Hindawi Publishing Corporation
Advances in Multimedia
Volume 2010, Article ID 416797, 14 pages
doi:10.1155/2010/416797

Review Article

UEP Concepts in Modulation and Coding

Werner Henkel,[1] Khaled Hassan,[1] Neele von Deetzen,[2] Sara Sandberg,[3] Lucile Sassatelli,[4] and David Declercq[5]

[1] *School of Engineering and Science, Jacobs University Bremen, 28759 Bremen, Germany*
[2] *Silver Atena Electronic Systems Engineering GmbH, Hamburg, Germany*
[3] *Department of Computer Science and Electrical Engineering, Luleå University of Technology, 97187 Luleå, Sweden*
[4] *University of Nice, Sophia-Antipolis, BP 2135, 06103 Nice, France*
[5] *ETIS ENSEA/UCP/CNRS, 95014 Cergy-Pontoise, France*

Correspondence should be addressed to Werner Henkel, w.henkel@jacobs-university.de

Received 28 February 2010; Accepted 27 June 2010

Academic Editor: C.-C. Kuo

First unequal error protection (UEP) proposals date back to the 1960's (Masnick and Wolf; 1967), but now with the introduction of scalable video, UEP develops to a key concept for the transport of multimedia data. The paper presents an overview of some new approaches realizing UEP properties in physical transport, especially multicarrier modulation, or with LDPC and Turbo codes. For multicarrier modulation, UEP bit-loading together with hierarchical modulation is described allowing for an arbitrary number of classes, arbitrary SNR margins between the classes, and arbitrary number of bits per class. In Turbo coding, pruning, as a counterpart of puncturing is presented for flexible bit-rate adaptations, including tables with optimized pruning patterns. Bit- and/or check-irregular LDPC codes may be designed to provide UEP to its code bits. However, irregular degree distributions alone do not ensure UEP, and other necessary properties of the parity-check matrix for providing UEP are also pointed out. Pruning is also the means for constructing variable-rate LDPC codes for UEP, especially controlling the check-node profile.

1. Introduction

Source-coded data, especially from scalable video and audio codecs, come in different importance levels. Thus, data has to be protected differently. We discuss different means of achieving unequal error protection (UEP) properties on the physical level and by different coding schemes. In physical transport, we concentrate on multicarrier modulation (OFDM, DMT) presenting bit-allocation options realizing UEP properties, additionally using hierarchical modulation, as well. Modulation-oriented UEP solutions prove to be a suitable and very flexible tool to define arbitrary protection levels, if access to the actual physical transport is possible. Other options are provided by channel coding and in here, we will especially discuss Turbo and LDPC codes providing UEP. The common approach for implementing UEP properties as in standard convolutional codes would certainly be puncturing [1]. Puncturing is simply omitting some of the output bits according to some pattern, thereby changing the denominator of the rate $R = k/n$, that is, reducing the n. Since puncturing is a well-known procedure, it will not be discussed in here too much. Pruning as an alternative has not been discussed as much except for [2, 3], but would allow for changing the code rate in the opposite direction, that is, modifying k in the rate. In its easiest form, pruning would just omit certain input bits to the encoder, thereby eliminating some transitions in the trellis. Some aspects of pruning as an additional tool for UEP Turbo-code construction will be studied.

Pruning in an LDPC context would mean eliminating variable nodes in the bipartite Tanner graph setting these variables to known values, for example, zero. This will in turn modify the check degree of connected check nodes. It will serve as a tool for designing check-node degree distributions for a given UEP profile.

After some more introductory remarks on UEP for video coding in Section 2, this paper provides a *tutorial* over possible UEP realizations, starting from multicarrier modulation oriented ones in Section 3. We propose a very flexible UEP bit-loading scheme derived from a nonUEP standard method originally proposed by Chow at al. [4].

A treatment of pruned Turbo codes will follow in Section 4. Pruning and puncturing approaches are utilized to adapt the rate of a mother code in both directions.

We also study LDPC codes with an irregular variable-node profile in Section 5 and outline that the UEP properties depend on the construction algorithm delivering the final check matrix.

Finally, we consider modifications of the check-node profile of LDPC codes by pruning in Section 6. This keeps the graph of the mother code for decoding and adapts it just by known (predefined) information, which is pruning. This offers the same advantages that we know from pruning and puncturing in convolutional codes, namely, keeping the actual decoder unchanged. We conclude with Section 7.

Note that whenever error-ratio performances are shown, they will be over E_s/N_0 to be able to really observe the UEP properties with varying channel signal-to-noise ratios. For LDPC codes, however, we may still use E_b/N_0 scales, but since there is no notion of local rates, the overall rate is used, and thus these figures are actually just using renormalized E_s/N_0 scales preserving the SNR spacing of all performance curves.

2. UEP and Rate Distortion

Before we actually discuss different UEP solutions, we should deliberate shortly how we should relate source coding qualities given by spatial and temporal resolution and signal-to-noise ratio margin separations or error rates. We start referencing a work by Huang and Liang [5], who relate a distortion measure to error probabilities. However, in the end, we will conclude that for the codes that we will study in here, such a treatment is not suitable. The actual video quality steps (spatial and temporal) to be provided at what SNR steps will be at the discretion of a provider and essentially a free choice.

Huang and Liang [5] simplify the treatment by relating MPEG I, P, and B frames to protection classes with different error probabilities. This is, of course, only addressing temporal resolution. As distortion measure, the mean squared error is used and is formulated as

$$D_{\text{total}} = \sum_{i=0}^{L} \frac{S_i}{S} [E_i + A_i P_i], \tag{1}$$

where L is the number of protection classes (layers), S is the total number of bits in the source data, where S_i correspond to the numbers in the different classes. E_i is the distortion introduced by the source-coding layer itself, without considering errors added by the channel whereas $A_i P_i$ refers to the influence of channel errors. P_i is the channel bit-error rate and A_i describes the sensitivity of the ith source-coding layer to bit-errors.

