linear programming technique based on a rate maximization criterion. Once the optimized variable node degree distribution is obtained, we continue with constructing the parity check matrix of the code using the progressive edge-growth (PEG) construction algorithm.

This report is organized as follows. Chapter 2 introduces background concepts needed to address the topics of this research work. Chapter 3 describes the system model, the implementation steps for optimizing the variable node degree distribution as well as the linear programming and the PEG algorithm. Chapter 4 presents the simulation results and the analysis. Chapter 5 summarizes the idea of this project, discusses the main findings, and addresses future possible work.

Chapter 2

Background

2.1 Low-Density Parity-Check (LDPC) Codes

2.1.1 Fundamentals

Low-Density Parity-Check (LDPC) codes are linear block codes described, as the name suggests, by a sparse parity-check matrix $\mathbf{H}$ with dimensions $(N-K) \times N$ where N represents the length of the codeword and K is the length of the information word. Accordingly, the rate of the code is expressed as $R = \frac{K}{N}$.

LDPC codes can be represented using a bipartite graph, also called a Tanner graph, as introduced in [3]. A Tanner graph is constructed using two types of nodes, variables nodes corresponding to the elements of the codeword and check nodes corresponding to the parity-check constraints, as well as the edges connecting them.

In order to describe how the parity-check matrix of an LDPC code maps to the corresponding Tanner graph we introduce the parity-check matrix below with $N = 6$ and $K = 2$.

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} . \tag{2.1}$$

The equivalent Tanner graph is presented in Fig. 2.1.

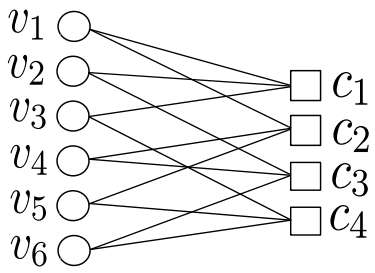

Figure 2.1: (2,3) regular Tanner graph

The circles on the left denote the variable nodes, v_i, or equivalently the columns of the H matrix while the squares on the right denote the parity-check constraints, c_j, which are given by the rows of the **H** matrix. The connection of any variable node to any check node by an edge e_t is controlled by the corresponding entry in the H matrix. As a result, the edges in the bipartite graph correspond to the non-zero entries in the **H** matrix. From the perspective of how connected a variable/check node is, the two categories of nodes behave differently. While for a variable node more connections result in a better protection against errors, for a check node fewer connections are preferred.

The number of edges connected to any node gives the degree of the respective node. Accordingly, we can distinguish two types of parity check matrices (or associated graphs): regular and irregular. Thus, regular LDPC codes have the property that the corresponding graph is regular and that inside each class of nodes (variable and check) all the nodes have the same degree. The Tanner graph presented in Fig. 2.1 has this characteristic since all the variable nodes have degree 2 while all the check nodes have degree 4. Consequently, if this property is not present and the degrees of the variable/check nodes vary among the classes of nodes, we refer to them as irregular LDPC codes.

Before further introducing the irregular LDPC codes, we present the definitions for two important concepts when using codes on graphs: cycle and girth. A closed path with edges starting from a node and ending at the same node is called a *cycle*. In addition, the length of the shortest cycle is referred to as the *girth* of the code.

2.1.2 Irregular LDPC Codes

Irregular LDPC codes have been shown to perform better than regular ones. For the case of irregular graphs the degrees of the two sets of nodes are chosen according to some degree distribution, using polynomials.

Moreover, for $\gamma(x)$ to be a degree distribution we require that the coefficients of $\gamma(x)$ are nonnegative and that $\gamma(1) = 1$.

The irregular variable and check node degree distributions can be expressed both from an edge as well as from a node perspective.

Considering the edge perspective first, we have

$$\lambda(x) = \sum_{i=2}^{d_{v_{max}}} \lambda_i x^{i-1} \tag{2.2}$$

for the variable node degree distribution and

$$\rho(x) = \sum_{i=2}^{d_{c_{max}}} \rho_i x^{i-1} \tag{2.3}$$

for the check node degree distribution, where $d_{v_{max}}$ and $d_{c_{max}}$ represent the maximum variable and check node degrees, respectively, while λ_i and ρ_i give the proportion of edges connected to variable and check nodes of degree i.

Similarly, the degree distributions from a node perspective are defined as

$$\widetilde{\lambda}(x) = \sum_{i=2}^{d_{vmax}} \widetilde{\lambda}_i x^{i-1} \text{ and } \widetilde{\rho}(x) = \sum_{i=2}^{d_{cmax}} \widetilde{\rho}_i x^{i-1} , \tag{2.4}$$

where the coefficients $\widetilde{\lambda}_i$ and $\widetilde{\rho}_i$ represent the proportions of variable and check nodes of degree i.
We then have

$$\lambda_i = \frac{\Lambda_i \cdot i}{E} \text{ and } \widetilde{\lambda}_i = \frac{\Lambda_i}{N} , \tag{2.5}$$

where Λ_i represents the number of variable nodes of degree i, $E = \sum_i \Lambda_i \cdot i$ gives the total number of edges and we have a total of $N = \sum_i \Lambda_i$ variable nodes.

Therefore, we can find the relation between $\lambda_i(\rho_i)$ and $\widetilde{\lambda}_i(\widetilde{\rho}_i)$

$$\widetilde{\lambda}_i = \frac{\Lambda_i}{N} = \frac{\Lambda_i/E}{N/E} = \frac{\lambda_i/i}{\sum_{j\geq 2} \lambda_j/j} \text{ and } \widetilde{\rho}_i = \frac{\rho_i/i}{\sum_{j\geq 2} \rho_j/j} . \tag{2.6}$$

In order to obtain the relation between λ_i and ρ_i, we realize that the total edge count (E) from the variable and from the check node perspective should match. The number of variable nodes of degree i is given by

$$\Lambda_i = N\widetilde{\lambda}_i = N\frac{\lambda_i/i}{\sum_{j\geq 2} \lambda_j/j} = N\frac{\lambda_i/i}{\int_0^1 \lambda(x)dx} , \tag{2.7}$$

and as a result the number of edges emanating from all variable nodes is

$$E = \sum_i \Lambda_i \cdot i = N\sum_{i\geq 2} \frac{\lambda_i}{\int_0^1 \lambda(x)dx} = N\frac{\sum_{i\geq 2} \lambda_i}{\int_0^1 \lambda(x)dx} = N\frac{1}{\int_0^1 \lambda(x)dx} , \tag{2.8}$$

since $\sum_{i\geq 2} \lambda_i = 1$.
Similarly, for M check nodes the total number of edges emanating from them should be the same and is expressed as

$$E = M\frac{1}{\int_0^1 \rho(x)dx} . \tag{2.9}$$

As mentioned previously, the total number of edges should match and equating the two expressions for E we obtain

$$M = N\frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx} , \tag{2.10}$$

which allows us to conclude that the variable and check node degree distributions are linked by the design rate of the code $R = \frac{K}{N} = 1 - \frac{M}{N}$

$$R = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx} = 1 - \frac{\sum_{j\geq 2} \rho_j/j}{\sum_{i\geq 2} \lambda_i/i} \ . \tag{2.11}$$

2.1.3 Decoding

LDPC codes are decoded using a general class of decoding algorithms called message-passing algorithms which includes the belief propagation (BP), also called the sum-product algorithm. Using these decoding techniques, the messages between the variable and the check nodes are exchanged iteratively along the edges of the graph. As a result, a variable node will receive messages from its check node neighbors and the outgoing output messages will be a function of all the incoming messages except the one along the edge on which the output message is to be sent. This restriction is essential for a good performance of the message-passing decoders which can be degraded by the presence of short cycles. The tree-like structure supports the independence of the messages and avoids short loops. The same procedure for transmitting and processing the messages applies for the check nodes, too.