For a rate-distortion relation, Huang and Liang write the total rate as

$$R_{\text{total}} = \sum_{i=0}^{L} \frac{S_i}{S} \left[R_i^{(S)} + R_i^{(C)} \right], \tag{2}$$

where $R_i^{(S)}$ and $R_i^{(C)}$ denote the source coding bit-rate and the added redundancy, respectively, for the ith layer.

This is a treatment that is reasonable for rate-compatible punctured convolutional codes [1] that will result in finite error rates. Capacity-achieving codes, however, will lead to a strong on-off characteristic due to the water-fall region in their BER curves. For such coding schemes, the SNR thresholds will define certain quality steps that will be made available to an end device. Equation (1) would then only represent the quality steps provided by the source coding, since P_i could be assumed to almost only assume the extreme values of zero and 0.5. In the case of capacity-achieving codes, it appears to be more suitable to simply relate source coding quality steps to classes and these again to SNR steps of the UEP channel coding. The SNR steps will then be either realized by different code rates of a Turbo or LDPC coding scheme or, alternatively, by bit-allocation and/or hierarchical modulation together with channel-coding with identical code rates. Combinations of modulation-based realizations of different protection levels and those based on codes with different protection levels is, of course, also possible. The quality steps provided by source coding, as well as the SNR steps provided by channel coding are then a choice of service and network providers.

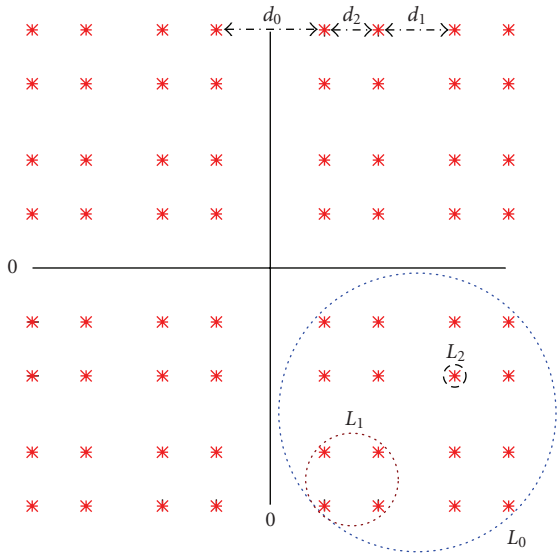
In the following, we will describe options that we investigated to realize UEP in multicarrier hierarchical modulation, Turbo-, and LDPC-coding. These schemes will prove to be very flexible, allowing the realization of arbitrary SNR level increments between quality classes.

3. Achieving UEP with Multicarrier Bit Loading

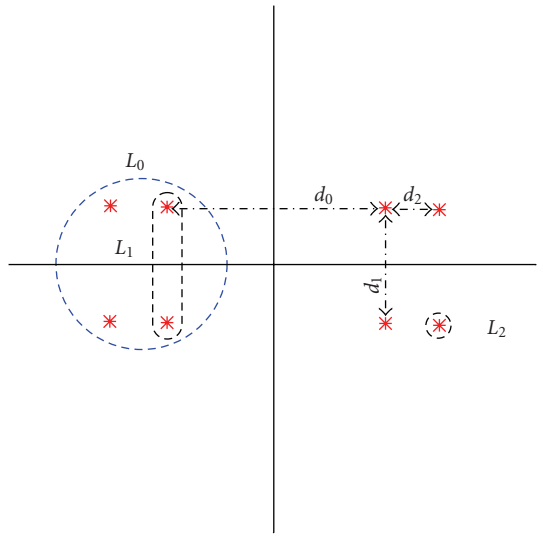
We begin our treatment with modulation-based UEP realizations, starting from hierarchical modulation without bit-loading, followed by bit-loading, to finally combine both concepts in bit-loaded hierarchical multicarrier modulation. We use different bit-loading algorithms to give a flavor of options that are possible, although space limitations will not allow to study all UEP modifications of known bit-loading algorithms.

3.1. Hierarchical Modulation. In hierarchical modulation, also known as embedded modulation [6], different symbols with unequal priorities can be embedded in each other thereby creating different Euclidean distances d_j between different priority classes j. The margin separations between these classes can easily be adjusted using the ratios of constellation distances d_j/d_i, where i and j are two different classes. There are different hierarchical constellation constructions in literature, for example, [6, 7]. However, for implementation convenience, we have selected the construction in [8] as shown in Figure 1. In this figure, we assume 3 different classes L_j, $j \in \{0, 1, 2\}$ and the performance priority ratios are assumed to be fixed to 3 dB, hence $d_0/d_1 = d_1/d_2 = \sqrt{2}$.

Figure 2 depicts the bit-error ratios in case of AWGN using a fixed hierarchical modulation 2/4/8-QAM (as defined in Figure 1(b)). Figure 2 also shows the comparison between AWGN and a Rayleigh fading channel (The channel is modeled as independent time-invariant Rayleigh fading composed of $\Lambda = 9$ different paths (echoes); each path



(a) A 4-QAM (L_0) is embedded in a 16-QAM (L_1) which is embedded in a 64-QAM (L_2)



(b) A BPSK (L_0) is embedded in a 4-QAM (L_1) which is embedded in a nonsquare 8-QAM (L_2)

Figure 1: Hierarchical quadrature amplitude modulation (QAM): (a) 4/16/64-QAM and (b) 2/4/8-QAM.

has its own amplitude β_l, delay τ_l, and random uniform phase shift $\theta_l \in [0, 2\pi)$, i.e., $h(\tau) = \sum_{l=0}^{\Lambda-1} \beta_l p_l e^{j\theta_l} \delta(\tau - \tau_l)$; p_l follows an exponentially decaying power profile [9]), where the 3-dB margin is strictly preserved in the AWGN case. However, in the case of a Rayleigh fading channel, this margin becomes wider, for example, almost 6 dB at a SER (symbol-error ratio) of $2 \cdot 10^{-3}$. Nevertheless, the order of the classes and the relative margin separations are roughly preserved. The overall system performance deteriorates due to the fixed modulation size and the fixed power allocation. Hence, further adaptation to channel conditions, using adaptive modulation and power allocation is a very important measure to keep the margin separation and an acceptable performance, as will be discussed in detail in the next section.

3.2. UEP Adaptive Modulation. Traditionally, bit-loading algorithms have been designed to assure the highest possible link quality achieving equal error probability. This results in performance degradations in case of variable channel conditions (no graceful degradation). In contrast, UEP adaptation schemes [10] allow for different parts of the same video stream/frame to acquire different link qualities. This can be done by allocating different parts of this stream to different subcarriers (with different bit-rate and error probabilities) according to the required QoS. Therefore, current research efforts [10–12] have been directed towards modifying the traditional bit-loading algorithms, for example, the ones by Hughes-Hartogs [13], Campello [14], Chow-Cioffi-Bingham [4], Fischer-Huber [15], in order to realize UEP. In [16], the algorithm by Fischer et al. has been modified in order to allow for different predefined error probabilities on different subcarriers. However, the allocation of subcarriers to the given classes is a computationally complex process. A more

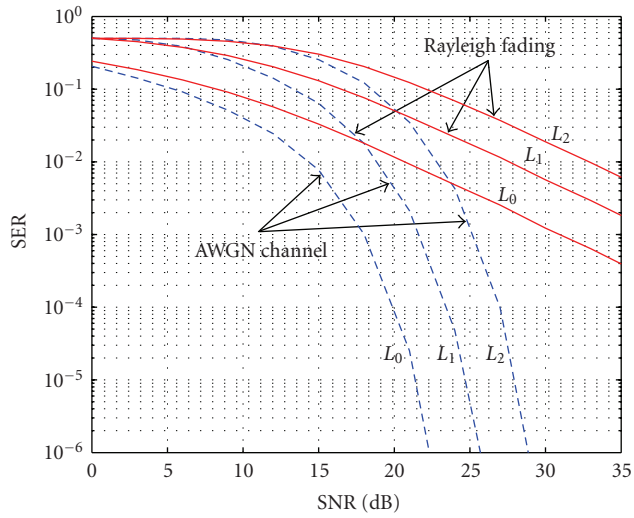


Figure 2: SER performance for 2/4/8 hierarchical QAM (defined in Figure 1(b)) assuming 3 different classes with a margin separation of 3 dB. In total, 6144 bits were placed on 2048 subcarriers.

practical approach has been described in [11] using a modified rate-adaptive Chow et al. bit-loading. This one modifies the margin γ in Shannon's capacity formula for the Gaussian channel in [4] by dedicating a different γ_j for each protection level j. The advantage here is the flexibility to adapt the modulation in order to realize any arbitrary margin separations between the priority classes. The modified UEP capacity formula is given by [17]

$$b_{k,j} = \log_2\left(1 + \frac{\mathrm{SNR}_{k,j}}{\gamma_j}\right), \tag{3}$$

where k is the carrier index. Equation (3) is rounded to

$$\hat{b}_{k,j} = \text{round}\left(b_{k,j}\right), \tag{4}$$

with quantization errors

$$\Delta b_{k,j} = b_{k,j} - \hat{b}_{k,j}. \tag{5}$$

The iterative modification of the overall γ_j (if the target bit-rate is not fulfilled) is performed in the same way as in the original Chow et al. algorithm, namely applying

$$\gamma_{0,\text{new}} = \gamma_{0,\text{old}} \cdot 2^{(B-B_T)/N_{\text{used}}}, \tag{6}$$

to one of the margins, for example, to γ_0. N_{used} is the number of actually used carriers amongst the total of N carriers (We approximated N_{used} by N in our computations), $B = \sum_{k,j} b_{k,j}$ is the total actual number of bits, B_T denotes the total target number. The margin spacing between the given L classes is selected as $\Delta\gamma$ in dB, such that

$$\gamma_j = \gamma_{j-1} - \Delta\gamma \ [\text{dB}], \tag{7}$$

where j here can take on values in $1, \ldots, L-1$ and γ_0 is computed in the iterative process [12].

As in the original algorithm, the quantization error $\Delta b_{k,j}$ is used in later fine-tuning steps to force the bit load to desired values if the iterations were not completely successful.

How should now different protection classes be mapped onto the given subcarriers? An iterative sorting and partitioning approach has be proposed in [10, 11]. The core steps of the algorithm have been simplified more in [18] using a straight-forward linear algebra approach to initialize γ_0 close to the final solution. The main steps in [11, 18] are given in the following.

(1) The N subcarriers are sorted in a descending order according to the channel state information; the sorted indices are stored in a vector M of size $1 \times N$.

(2) In [10, 11], γ_0 is initially set to an arbitrary value (as in [4]). However, in [18], γ_m of the middle priority class is calculated initially using the average SNR ($\overline{\text{SNR}}$) as $\gamma_{m_{\text{init}}} = \overline{\text{SNR}}/2^{B_T/N}$, and enhanced more using

$$\gamma_m = 2^{\left(\sum_{k=1}^{N-1} \log_2(\gamma_{m_{\text{init}}} + \text{SNR}_k) - B_T\right)/N}. \tag{8}$$

Thereafter, the noise margins of the other classes are computed according to (7).