The exchanged messages are random variables (RVs) and take the form of log-likelihood ratios (LLRs)[1] such that their sign will determine the bit while their magnitude is understood as a measure of reliability of the prediction. The decoder update rules for the variable and check nodes during BP are [10]

$$L_{v_i c_j}^{(l)} = \begin{cases} L_{ch,i}, & \text{if } l = 0 \\ L_{ch,i} + \sum_{k\neq j} L_{c_k v_i}^{(l)}, & \text{if } l \geq 1 \end{cases} \ , \tag{2.12}$$

$$L_{c_j v_i}^{(l)} = 2 \tanh^{-1} \left(\prod_{k\neq j} \tanh \left(L_{v_k c_j}^{(l-1)} \right) \right) \ , \tag{2.13}$$

respectively, where $L_{v_i c_j}^{(l)}$ represents the LLR message from the variable node i to the check node j during the l^{th} iteration and $L_{ch,i}$ represents the log-likelihood ratio of the codeword bit associated with the variable node v_i based only on the channel information for this bit. The summation/product does not include the message from the edge on which the resulting message will be sent. For every discrete time step l, one iteration of message exchanging begins with the check nodes processing the information received from the previous iteration and transmitting the messages to their neighbors, continued with the variable nodes processing the information received during the same iteration and sending it back to the check nodes to be processed in the following iteration. We further observe that at $l = 0$ the initial message from each of the check nodes is 0 while at this point the variable nodes

[1]The log-likelihood ratio of a received value y is $L(x) = \ln \frac{p(x=+1/y)}{p(x=-1/y)}$.

will transmit the intrinsic information from the channel[2] to the check nodes to be available during the next iteration.

2.1.4 Density Evolution

As shown in [5] the messages at the input of the variable and check nodes can be computed as mutual information using density evolution (DE) and a Gaussian approximation.

Density evolution is an algorithm used to predict the decoding performance by analyzing the distribution of the messages transmitted under message-passing decoding [6]. As mentioned in [7] the evolution of the message distributions is investigated considering that the random variable messages are independent.

The Gaussian approximation differs for regular and irregular LDPC codes. The empirical results performed on regular graphs using density evolution, in [5], showed that for the regular case the output messages in (2.12) and (2.13)[3] can be well approximated by Gaussian densities. For the message at the output of a variable node the density can be well approximated by a Gaussian because, despite of the incoming messages being Gaussian or not, the sum will be Gaussian, either by summation of Gaussians or by the central limit theorem. However, the Gaussian assumption for the message densities at the output of the check nodes is not as rigouros as the one for variable nodes due to the tanh computation present in the update rule (which applies for both the regular and the irregular case). The accuracy of the approximation for the check nodes tends to decrease in particular when the mean of the output message is close to zero [5].

On the other hand, if we consider irregular LDPC codes, the assumption that the individual outputs of a variable and a check node are Gaussian still applies, but the message densities are approximated to be a Gaussian mixture due to the different densities from nodes with different degrees [5].

After this approximation is considered, one can readily apply the properties of a Gaussian for the exchanged messages. Since a Gaussian is a function of only its mean and its variance our algorithm is greatly simplified by having to store only the means and the variances at each iteration. Furthermore, as proved in [6], the consistency[4] is preserved during density evolution and, as a result, allows for further simplification when applied. Therefore, we have the LLR of a Gaussian to be a consistent random variable while the mean m and the variance σ^2 are linked by $\sigma^2 = 2m$ further reducing the required quantities to only the storage of the mean.

Applying density evolution along with the Gaussian approximation, we obtain the mutual information messages exchanged between the variable and the check nodes

[2]The intrinsic LLR is the maximum likelihood (ML) ratio, i.e., $\ln \frac{p(y|x=+1)}{p(y|x=-1)}$

[3]Each node will get independent and identically distributed (i.i.d) messages from its neighbors.

[4]A density of probability $f(x)$ is said to be consistent if and only if $f(x) = f(-x)e^x$, where $f(x)$ represents the density of an LLR message, is met for $\forall x \in \mathbb{R}$

given in [8]

$$x_{cv}^{(l-1)} = 1 - \sum_{j=2}^{d_{c_{max}}} \rho_j J\left((j-1)\, J^{-1}\left(1 - x_{vc}^{(l-1)}\right)\right) , \tag{2.14}$$

$$x_{vc}^{(l)} = \sum_{i=2}^{d_{v_{max}}} \lambda_i J\left(\frac{2}{\sigma^2} + (i-1)\, J^{-1}\left(x_{cv}^{(l-1)}\right)\right) , \tag{2.15}$$

where l denotes the iteration number and $J(\cdot)$ computes the mutual information as a function of the mean, $x = J(m)$ using

$$J(m) = 1 - \frac{1}{\sqrt{4\pi m}} \int_{\mathbb{R}} \log_2\left(1 + e^{-z}\right) \cdot e^{-\frac{(z-m)^2}{4m}} dz , \tag{2.16}$$

which holds for $z \sim \mathcal{N}(m, 2m)$, a consistent Gaussian random variable.

From (2.14) and (2.15), density evolution can be summarized as a function of the degree distributions, noise variance, and the mutual information from the previous iteration as

$$x_{vc}^{(l)} = F\left(\boldsymbol{\lambda}, \boldsymbol{\rho}, \boldsymbol{\sigma}^2, x_{vc}^{(l-1)}\right) . \tag{2.17}$$

In order to ensure that density evolution allows for predicting the decoding behavior, the mutual information has to increase after every iteration which results in the convergence of the density evolution for the respective noise variance

$$x_{vc}^{(l)} > x_{vc}^{(l-1)} \quad \text{or} \quad F\left(\boldsymbol{\lambda}, \boldsymbol{\rho}, \boldsymbol{\sigma}^2, x_{vc}^{(l-1)}\right) > x_{vc}^{(l-1)} . \tag{2.18}$$

The constraint above is refered to as the convergence condition.

Another condition that has to be fulfilled to guarantee that density evolution converges for a mutual information close to one is the stability condition [8]. This condition yields an upper bound on the number of degree-2 variable nodes. Considering that all the bits are affected by the same noise variance, the stability condition is formulated as

$$\frac{1}{\lambda^{'}(0)\rho^{'}(1)} > e^{-r} = \int_{\mathbb{R}} P_0(x)\, e^{-\frac{x}{2}} dx , \tag{2.19}$$

where $P_0(x)$ represents the message density associated with the received values while $\lambda^{'}(x)$ and $\rho^{'}(x)$ are the derivatives of the degree polynomials. For an AWGN channel, e^{-r} can be derived as in [9]

$$e^{-r} = e^{-\frac{1}{2\sigma^2}} , \tag{2.20}$$

obtaining another formulation of the stability condition

$$\lambda^{'}(0) < \left[\rho^{'}(1) e^{-\frac{1}{2\sigma^2}}\right]^{-1} . \tag{2.21}$$

2.2 Hierarchical Modulation

For this project we will investigate if and how a 16-QAM hierarchical constellation can support the design of irregular LDPC codes. This section is devoted to introducing the concept of hierarchical modulation, showing how unequal error protection (UEP) can be achieved using this technique, as well as presenting the motivation behind using it together with irregular LDPC codes. The error probabilities of the different bits inside a symbol will be investigated.

Figure 2.2 presents a traditional, 16-QAM uniform constellation for which the bit mapping has been done using a Gray code.