(3) $\hat{b}_{k,j}$ is calculated as in (4); the number of subcarriers for each priority class are selected to fulfill the individual target bit-rate T_j, using a binary search, as in [11].

(4) If the target bit-rate B_T is not fulfilled, all γ_j are adjusted using (6) together with (7), subsequently repeating from (3).

(5) Else, if the maximum number of iterations is achieved without fulfilling B_T, further tuning based on the quantization error (5) is performed as in [4].

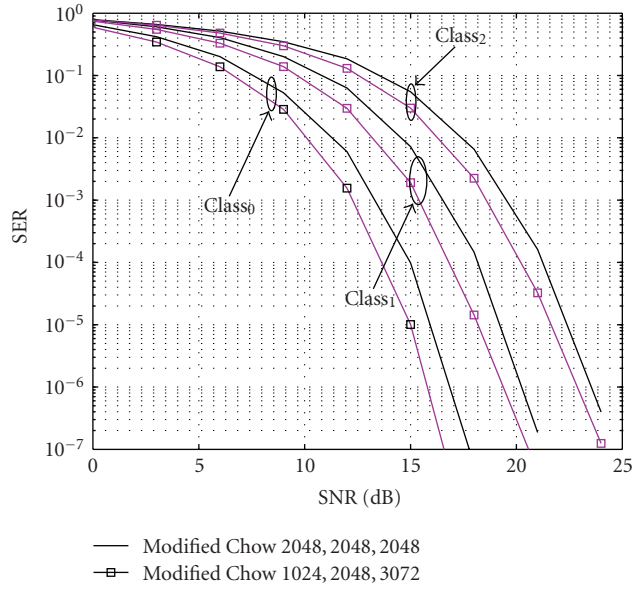


Figure 3: SER performance for the modified Chow algorithm assuming 3 different classes with margin separations of 3 dB, and 6144 bits on 2048 subcarriers with two scenarios: $T_0 = 1024$, $T_1 = 2048$, $T_2 = 3072$, and $T_j = 2048$, $j = 0.2$.

The main drawback of the previous two methods [16, 18], is the inefficient energy utilization, where energy is wasted in allocating it to weak subcarriers. The algorithm by Hughes-Hartogs [13] is seen as the energy optimum bit-loading approach, however, it requires lengthy searching and sorting steps and nonlinear operations. Campello's bit-loading [14], which is a linear representation of the Levin bit-loading algorithm [19], is a simple alternative in between Hughes-Hartogs and Chow et al.. It achieves *almost the same* optimum power allocation requiring only a fraction of the complexity due to quantization of the channel-gain-to-noise ratio $\mathcal{G}_k$ based, again, on Shannon's formula. However, carriers of similar levels of $\mathcal{G}_k$ can be gathered into G smaller groups, where $G \ll N$. Hence, all carriers in each of these groups can be adapted simultaneously. Therefore, the algorithm can easily allocate bits according to these quantized groups, later it tunes following the Hughes-Hartogs criterion of minimum power increment. In addition to the simplicity, Campello's bit-loading can be thought of as a practical solution for limited (quantized) channel feedback systems [12]. However, in this paper, we will discuss the UEP applications of the Hughes-Hartogs algorithm, only.

Figure 3 shows the performance of the modified Chow et al. algorithms assuming multicarrier modulation with 2048 subcarriers and Rayleigh fading. The performance deteriorated when adding more bits to the first class (see the scenario $T_j = 2048$). The performance of the adaptive (nonhierarchical) Rayleigh fading case (in Figure 3) exceeds the hierarchical nonadaptive AWGN (in Figure 2). This shows the inefficiency of the hierarchical modulation.

3.3. UEP Adaptive Hierarchical Modulation. For the optimal power bit-loading algorithms (like Hughes-Hartogs), we opt for hierarchical modulation to realize UEP classes [12] together with bit-allocation instead of carrier grouping, since it realizes different classes more efficiently without tedious binary searches for the carrier groups separation. In this approach, the highest priority class first consumes the good-SNR subcarriers with the minimum incremental power (calculated based on the maximum allowed symbol-error rate P_{e_0} (SER) and the channel coefficients). Thereafter, the bits of the following classes are allowed to be allocated to either already used subcarriers in hierarchical fashion if their incremental powers are the minimum ones. However, if the incremental powers are not sufficient to allocate more bits in hierarchical fashion, free subcarriers can instead be used based on the same given margin separation $\Delta\gamma_j$, which is identical to the one given by the hierarchical modulation. Therefore, the only required information to establish our algorithm now is the SER P_{e_0} of the first class. The other SER, of the less important data, P_{e_j}, are calculated using the given margin γ_j and the given P_{e_0}, as in [11, 12].

In here, we are going to describe the complete power minimization hierarchical bit-loading algorithm. This algorithm can be considered as a margin-adaptive bit-loading defined as

$$\begin{aligned} &\min_{\mathcal{E}_k} \quad \mathcal{E}_\sigma = \sum_{k=0}^{N-1} \mathcal{E}_k, \\ &\text{subject to} \quad B = \sum_{k=0}^{N-1} \log_2\left(1 + \frac{\mathcal{E}_k \mathcal{G}_k}{\Gamma}\right), \quad \mathcal{E}_\sigma \le \mathcal{E}_{\text{tot}}, \end{aligned} \tag{9}$$

where $\mathcal{E}_k$ is the power allocated to the kth subcarrier, $\mathcal{E}_{\text{tot}}$ is the given target power, $\mathcal{E}_\sigma$ is the accumulated power, $\mathcal{G}_k$ is the channel gain (λ_k) to the noise (σ_n^2) ratio, and the "gap" approximation is given by $\Gamma = (2/3)[\text{erfc}^{-1}(P_{e_j}/2)]^2$ [20]. If the total target rate is tight to a certain value B_T and $\mathcal{E}_{\text{tot}}$ is still greater than $\mathcal{E}_\sigma$, then the performance can be further enhanced by scaling up the effective power allocation $\mathcal{E}_k$ by the ratio $\mathcal{E}_{\text{tot}}/\mathcal{E}_\sigma$. This is called "margin maximization" criterion, where the maximum system margin is defined as

$$\gamma_{\max} = \frac{\mathcal{E}_{\text{tot}}}{\mathcal{E}_\sigma}. \tag{10}$$

The complete algorithm is as follows.