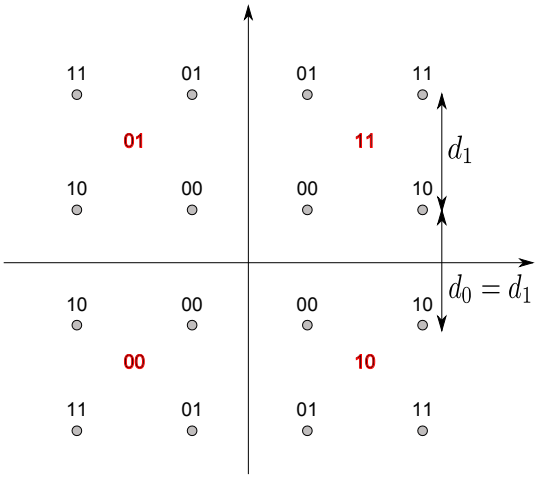

Figure 2.2: 16-QAM constellation with Gray-coded bit mapping

In comparison to Fig. 2.2, Fig. 2.3 introduces the non-uniform, hierarchical version of the traditional one.

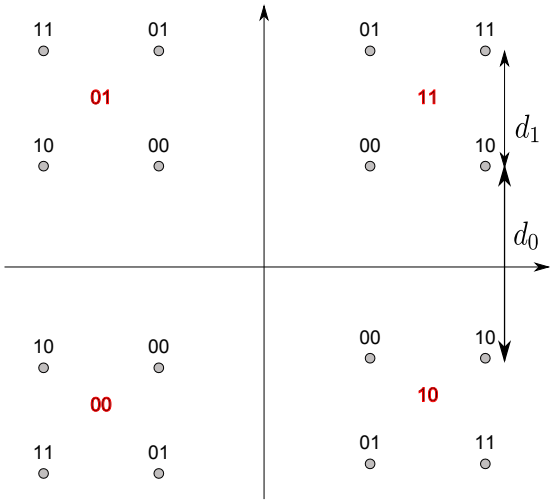

Figure 2.3: 16-QAM hierarchical constellation with Gray-coded bit mapping

One can easily notice that the non-uniformity of the symbols in the hierarchical constellation will result in error probabilities which are considerably different for the individual bits inside the symbol. Hierarchical modulation is therefore of interest when UEP inside a symbol is desired. Another beneficial outcome is the flexibility in controlling the amount of UEP, made possible by varying the ratio of the distances d_0 (minimum distance between quadrants) and d_1 (minimum distance between the symbols inside a quadrant).

Referring to the labeling presented in Fig. 2.3, we can observe that the hierarchical constellation appears as being formed from an outer QPSK scheme which is addressed by the two most significant bits (MSBs) of the symbol and an inner QPSK scheme described by the remaining least significant bits of the symbol (LSBs). As a result, using 16-QAM hierarchical modulation, we obtain two equivalent channels (two modulation classes), one corresponding to the outer scheme on which the most important information bits should be assigned and the second on which the less significant bits will be sent. The ratio of the distances will dictate how much more the first channel is protected against errors than the second one.

For example, considering that we are interested in providing UEP and we require a small error probability for the MSBs, this can be easily achieved by increasing, as much as needed, d_0. Nevertheless, as one might predict, the protection of the important bits is achieved at the expense of the quality of the less significant ones which results in a decrease of the overall channel capacity of the hierarchical modulation compared to the uniform case [10].

Using the exact formulas derived in [11] for the channels' bit-error probabilities in the case of a Gray-coded hierarchical 16-QAM constellation, we have for the first (and better protected) channel

$$P_{b,c_1} = \frac{1}{2}\left(\frac{1}{2}\mathrm{erfc}\left(\sqrt{\frac{E_s}{N_0}}\frac{d_0}{2}\right) + \frac{1}{2}\mathrm{erfc}\left(\sqrt{\frac{E_s}{N_0}}\left(\frac{d_0}{2}+d_1\right)\right)\right) , \tag{2.22}$$

while for the second one, we obtain

$$P_{b,c_2} = \frac{1}{2}\left(\mathrm{erfc}\left(\sqrt{\frac{E_s}{N_0}}\frac{d_1}{2}\right) + \frac{1}{2}\mathrm{erfc}\left(\sqrt{\frac{E_s}{N_0}}\left(d_0+\frac{d_1}{2}\right)\right) - \frac{1}{2}\mathrm{erfc}\left(\sqrt{\frac{E_s}{N_0}}\left(d_0+\frac{3d_1}{2}\right)\right)\right) , \tag{2.23}$$

where "erfc" denotes the complementary error function and E_s/N_0 represents the signal-to-noise ratio with respect to the symbol energy[5]. Accordingly, the average energy for the hierarchical 16-QAM constellation takes the form

$$E_{\mathrm{avg}} = \frac{4}{16}\left[4\left(\frac{d_0}{2}\right)^2 + 4\left(\frac{d_0}{2}+d_1\right)^2\right] \tag{2.24}$$

$$= \frac{d_0^2}{2} + d_0 d_1 + d_1^2 , \tag{2.25}$$

[5]The relation between the symbol energy and the energy of an information bit in the case of a coded system with modulation rate $R_m = \log_2(M)$ and code rate R_c is $E_s = R_c \cdot R_m \cdot E_b$

and is normalized to 1 such that we obtain the relation between d_0, d_1 and the constellation parameter $\alpha = \frac{d_0}{d_1}$ which we will vary, to be

$$d_1 = \sqrt{\frac{1}{\frac{\alpha^2}{2} + \alpha + 1}} \tag{2.26}$$

$$d_0 = \alpha d_1 \, . \tag{2.27}$$

Nevertheless, as we will make explicit in the next sections, for the DE mutual information updates, the individual noise variances for the different modulation classes are required. The noise variance for each modulation class can be computed from the corresponding bit-error probability using the equivalent BPSK description of the channels as in [8]. Thus, the error probability of the corresponding binary symmetric channel is given as

$$P_{b,c_1} = \frac{1}{2} \operatorname{erfc}\left(\frac{1}{\sqrt{2}\sigma_{c_1}}\right) \quad \text{and} \quad P_{b,c_2} = \frac{1}{2} \operatorname{erfc}\left(\frac{1}{\sqrt{2}\sigma_{c_2}}\right) \, , \tag{2.28}$$

where the bit-error probability of the first channel P_{b,c_1} stands for the bit-error probability of the first two bits (P_{b,b_0} and P_{b,b_1}) in the symbol while P_{b,c_2} corresponds to the remaining two bits (P_{b,b_2} and P_{b,b_3}).
Accordingly, we obtain the noise variances to be

$$\sigma_{c_1}^2 = \frac{1}{2\left(\operatorname{erfc}^{-1}\left(2P_{b,c_1}\right)\right)^2} \quad \text{and} \quad \sigma_{c_2}^2 = \frac{1}{2\left(\operatorname{erfc}^{-1}\left(2P_{b,c_2}\right)\right)^2} \, . \tag{2.29}$$

Regarding the motivation behind researching the effect of unequal error protection modulation with LDPC codes, our aim is to investigate if the non-uniformity introduced by the different error probabilities of the bits, as a result of using hierarchical modulation, will lead to less irregular variable node degree distributions.

2.3 LDPC code design for hierarchical higher order constellations

Most of the time, LDPC codes are designed assuming that all the bits are transmitted over one equivalent channel described by the corresponding noise variance σ^2 and protected to the same extent against errors. Since we are using hierarchical modulation and we obtain two different modulation classes, described by different error probabilities and implicitly noise variances, the design of the LDPC code will change as well.

One important observation is that we are interested in designing only one LDPC code for the resulting channels and not an individual code for each level of protection (multilevel coding). The reason why we opt for one LDPC code that will provide UEP inside a codeword is the fact that long LDPC codes perform better and mostly, the number of bits to be protected more are fewer than the ones that are less significant.

The different assumptions made for the standard code design and for the HOC design as introduced in [8] are presented in Fig. 2.4.

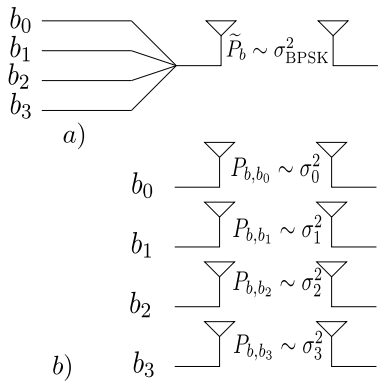

Figure 2.4: Assumptions made for a) Standard code design and b) HOC design

In the above, $\sigma_0^2 = \sigma_1^2 = \sigma_{c_1}^2$ while $\sigma_2^2 = \sigma_3^2 = \sigma_{c_2}^2$ while σ^2_{BPSK} represents the noise variance of a channel that would be described by the average bit-error probability of the HOC scenario and provides a fair way to compare the two designs [8].

Chapter 3

Implementation

Our design consists of one LDPC code for which we keep the check node degree distribution fixed and we adapt the variable node degree distribution by dividing it into sub-degree distributions and using the different protection levels from the modulation classes to optimize the resulting sub-degree distributions using a linear programming (LP) technique. Apart from the natural protection resulting from the hierarchical modulation we do not consider any other UEP capability for our system.

3.1 System Model

After applying the LDPC code every codeword bit is assigned to one of the two modulation classes available (M_j with $j = 1, \ldots, N_s = 2$). Each modulation class is described by a different bit-error ratio for the bits in a symbol such that only the bits that belong to one modulation class have the same bit-error probability.
The vector $\boldsymbol{\beta} = [\beta_1, \beta_2]$ describes the proportion of bits assigned to each modulation class and

$$N_{M_j} = \beta_j \cdot N \ , \tag{3.1}$$

represents the number of variable nodes associated with the modulation class M_j. Further on,N_{M_j} and N can then be reformulated in terms of $\boldsymbol{\lambda}_{M_j}$ and $\boldsymbol{\lambda}$

$$\frac{N_{M_j}}{E} = \sum_{i=2}^{d_{v_{max}}} \frac{\lambda_{M_j,i}}{i} \ \text{ and } \ \frac{N}{E} = \sum_{j=1}^{N_s=2} \sum_{i=2}^{d_{v_{max}}} \frac{\lambda_{M_j,i}}{i} \ . \tag{3.2}$$

The vector $\boldsymbol{\lambda}$ contains the overall node degree distribution coefficients. Accordingly, $\lambda_{M_j,i}$ represents the proportion of edges connected to variable nodes of degree i that belong to the modulation class M_j. We also define $\boldsymbol{\lambda}_{M_j}$ to be the vector characterizing the sub-degree distributions such that

$$\boldsymbol{\lambda}_{M_j} = \left[\lambda_{M_j,2}, \ldots, \lambda_{M_j,d_{v_{max}}}\right]^T \ \text{ and } \ \boldsymbol{\lambda} = \left[\boldsymbol{\lambda}_{M_1}, \boldsymbol{\lambda}_{M_2}\right]^T \ , \tag{3.3}$$

where $(\cdot)^T$ denotes the transpose on the vector on which it is applied.
As mentioned before, the check node degree distribution is constant and described by the vector

$$\boldsymbol{\rho} = \left[\rho_2, \ldots, \rho_{d_{c_{max}}}\right]^T \ . \tag{3.4}$$

3.2 Optimizing the Degree Distribution for HOC

With the previous explanations of our code design for HOC and the two resulting modulation classes we can adapt the DE mutual information updates initially formulated in (2.14) and (2.15) to incorporate the new requirements. These can be summarized by accounting for the two modulation classes and the corresponding variances, equivalent to splitting the variable node degree distribution into sub-degree distributions mapping to the different N_s noise variances.

We notice that only (2.15), the update from the variable to the check nodes, has to be adapted since the check node degree distribution is kept constant and the check to variable node updates are not affected. The update rule can then be reformulated as

$$x_{vc}^{(l)} = \sum_{j=1}^{N_s} \sum_{i=2}^{d_{v_{max}}} \lambda_{M_j,i} J\left(\frac{2}{\sigma_j^2} + (i-1)\, J^{-1}\left(x_{cv}^{(l-1)}\right)\right) . \tag{3.5}$$

The already introduced convergence condition (2.18) does not require any change while the stability condition (2.19) has to be modified to illustrate the different noise variances in our scheme, which imply different message densities. As a result, we make use of the proportions describing the modulation classes and obtain the approximation we will employ, as presented in [8],

$$e^{-r} = \int_{\mathbb{R}} \sum_{j=1}^{N_s=2} \beta_j P_{0,j}(x)\, e^{-\frac{x}{2}} dx = \sum_{j=1}^{N_s=2} \beta_j e^{-\frac{1}{2\sigma_j^2}} , \tag{3.6}$$

for the lower bound of $\frac{1}{\lambda'(0)\rho'(1)}$.

In the above, $\lambda'(0)$ gives the relation to the number of degree-2 variable nodes from the two modulation classes

$$\lambda'(0) = \sum_{j=1}^{N_s=2} \lambda_{M_j,2} = \lambda_{M_1,2} + \lambda_{M_2,2} , \tag{3.7}$$

while the derivative of the check node degree distribution is

$$\rho'(1) = \sum_{k=2}^{d_{c_{max}}} (k-1)\rho_k . \tag{3.8}$$

Apart from the two conditions presented above, another constraint arises due to the splitting of the variable node degree distribution. This new condition is refered to as the proportion distribution constraint and is based on the intuitive idea that the sub-degree distributions should function as an overall distribution and satisfy

$$\sum_{j=1}^{N_s=2} \sum_{i=2}^{d_{v_{max}}} \lambda_{M_j,i} = 1 . \tag{3.9}$$

One final remark on the density evolution constraints is the fact that they should be rephrased to be a function of only $\boldsymbol{\lambda}$, $\boldsymbol{\rho}$, $\boldsymbol{\beta}$, $\boldsymbol{\sigma}^2$, and R[1] without any dependence on the code length (N, K).

[1]However in our case R is to be maximized and is therefore not used in this context.

Having characterized density evolution for our system, we are left with optimizing the variable node degree distribution using a linear programming approach. Our choice of an optimization criterion is maximizing the code rate for a given E_s/N_0.

Recalling (2.11), we introduce the code rate formula which accounts for the modulation classes as

$$R = 1 - \frac{\sum_{k=2}^{d_{c_{max}}} \frac{\rho_k}{k}}{\sum_{j=1}^{N_s=2} \sum_{i=2}^{d_{v_{max}}} \frac{\lambda_{M_j,i}}{i}} . \tag{3.10}$$

Our optimization strategy is based on maximizing the rate and this is equivalent to maximizing the denominator of the fraction in (3.10) which will become our target function.

3.3 Linear Programming Optimization Algorithm

The linear programming algorithm requires the check node degree distribution vector $\boldsymbol{\rho}$, the constellation parameter α, the maximum variable node degree $d_{v_{max}}$, the SNR E_s/N_0, and the proportion vector $\boldsymbol{\beta}$. Consequently, the routine will deliver the optimized variable node degree distribution which maximizes the code rate R.