(1) Initially, allocate $L \times N$ zeros to the bit-loading matrix $\mathbf{B}$ and N zeros to the power-loading vector $\mathcal{E}$ and the incremental power vector $\Delta\mathcal{E}$.

(2) Set $j = 0$ and the maximum allowed number of bits on each class to $b_{j,\max}$, such that the summation over all j is less than the maximum number of bits per carrier $b_{j,\max}$.

(3) Compute the incremental power steps $\Delta\mathcal{E}_k$, for every subcarrier assuming a single bit addition, using the following approximate equation (as in [20]):

$$\Delta\mathcal{E}_k = \frac{(2/3)\left[\text{erfc}^{-1}\left(P_{e_j}/2\right)\right]^2}{\mathcal{G}_k}\left(2^{\left(\sum_{j=0}^{L-1} B_{j,k}+1\right)} - 2^{\left(\sum_{j=0}^{L-1} B_{j,k}\right)}\right), \tag{11}$$

where P_{e_j} of the current class j is calculated using the previous class probability of error $P_{e_{j-1}}$ and $\Delta\gamma$ as follows:

$$\begin{aligned} P_{e_j} \simeq &\left(1 - 2^{-\left(\sum_{i=0}^{j-1} B_{i,k}+1\right)/2}\right) \\ &\times \text{erfc}\left(\sqrt{10^{-\Delta\gamma/10}\text{erfc}^{-2}\left(\frac{P_{e_{j-1}}}{1-2^{-\left(\sum_{i=0}^{j-1} B_{i,k}\right)/2}}\right)}\right), \end{aligned} \tag{12}$$

which is valid for high constellation order since $P_{e_{j-1}} \simeq (1 - 2^{-(\sum_{i=0}^{j-1} B_{i,k})/2})\,\text{erfc}(\sqrt{3\gamma_{j-1}/2})$, as in [10], and $\gamma_{j-1} = 10^{-\Delta\gamma/10}\gamma_j$, that is, if P_{e_0} is given, the other classes P_{e_j} can be computed according to (12).

(4) Find the minimum $\Delta\mathcal{E}_k$ among all subcarriers, then increment $B_{j,k}$ such that $B_{j,k} \le b_{j,\max}$ allowed for each hierarchical level.

(5) Increment the power of this subcarrier k by the value $\Delta\mathcal{E}_i$.

(6) If the target bit-rate of the jth class is not fulfilled and

(a) the sum of the powers $\sum_{k=0}^{N-1} \mathcal{E}_k$ is less than the target energy $\mathcal{E}_{\text{tot}}$, go to (3),

(b) else, stop and go to (8) to finalize the margin maximization approach.

(7) If the target bit-rate of the jth class is fulfilled and j is less than the number of the given classes L,

(a) if the sum of the energy is less than the target energy $\mathcal{E}_{\text{tot}}$, increment j such that, $j < L$, then go to (3),

(b) else, stop the iterations for this class.

(8) Scale-up the allocated energy $\mathcal{E}_k$ using (10), then

$$\mathcal{E}_{k,\text{new}} = \mathcal{E}_{k,\text{old}} \cdot \gamma_{\max}. \tag{13}$$

The matrix $\mathbf{B}$ has L hierarchy levels as its rows. Non-allocation of leading row(s) means that first protection level data have not been put on the corresponding carrier. Nevertheless, lower-priority data may follow and still use a smaller hierarchical signal set.

Figure 4 depicts the performance of the modified Hughes-Hartogs algorithm in the case of allocating 1024 bits for the first priority class, 2048 bits for the second priority class, and 3072 bits for the least priority one. The number of subcarriers are assumed to be 2048, the same as before, and $b_{j,\max}$ of each modulation layer is 6 bits. It is clear from Figure 4 that the 3 dB spacing is

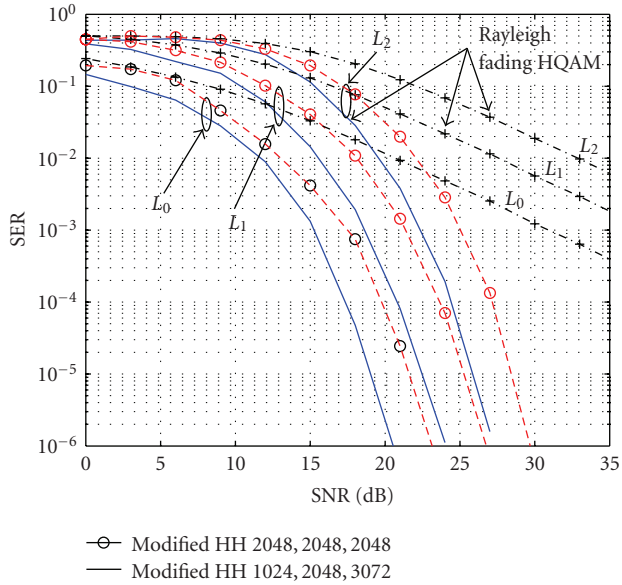


FIGURE 4: SER performance in Rayleigh fading for the modified Hughes-Hartogs algorithm with adaptive hierarchical QAM assuming 3 different classes with a margin separation of 3 dB. In total, this figure has 6144 bits on 2048 subcarriers with two scenarios: $T_0 = 1024$, $T_1 = 2048$, $T_2 = 3072$, and $T_j = 2048$, $j = 0.2$.

better preserved in the case of the Rayleigh channel in Figure 1 (without bit and power-loading). We also observe the same performance degradation as in the modified Chow algorithm, when adding more bits to the first class. Finally, one can also see from Figure 4 that the performance of the nonhierarchical modified Chow algorithm outperforms the hierarchical Hughes-Hartogs UEP, which is due to the power-inefficiency of hierarchical constellations.