Optimize

$$\max_{\boldsymbol{\lambda}} \sum_{j=1}^{N_s=2} \sum_{i=2}^{d_{v_{max}}} \frac{\lambda_{M_j,i}}{i} , \tag{3.11}$$

subject to

1. Proportion distribution constraints

   (a) From (3.9)

   $$\sum_{j=1}^{N_s=2} \sum_{i=2}^{d_{v_{max}}} \lambda_{M_j,i} = 1$$

   (b) From (3.1) and (3.2), $\forall j \in \{1, 2\}$

   $$\sum_{i=2}^{d_{v_{max}}} \frac{\lambda_{M_j,i}}{i} - \beta_j \sum_{j=1}^{N_s=2} \sum_{i=2}^{d_{v_{max}}} \frac{\lambda_{M_j,i}}{i} = 0 \tag{3.12}$$

2. Convergence constraint, see (2.18)

   $$F\left(\boldsymbol{\lambda}, \boldsymbol{\rho}, \boldsymbol{\sigma}^2, x_{vc}^{(l-1)}\right) > x_{vc}^{(l-1)} \tag{3.13}$$

3. Stability condition, see (3.6), (3.7), and (3.8)

   $$\sum_{j=1}^{N_s=2} \lambda_{M_j,2} < \left[\sum_{k=2}^{d_{c_{max}}} (k-1)\rho_k \cdot \sum_{j=1}^{N_s=2} \beta_j e^{-\frac{1}{2\sigma_j^2}} \right]^{-1} \tag{3.14}$$

3.4 Progressive Edge-Growth Construction Algorithm

Once the optimized variable node degree distribution is obtained, the next step is constructing the parity-check matrix. In the literature, there exist several algorithms for constructing parity-check matrices such as: Random, Zigzag, Progressive Edge-Growth (PEG), Approximate Cycle Extrinsic Message Degree (ACE) as well as PEG-ACE and modified versions of the original ones with added capabilities. For this project, we have focused on the PEG which stands out due to its low complexity accompanied by good performance especially for the case when the degree distributions employed are optimized using density evolution [12].

From [13] it is known that iterative message-passing decoding algorithms result in optimum decoding if applied on cycle-free Tanner graphs. Hence, the construction algorithms aim at minimizing the effect of cycles by maximizing the girth of the code. Since achieving the largest possible girth involves a high complexity, suboptimum algorithms are preferred and exhibit a good performance.

Before introducing the PEG algorithm, further details on how a tree is formed are required. In general, given a Tanner graph, a tree that starts at a variable node v_i is expanded by unfolding the Tanner graph until the desired depth starting from the variable node of interest and first traversing all the edges connecting it to the check nodes. Following this, the edges further connecting each of the previously reached check-nodes, excluding the one through which they were reached, are traversed. The process continues in the same manner and is demonstrated in Fig. 3.1.

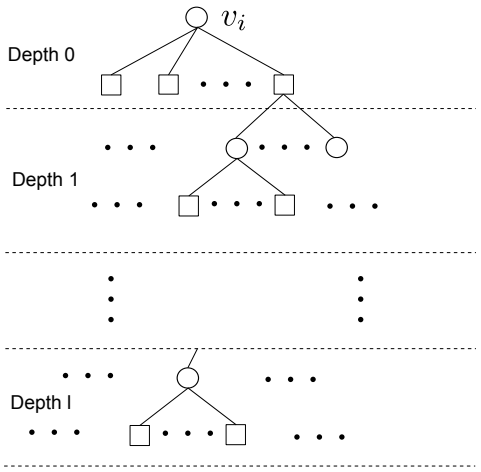

Figure 3.1: Expanded tree of depth l from a variable node v_i, taken from [12]

From the figure above, we can observe that the variable nodes that appear for the first time at depth l have a distance[2] of $2l$ to the variable node from which the tree was expanded (v_i), while the check nodes having the same properties have a distance of $2l + 1$ to v_i.

[2]The distance between two nodes is defined as the number of edges of the shortest path that connects them.

Our choice of construction algorithm, the PEG algorithm, is also suboptimum [12] and is characterized by an edge-selection procedure which supports the placement of a new edge such that it will affect the girth of the graph the least. This procedure results in maximizing the local girth of a variable node with every newly placed edge.

The PEG algorithm requires the number of variable and check nodes as well as the degree distributions. Initially, the nodes are assigned a number of empty sockets corresponding to their degree. The edge assignment proceeds with the variable nodes having the smallest degrees. To minimize the impact on the girth, a tree is expanded from the current variable node up to some depth l. The main idea is that the algorithm tries to connect the variable node to a check node not yet reached at the current depth. In case this is not possible, the check node that will lead to the largest cycle through the variable node has priority and an edge will be placed to connect to it. If there are more choices of check nodes, the check node with the lowest degree will be chosen. If there are again multiple choices, one of the check nodes is chosen at random.

As we will see, the algorithm imposes a constraint on the check nodes for them to meet the required check node degree distribution. Also, we define $\mathcal{N}_{v_i}^{l}$ as the set of check nodes already reached by a tree of depth l expanded from v_i and accordingly, $\widetilde{\mathcal{N}}_{v_i}^{l}$ to be the complementary set containing the check nodes not yet reached by the tree.

The next step, after obtaining the H matrix, consists of assigning each one of the variable nodes to one of the two channels available such that we can ensure the proper noise level during decoding as well as the proper LLR computation.

Once the parity-check matrix is obtained, we make use of the information it provides (the degree of each one of the variable nodes obtained as the sum of the elements of **H** along the first dimension) in combination with the information provided by the optimized sub-degree distributions (the number of variable nodes for the degrees available in each modulation class). Starting from the better protected channel, we assign, for each degree at a time, the required number of variable nodes as the first positions in the obtained profile of the **H** matrix which map to this degree. After each round of assigning the variable nodes, the entries corresponding to those positions in the profile of the **H** matrix are zeroed such that only the entries given the variable nodes not assigned yet will remain. This process is accompanied by flagging the assigned positions in a vector of length N, with 1 if it corresponds to the first channel and 2 if it is to be sent on the second channel. This vector will be used when allocating the equivalent binary channels as well as when computing the LLRs.

- Assign as many sockets to the variable and check nodes as their degree distribution require.
- Sort the variable nodes in ascending order according to the number of sockets assigned.
- for $i = 1$ to N
  - for $k = 1$ to d_{v_i}
    - if $k == 1$ (if first edge to be constructed)
      connect it randomly to a check node that has the lowest degree under the current graph setting
    - else (if any other edge)
      - expand a tree of depth l from the current variable node v_i until the cardinality of $\mathcal{N}_{v_i}^{l}$ stops increasing but is less than $N - K$, or until $\widetilde{\mathcal{N}}_{v_i}^{l} \neq \varnothing$ but $\widetilde{\mathcal{N}}_{v_i}^{l+1} = \varnothing$
      - find the check nodes from $\widetilde{\mathcal{N}}_{v_i}^{l}$ that do not violate the intended check node degree distribution
      - pick the check node with the lowest degree from the candidate list
    - end
  - end
- end

Algorithm 1: PEG algorithm

Chapter 4

Simulation Results and Analysis

In this section, we present the simulation results obtained for the 16-QAM case. We begin with introducing the optimized variable node degree distributions found using the linear programming algorithm for different operating points (described by different E_s/N_0 values), for the traditional as well as for the hierarchical case. Further on, we present the code rate obtained for each case as a result of the LP optimization. We conclude with presenting bit-error ratio performances of the resulting LDPC codes as a comparison between the hierarchical and the non-hierarchical case.

The check node degree distribution was taken from [10] and fixed to be

$$\rho(x) = 0.00749x^8 + 0.99101x^9 + 0.00150x^{10} \ , \tag{4.1}$$

while the maximum degree of the variable node is $d_{v_{max}} = 30$. The length of the codeword is $N = 4096$, the proportion of bits assigned to the two modulation classes $\beta = [0.5, 0.5]$ and the number of decoding iterations chosen to be 50.

4.1 Optimized Variable Node Degree Distributions

The optimized variable node degree distributions are presented in Tables 4.1, 4.2, and 4.3 for different values of E_s/N_0.