An example of combining hierarchical modulation schemes with Turbo coding of different rates is given in [21]. How such different rates are obtained in a flexible way, is shown in the following section.

In this paper, we only focus on (almost) capacity-achieving codes. Turbo codes are known for their error-floor behavior, nevertheless they are suited for smaller codeword lengths, that is, interleaver sizes. If the error floor is an issue, outer Reed-Solomon codes may be applied. There are, of course, manyfold options with smaller codeword lengths or delays, such as rate-compatible convolutional codes based on puncturing, which we are to some extent addressed inside the following Turbo-code section. Just to mention another example, one may also think of multilevel coded modulation with corresponding rate choices according to the desired SNR steps [22]. Actually, also there, Turbo- and LDPC codes can be chosen for the different layers.

4. Achieving UEP with Convolutional Codes for Applications in Turbo Coding

In this section, we describe methods of achieving unequal error protection with convolutional codes which can later be applied in Turbo codes. A straightforward approach of varying the performance of a convolutional code is puncturing, that is, excluding a certain amount of code bits from transmission and, thus, increasing the code rate $R = k/n$, where k and n are the numbers of information bits and code bits. Another approach is called pruning, which modifies the number of input bits to the encoder k, that is, the numerator of the code rate instead of the denominator. In contrast to [2, 3], we present a more flexible way of pruning in the following. In order to modify the number of encoder input bits, certain positions in the input sequence could be reserved for fixed values, that is, 0 or 1 for binary codes. The code rate of a pruned convolutional code can be given as

$$R = \frac{L_p \cdot k - n_0}{L_p \cdot n}, \tag{14}$$

where n_0 denotes the number of digits fixed to a certain value and L_p is the pruning period. In (14), we assumed pruning of parities. Pruning in systematic information would also lead to subtracting n_0 in the denominator. At the receiver, the pruning pattern is known such that the reliability of the fixed zeros can be set to infinity (or equivalently, the probability can be set to 1) and may help decoding the other bits reliably.

A possible pruned input sequence to a 2-input encoder with certain positions fixed to 0 could be

$$\mathbf{u} = \begin{pmatrix} u_1 & 0 & 0 & 0 & 0 \\ u_2 & u_3 & u_4 & u_5 & u_6 \end{pmatrix}, \tag{15}$$

where the pruning period is $L_p = 5$. Thereby, code rates other than that of the mother code can easily be achieved. Using puncturing and pruning, a family of codes with different error correction capabilities may be constructed. Figure 5 shows a set of bit-error rate curves of Turbo codes using pruned and punctured recursive, systematic convolutional component codes.

When performing a computer search for a suitable pruning scheme, it is usually not sufficient to study pruning patterns alone. Additionally, it has to be ensured that at interval boundaries between blocks of different protection levels, the states at joint trellis segments are the same as already required in rate-compatible punctured convolutional codes [1]. With the improved approach shown above, this problem does automatically not arise any more since the decoder is operating on one and the same trellis, namely the mother trellis, only varying certain a-priori probabilities. Thus, trellis structures do not change at transitions between different protection intervals at all.

Concerning the minimum distance of the subcode, it is in either case greater than or equal to the minimum distance of the mother code since, as stated above, both codes can be illustrated by the same trellis. Fixing certain probabilities of a zero to be infinity means pruning those paths corresponding to a one. Either, if the minimum weight path is pruned, the minimum distance of the code is increased or if it is not pruned, the minimum distance stays the same.

The proposed technique is in a way dual to puncturing with comparable complexity. Puncturing increases the rate

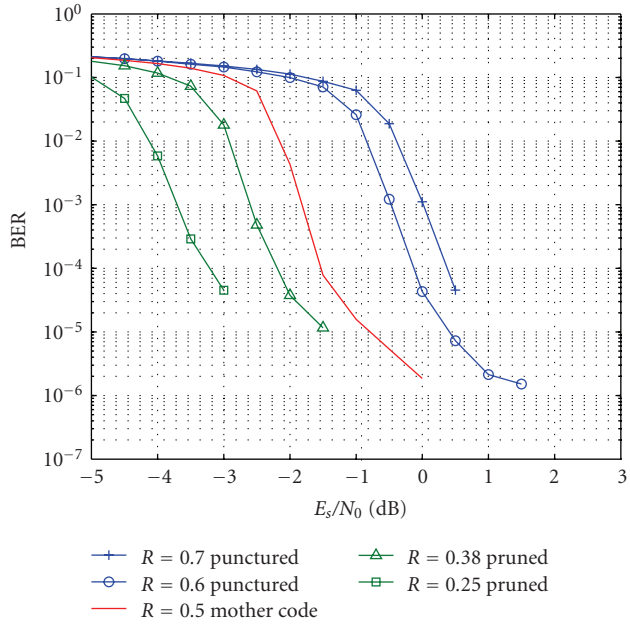


FIGURE 5: Set of bit-error rate curves of pruned and punctured Turbo codes built from RSC codes (for parameters, see the appendix).

by erasing output bits, whereas pruning reduces it by omitting input bits (fixing its value). With puncturing, there is no knowledge about the erased bits in the decoding. With pruning, we add perfect knowledge about certain bits and may enhance the decoding performance in iterative decoding through increased extrinsic information. Occasional pruning has also once been used to improve the NASA serial concatenation of convolutional and Reed-Solomon codes in [23].