As we can observe from these results, the variable node degree distributions are more irregular for lower values of E_s/N_0. Comparing the sub-degree distributions obtained for the resulting modulation classes $\lambda^{(1)}(x)$ and $\lambda^{(2)}(x)$, it can be easily concluded that for both the traditional and the hierarchical modulation cases the sub-distribution corresponding to the second modulation class, $\lambda^{(2)}(x)$, is more irregular.

Nevertheless, when comparing the degrees present in the overall distributions ($\boldsymbol{\lambda}(x)$) from Table 4.4 for $\alpha = 1$, $\alpha = \sqrt{2}$, $\alpha = \sqrt[4]{2}$, and $\alpha = 2$ we do not notice any sustained difference in the degree of irregularity characterizing the traditional and the hierarchical case. This is contrary to our expectations that the non-uniformity introduced by the hierarchical modulation might support the LDPC code and may lead to less irregular designs.

The 5 dB case exhibits an interesting behavior, since for any value of $\alpha > 1$, the obtained distributions are composed of variable nodes with the same degrees.

Table 4.1: Optimum variable node sub-degree distributions for the resulting modulation classes at $E_s/N_0 = 5$ dB

@ E_s/N_0	α	$\boldsymbol{\lambda}_{M_1}(x)$	$\boldsymbol{\lambda}_{M_2}(x)$
5 dB	$\alpha = 1$	$\lambda_{M_1,2} = 0.0832$ $\lambda_{M_1,3} = 0.1559$ $\lambda_{M_1,9} = 0.0361$ $\lambda_{M_1,30} = 0.3113$	$\lambda_{M_2,2} = 0.1234$ $\lambda_{M_2,4} = 0.1007$ $\lambda_{M_2,5} = 0.0020$ $\lambda_{M_2,8} = 0.1570$ $\lambda_{M_2,30} = 0.0304$
	$\alpha = \sqrt[4]{2}$	$\lambda_{M_1,2} = 0.0881$ $\lambda_{M_1,3} = 0.1598$ $\lambda_{M_1,30} = 0.3226$	$\lambda_{M_2,2} = 0.1195$ $\lambda_{M_2,3} = 0.0046$ $\lambda_{M_2,4} = 0.0979$ $\lambda_{M_2,8} = 0.1593$ $\lambda_{M_2,9} = 0.0103$ $\lambda_{M_2,30} = 0.0380$
	$\alpha = \sqrt{2}$	$\lambda_{M_1,2} = 0.0915$ $\lambda_{M_1,3} = 0.1548$ $\lambda_{M_1,30} = 0.3168$	$\lambda_{M_2,2} = 0.1168$ $\lambda_{M_2,3} = 0.0132$ $\lambda_{M_2,4} = 0.0935$ $\lambda_{M_2,8} = 0.1525$ $\lambda_{M_2,9} = 0.0082$ $\lambda_{M_2,30} = 0.0526$
	$\alpha = 2$	$\lambda_{M_1,2} = 0.0958$ $\lambda_{M_1,3} = 0.1448$ $\lambda_{M_1,30} = 0.3211$	$\lambda_{M_2,2} = 0.1132$ $\lambda_{M_2,3} = 0.0256$ $\lambda_{M_2,4} = 0.0856$ $\lambda_{M_2,8} = 0.1395$ $\lambda_{M_2,9} = 0.0060$ $\lambda_{M_2,30} = 0.0683$

Table 4.2: Optimum variable node sub-degree distributions for the resulting modulation classes at $E_s/N_0 = 7$ dB

@ E_s/N_0	α	$\boldsymbol{\lambda}_{M_1}(x)$	$\boldsymbol{\lambda}_{M_2}(x)$
7 dB	$\alpha = 1$	$\lambda_{M_1,2} = 0.1130$ $\lambda_{M_1,3} = 0.2431$	$\lambda_{M_2,2} = 0.1424$ $\lambda_{M_2,3} = 0.0424$ $\lambda_{M_2,6} = 0.2321$ $\lambda_{M_2,16} = 0.0455$ $\lambda_{M_2,17} = 0.1815$
	$\alpha = \sqrt[4]{2}$	$\lambda_{M_1,2} = 0.1181$ $\lambda_{M_1,3} = 0.2345$	$\lambda_{M_2,2} = 0.1375$ $\lambda_{M_2,3} = 0.0601$ $\lambda_{M_2,6} = 0.1737$ $\lambda_{M_2,7} = 0.0570$ $\lambda_{M_2,19} = 0.1489$ $\lambda_{M_2,20} = 0.0701$
	$\alpha = \sqrt{2}$	$\lambda_{M_1,2} = 0.1183$ $\lambda_{M_1,3} = 0.2306$	$\lambda_{M_2,2} = 0.1367$ $\lambda_{M_2,3} = 0.0611$ $\lambda_{M_2,5} = 0.0396$ $\lambda_{M_2,6} = 0.1478$ $\lambda_{M_2,12} = 0.1028$ $\lambda_{M_2,26} = 0.1122$ $\lambda_{M_2,27} = 0.0509$
	$\alpha = 2$	$\lambda_{M_1,2} = 0.1154$ $\lambda_{M_1,3} = 0.2154$ $\lambda_{M_1,30} = 0.0468$	$\lambda_{M_2,2} = 0.1358$ $\lambda_{M_2,3} = 0.0359$ $\lambda_{M_2,4} = 0.0977$ $\lambda_{M_2,8} = 0.1386$ $\lambda_{M_2,9} = 0.0300$ $\lambda_{M_2,30} = 0.1844$

Table 4.3: Optimum variable node sub-degree distributions for the resulting modulation classes at $E_s/N_0 = 9$ dB

@ E_s/N_0	α	$\boldsymbol{\lambda}_{M_1}(x)$	$\boldsymbol{\lambda}_{M_2}(x)$
9 dB	$\alpha = 1$	$\lambda_{M_1,2} = 0.3483$ $\lambda_{M_1,3} = 0.0173$	$\lambda_{M_2,3} = 0.4448$ $\lambda_{M_2,6} = 0.1897$
	$\alpha = \sqrt[4]{2}$	$\lambda_{M_1,2} = 0.3074$ $\lambda_{M_1,3} = 0.0738$	$\lambda_{M_2,2} = 0.0381$ $\lambda_{M_2,3} = 0.4007$ $\lambda_{M_2,7} = 0.1799$
	$\alpha = \sqrt{2}$	$\lambda_{M_1,2} = 0.2308$ $\lambda_{M_1,3} = 0.1784$	$\lambda_{M_2,2} = 0.1081$ $\lambda_{M_2,3} = 0.2722$ $\lambda_{M_2,7} = 0.2105$
	$\alpha = 2$	$\lambda_{M_1,2} = 0.1723$ $\lambda_{M_1,3} = 0.2347$	$\lambda_{M_2,2} = 0.1456$ $\lambda_{M_2,3} = 0.1778$ $\lambda_{M_2,8} = 0.1684$ $\lambda_{M_2,9} = 0.1012$

Table 4.4: Degrees of the overall variable node distributions

@ E_s/N_0	α	$\boldsymbol{\lambda}(x)$
5 dB	$\alpha = 1$	λ_2 λ_3 λ_4 λ_5 λ_8 λ_9 λ_{30}
	$\alpha = \sqrt[4]{2}$	λ_2 λ_3 λ_4 λ_8 λ_9 λ_{30}
	$\alpha = \sqrt{2}$	λ_2 λ_3 λ_4 λ_8 λ_9 λ_{30}
	$\alpha = 2$	λ_2 λ_3 λ_4 λ_8 λ_9 λ_{30}
7 dB	$\alpha = 1$	λ_2 λ_3 λ_6 λ_{16} λ_{17}
	$\alpha = \sqrt[4]{2}$	λ_2 λ_3 λ_6 λ_7 λ_{19} λ_{20}
	$\alpha = \sqrt{2}$	λ_2 λ_3 λ_5 λ_6 λ_{12} λ_{26} λ_{27}
	$\alpha = 2$	λ_2 λ_3 λ_4 λ_8 λ_9 λ_{30}
9 dB	$\alpha = 1$	λ_2 λ_3 λ_6
	$\alpha = \sqrt[4]{2}$	λ_2 λ_3 λ_7
	$\alpha = \sqrt{2}$	λ_2 λ_3 λ_7
	$\alpha = 2$	λ_2 λ_3 λ_8 λ_9

This behavior is also observed for the 9 dB case with $\alpha = \sqrt{2}$ and $\alpha = \sqrt[4]{2}$ which might point in the direction that for the hierarchical case the distributions tend to be composed of variable nodes with the same degrees, however, in different proportions.

4.2 Maximized Code Rate

The code rates delivered by the linear programming technique for different E_s/N_0 values and constellation parameters α are presented in Table 4.5. As already introduced above, the linear programming algorithm is based on a rate maximization criterion.

From the results presented in Table 4.5, we observe that, as expected, increasing E_s/N_0 results in higher code rates, while employing a hierarchical constellation results in lower rates when compared to the traditional case at the same E_s/N_0. One exception from this is the 5 dB case with $\alpha = \sqrt[4]{2}$ for which the rate is actually slightly bigger than the one for $\alpha = 1$ and with $\alpha = \sqrt{2}$ where the rate is the same as for the case with $\alpha = 1$. We will further analyze these results using the BER performance curves presented in the next section. We can also observe that the difference in code rates between the uniform and the non-uniform modulation increases with increasing the SNR, and, evaluated at the same SNR, with increasing α.

Table 4.5: Maximized code rate

@ E_s/N_0	α	R
5 dB	$\alpha = 1$	0.4848
	$\alpha = \sqrt[4]{2}$	0.4855
	$\alpha = \sqrt{2}$	0.4848
	$\alpha = 2$	0.4799
7 dB	$\alpha = 1$	0.5957
	$\alpha = \sqrt[4]{2}$	0.5949
	$\alpha = \sqrt{2}$	0.5912
	$\alpha = 2$	0.5758
9 dB	$\alpha = 1$	0.6909
	$\alpha = \sqrt[4]{2}$	0.6882
	$\alpha = \sqrt{2}$	0.6820
	$\alpha = 2$	0.6617

4.3 Bit-Error Ratios (BER)

This last section presents the bit-error ratio performances for two of the operating points introduced above. Since the BER values are plotted against E_b/N_0, the point at which the code was designed (defined as E_s/N_0 and given by the bending point of the curve), is not directly mapped to the plots below. From before we know that E_b/N_0 and E_s/N_0 are connected by the code rate and by the modulation rate. The code rate is readily available to us from Table 4.5 while the modulation rate, for the 16-QAM case is just $\log_2(16) = 4$.
Using the equation below, it is straightforward to find the operating point defined this time by E_b/N_0 as

$$E_s/N_0 = 4 \cdot R \cdot E_b/N_0 \ , \tag{4.2}$$

which applies for linear values of E_s/N_0 and E_b/N_0. Since we are interested to have the operating point in dB scale, we use

$$E_b/N_0 \ [\text{dB}] = 10\log_{10}\frac{10^{\frac{E_s/N_0 \ [\text{dB}]}{10}}}{4 \cdot R} \ , \tag{4.3}$$

and obtain the values presented in the table below.

Table 4.6: Mapping of the operating points from E_s/N_0 to E_b/N_0

@ E_s/N_0	α	E_b/N_0
5 dB	$\alpha = 1$	2.124 dB
	$\alpha = \sqrt[4]{2}$	2.118 dB
	$\alpha = \sqrt{2}$	2.168 dB
	$\alpha = 2$	2.168 dB
9 dB	$\alpha = 1$	4.585 dB
	$\alpha = \sqrt[4]{2}$	4.602 dB
	$\alpha = \sqrt{2}$	4.642 dB
	$\alpha = 2$	4.773 dB

Using the results from Table 4.6, we can better interpret the BER plots in Fig. 4.1 and Fig. 4.2.
We can observe that for the 5 dB operating point the curve associated with $\alpha = \sqrt[4]{2}$, which resulted in a slightly better rate than the traditional case with $\alpha = 1$, shows indeed a very close performance to it. Furthermore, the performance for the $\alpha = \sqrt{2}$ scenario is very similar to the $\alpha = 1$ case which supports the results obtained for the rate.

Overall, the performance curves at 5 dB are pretty close to each other and as a result, for the same BER requirement the difference in E_b/N_0 for the different α's is not large when compared to the 9 dB case. For the latter, the curves spread out more and the differences in performance are also more noticeable. For example, the curve for $\alpha = \sqrt{2}$ is much farther away from $\alpha = 1$ than we observed in the 5 dB case, while the $\alpha = \sqrt[4]{2}$ case follows the reference ($\alpha = 1$) closely.

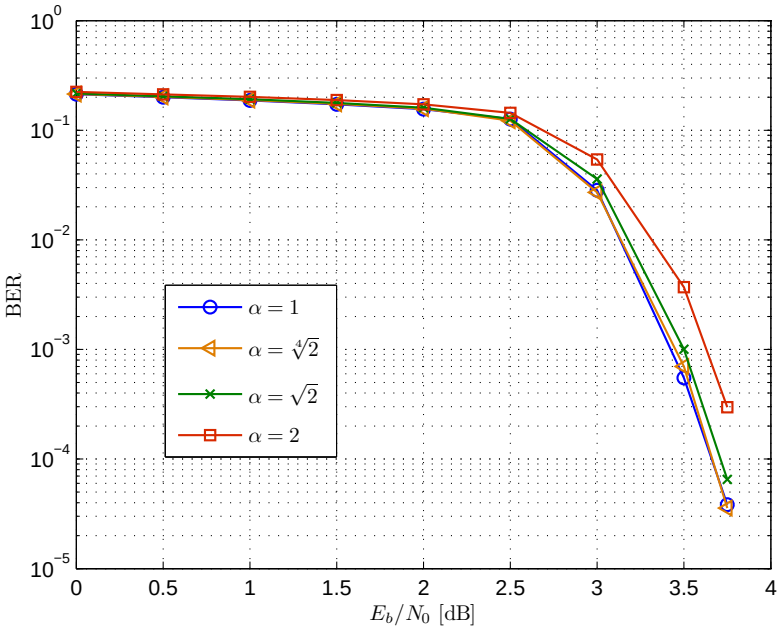

Figure 4.1: Bit-error ratio performance - $E_s/N_0 = 5$ dB

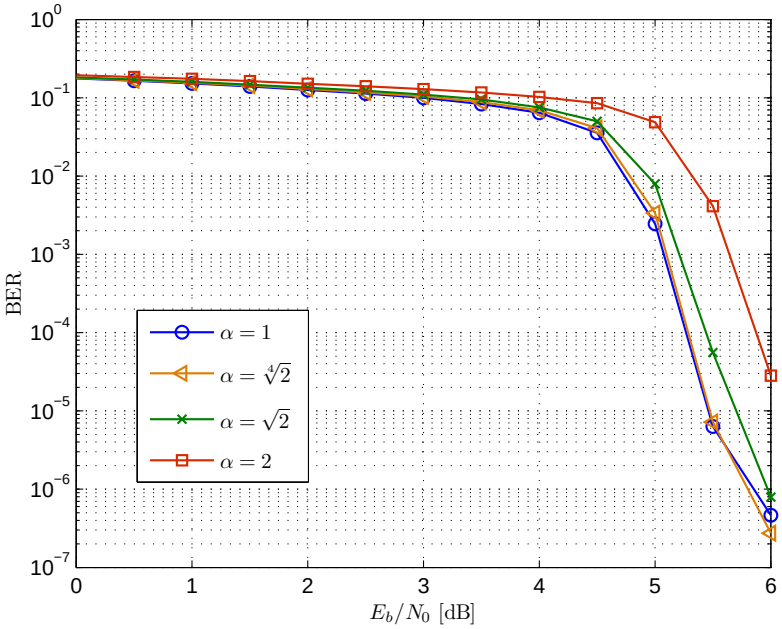

Figure 4.2: Bit-error ratio performance - $E_s/N_0 = 9$ dB

For the 9 dB case we can also observe the beginning of a flooring region for the curves at a BER around 10^{-6} which is to be expected. In order to obtain reliable values for BER when considering E_b/N_0 greater than 6 dB, the number of runs required, and implicitly the simulation time, is too big and was not considered.