We ran an exhaustive computer search in order to find mother codes together with different pruning patterns which behave well in iterative decoding. We used EXIT charts [24] for the evaluation of the convergence behavior. One assessment criterion was amongst others the convergence threshold, which is the lowest SNR where error-free decoding is theoretically possible, that is, where the tunnel between the EXIT curves opens and the mutual information between the decoded and the transmitted sequence is one (or very near to one). Furthermore, we report the area between the EXIT curves, since it is a measure of how close the waterfall region is to the Shannon limit and how steep it is [25]. Although this has formally only been proved for the binary erasure channel, it has been observed for the additive white Gaussian noise (AWGN) channel, as well. We also give the approximate distance δ of the convergence point from the Shannon limit in dB. The minimum distance of the mother convolutional codes and their pruned subcodes was determined by evaluating low-weight input sequences. Table 2 shows three convolutional mother codes with constraint lengths $L_c = 3$, $L_c = 4$, and $L_c = 5$ with reasonably fast convergence. The convolutional mother code rate is $R_{\mathrm{CC}} = 1/2$ in all cases, such that the Turbo code rate is $R_{\mathrm{TC}} = 1/3$. The pruning pattern search was performed for pruning periods up to $L_p = 6$.

The code table shows that the higher the degree of pruning (and the lower the code rate), the larger is the minimum distance. This is natural, since with a large number of constraints, it is more likely that the minimum distance path is erased.

5. Necessary Degree Distribution Properties of UEP-LDPC Codes

Irregular low-density parity-check (LDPC) codes are very suitable for UEP, as well, and can be designed appropriately according to the requirements. Irregular LDPC codes provide UEP simply by modification of the parity-check matrix and a single encoder and decoder may still be used for all bits in the codeword. The sparse parity-check matrix $\mathbf{H}$ of an LDPC code may be represented by a Tanner graph, introduced in [26], which facilitates the description of a decoding algorithm known as the message-passing algorithm [27]. Such a code may be described by variable node and check-node degree distributions defined by the polynomials [28] $\widetilde{\lambda}(x) = \sum_{i=2}^{d_{v_{\max}}} \widetilde{\lambda}_i x^{i-1}$ and $\widetilde{\rho}(x) = \sum_{i=2}^{d_{c_{\max}}} \widetilde{\rho}_i x^{i-1}$, where $d_{v_{\max}}$ and $d_{c_{\max}}$ are the maximum variable and check-node degree of the code, respectively. The coefficients of the degree distributions describe the proportion of nodes with a certain degree. Within this section, we concentrate on irregular LDPC codes, where the UEP is due to the irregularity of the variable nodes, and the check-node degrees are mostly concentrated. UEP is usually obtained by assigning important bits to high-degree variable nodes and less important bits to the lower degrees, [29–31]. Information bits may be grouped into protection classes according to their error protection requirements or importance and the parity bits are grouped into a separate protection class with least protection. Generally, the average variable node degrees of the classes are decreasing with importance. Good degree distributions are commonly computed by means of density evolution using a Gaussian approximation [32].

Based on an optimized degree distribution pair of $\widetilde{\lambda}(x)$ and $\widetilde{\rho}(x)$, a corresponding parity-check matrix may be constructed. Several construction algorithms can be found in literature. The most important ones are the random construction (avoiding only length-4 cycles between degree-2 variable nodes), the ACE (approximate cycle extrinsic message degree) algorithm [33], the PEG (progressive edge-growth) algorithm [34], and the PEG-ACE algorithm [35]. It is widely believed that an irregular variable node degree distribution is the only requirement to provide UEP, see for example [29, 30]. Surprisingly, we found that constructing parity-check matrices using these different algorithms, based on the same degree distribution pair, results in codes with very different UEP capabilities: The random and the ACE algorithms result in codes which are UEP-capable, whereas the PEG and the PEG-ACE algorithms result in codes that do not provide any UEP [36].

Since the degree distribution pairs are equal for all algorithms, a more detailed definition of the degree distribution is necessary. The multi-edge type generalization [37] may be used, but is unnecessarily detailed for our purpose.