In the literature, [14] presents a comprehensive analysis on hierarchical modulation which uses capacity as a suitable measure to evaluate its performance. Part of the results are obtained for a 16-QAM hierarchical modulation with $\alpha = 2$ and $\alpha = 4$[1]. From the results presented in the paper, the authors conclude that, as expected, the overall 16-QAM hierarchical modulation capacity is always less than the uniform 16-QAM one for any value of $\alpha > 1$. Nevertheless, hierarchical modulation may have practical advantages in the receiver design and the reception of high-priority information in low SNR environments.

[1] As mentioned in [14], the DVB-SH recommends these values for α.

Chapter 5

Conclusion and Future Work

The current research work investigated how an unequal error protection modulation can support the design of low-density parity-check codes. We considered a 16-QAM hierarchical constellation and proposed a linear programming technique to optimize the variable node degree distribution. We allowed for the two different modulation classes by splitting the degree distribution into two sub-degree distributions and based the optimization process on a rate maximization criterion. The resulting code offered therefore unequal error protection inside a modulation symbol. The parity-check matrix was further constructed using the progressive edge-growth construction algorithm.

By varying the constellation parameter α at different operating points we compared the obtained degree distributions and evaluated the overall BER performance. The variable node degree distributions obtained for the traditional ($\alpha = 1$) and for the hierarchical cases ($\alpha > 1$) did not confirm our understanding that, for the latter scenarios, the non-uniformities added by having different bit-error probabilities inside a symbol might support the LDPC code and result in less irregular degree distributions when compared to the uniform case. Nevertheless, we observed that for hierarchical α's the degree distributions tend to concentrate around the same degrees and that the sub-degree distribution that maps to the second modulation class is more irregular than the one characterizing the first modulation class. Another important idea is that the irregularity of the overall distributions decreases with increasing the design E_s/N_0.

The BER curves supported the results obtained for the maximum rates delivered by the LP algorithm and presented the different behaviors at 5 dB and 9 dB.

Future work for this project can focus on using a different strategy for optimizing the degree distribution and, perhaps on employing another code construction algorithm. These steps would allow for a comparison with the already obtained results. Since the check node degree distribution used for this research work was not chosen such that it represents a good match for the variable node degree distribution (due to the unknown rate which is to be maximized) but rather taken from previous research results as an optimized check node degree distribution for a rate $R = 1/2$ code, a further optimization of the check node distribution, once the rate is known, can also be considered for future work.

List of Symbols

c_i	the i^{th} check node
d_0, d_1	intersymbol distances inside a constellation
$d_{c_{max}}$	maximum check node degree
d_{v_i}	degree of variable node v_i
$d_{v_{max}}$	maximum variable node degree
E	total number of edges
E_b/N_0	signal-to-noise ratio w.r.t. bit energy
E_s/N_0	signal-to-noise ratio w.r.t. symbol energy
H	parity-check matrix
$L(x)$	log-likelihood ratio of a received value y
$L_{v_i c_j}^{(l)}$	LLR message sent from variable node i to check node j during the l^{th} iteration
$\lambda(x)$	variable node degree distribution (edge perspective)
$\widetilde{\lambda}(x)$	variable node degree distribution (node perspective)
λ_i	proportion of edges connected to variable nodes of degree i
$\widetilde{\lambda}_i$	proportion of variable nodes of degree i
Λ_i	number of variable nodes of degree i
m	mean of a Gaussian random variable
M	total number of check nodes
M_j	modulation class j
N	total number of variable nodes
N_s	number of modulation classes
$\mathcal{N}_{v_i}^l$	set of check nodes reached by a tree of depth l rooted from the variable node v_i
$\widetilde{\mathcal{N}}_{v_i}^l$	set of check nodes not reached by a tree of depth l rooted from the variable node v_i
$\mathcal{N}(m, 2m)$	Gaussian random variable with mean m and variance σ^2
R	code rate
$\rho(x)$	check node degree distribution (edge perspective)
$\widetilde{\rho}(x)$	check node degree distribution (node perspective)
ρ_i	proportion of edges connected to check nodes of degree i
$\widetilde{\rho}_i$	proportion of check nodes of degree i
σ^2	noise variance
v_i	the i^{th} variable node
x_{vc}	mutual information from a variable to a check node
x_{cv}	mutual information from a check to a variable node

List of acronyms

ACE	Approximate cycle extrinsic message degree
AWGN	Additive white Gaussian noise
BER	Bit-error ratio
BI-AWGN	Binary input additive white Gaussian noise
BP	Belief propagation
BPSK	Binary phase shift keying
DE	Density evolution
HOC	Higher order constellation
LDPC	Low-density parity-check codes
LLR	Log-likelihood ratio
LP	Linear programming
LSB	Least significant bit
ML	Maximum likelihood
MSB	Most significant bit
PEG	Progressive edge-growth
QAM	Quadrature amplitude modulation
QPSK	Quadrature phase shift keying
RV	Random variable
SNR	Signal-to-noise ratio
UEP	Unequal error protection

Bibliography

[1] R. Gallager, "Low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 8, pp. 21-28, Jan. 1962.

[2] H. Pishro-Nik, N. Rahnavard, and F. Fekri, "Nonuniform Error Correction Using Low-Density Parity-Check Codes," *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp. 2702-2714, July 2005.

[3] M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, pp. 533-547, Sept. 1981.

[4] T. J. Richardson and R. L. Urbanke, "The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599-618, Feb. 2001.

[5] S. Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Inf. Theory*, vol. 47, pp. 657-670, Feb. 2001.

[6] T. J. Richardson and R. L. Urbanke, "Fixed points and stability of density evolution," *Communications in information and systems*, vol. 4, no. 1, pp. 103-116, Sept. 2004.

[7] T. J. Richardson, M. A. Shokrolahi, and R. L. Urbanke, "Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619-637, Feb. 2001.

[8] S. Sandberg and N. von Deetzen, "Design of Bandwidth-Efficient Unequal Error Protection LDPC Codes," *IEEE Trans. Communications*, vol. 58, no. 3, pp. 802-811, March 2010.

[9] S. J. Etesami, "Secure Communication," Maters' thesis, School of Engineering and Science, Jacobs Univ. Bremen, Bremen, Germany, 2011.

[10] N. von Deetzen, "Modern Coding Schemes for Unequal Error Protection," Ph. D. dissertation, School of Engineering and Science, Jacobs Univ. Bremen, Bremen, Germany, 2009.

[11] P. V. Vitthaladevuni and M. Alouini, "BER Computation of 4/M - QAM Hierarchical Constellations," *IEEE Trans. Broadcasting*, vol. 47, no. 3, Sept. 2001.

[12] X. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and Irregular Progressive Edge-Growth Tanner Graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, Jan. 2005.

[13] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 6, pp. 533-547, Sept. 1981.

[14] H. Mric, J. Lacan, C. Amiot-Bazile, F. Arnal, and M. Boucheret, "Generic Approach for Hierarchical Modulation Performance Analysis: Application to DVB-SH," Presented at CoRR, 2011.