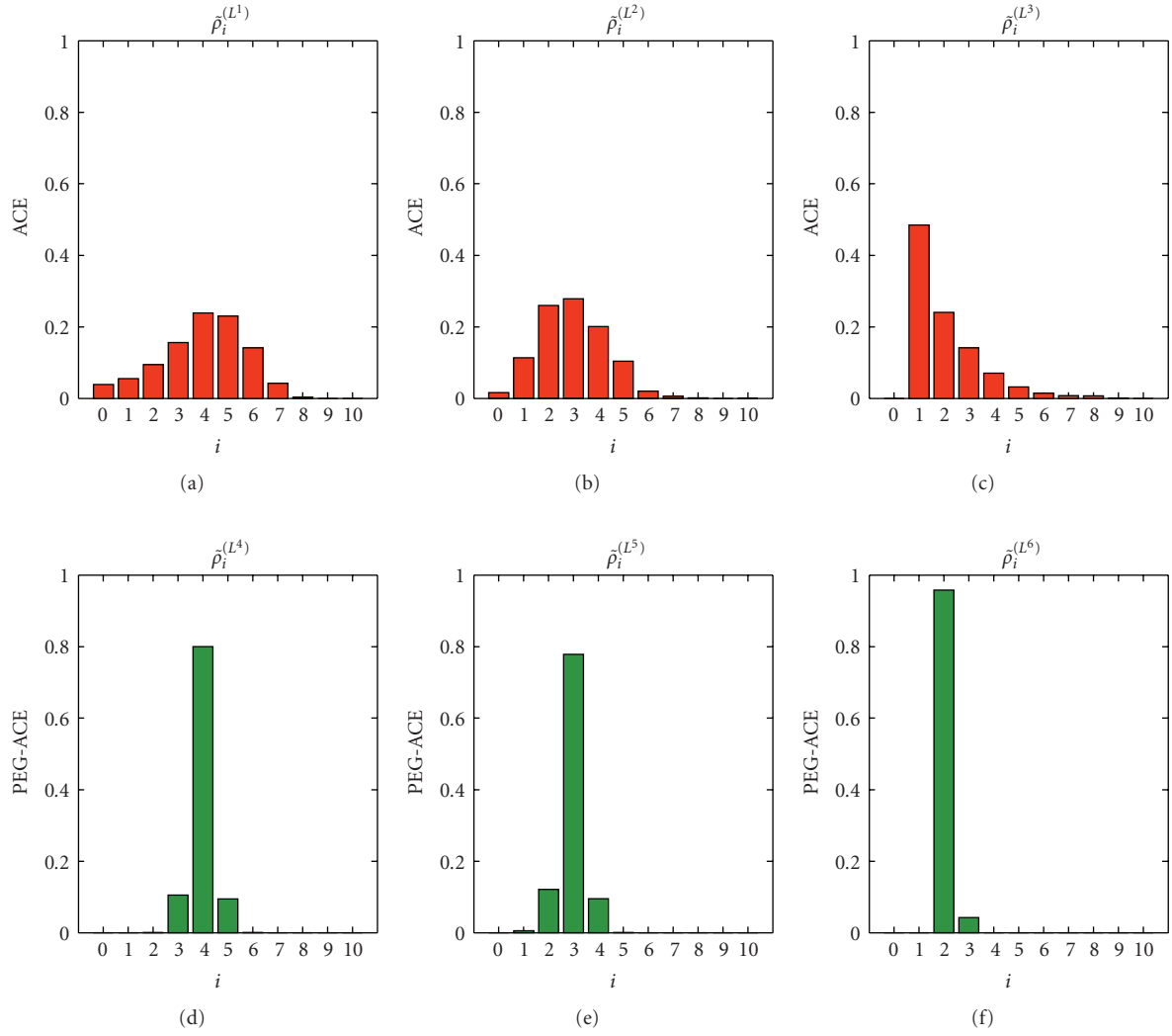


Figure 6: Detailed check-node degree distribution coefficients for the codes constructed by the ACE and the PEG-ACE algorithms.

Instead, a subclass of the multi-edge type LDPC codes is considered. Let $\tilde{\rho}^{(L^i)}(x)$ be the detailed check-node degree distribution, where the coefficients $\tilde{\rho}_j^{(L^i)}$ correspond to the fraction of check-nodes which have j edges to variable nodes in protection class L^i, regardless of the other edges.

Figure 6 shows the coefficients of the detailed check-node degree distribution for codes constructed by the ACE and the PEG-ACE algorithm and for three protection classes. The results can also be seen as histograms of the number of edges from check-nodes to the protection classes. It can be seen that the histograms corresponding to the nonUEP algorithm (PEG-ACE) are much "peakier" than those corresponding to the UEP-capable algorithm (ACE). Knowing that the overall check-node degrees are concentrated around $d_c = 9$, this means that for the PEG-ACE code, a large fraction of check-nodes has the same number of edges to the different classes, that is, most check-nodes have 4 edges to L^1, 3 edges to L^2, and 2 edges to L^3, reasoned by the different variable node degrees of the classes. In the case of the ACE code, the number of edges to different protection classes vary much more and there are many different types of check-nodes. Based on this detailed check-node degree distribution, one may perform a detailed mutual information evolution of the messages over the decoding iterations [36].

Figure 7 shows the mutual information of messages going from check-nodes to variable nodes of different protection classes (denoted by I_{appc}) as a function of the number of iterations for an ACE code and a PEG-ACE code. It is obvious that the ACE code does provide different protection levels even after the check-node update operation, while the mutual information values of the PEG-ACE code are almost identical for all protection classes. The reason is that all PEG-ACE check-nodes obtain similar values for their updates and average the UEP coming from the variable nodes (due to their different degrees). In contrast, the check-nodes of the ACE code produce different updates for different protection classes, leading to UEP even after a high number of iterations. Based on this, the resulting *a posteriori* mutual information values of the variable nodes from different protection classes (denoted by I_{appv}) are depicted in Figure 8. The figure

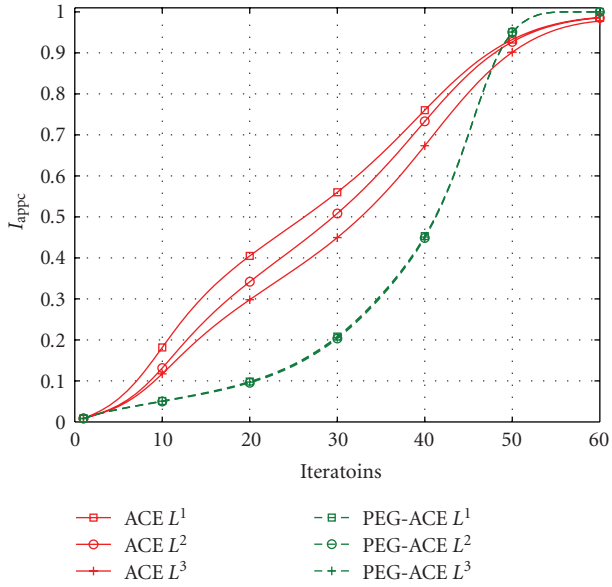


Figure 7: Check node *a posteriori* MI as a function of the number of decoder iterations at $E_b/N_0 = 0.7$ dB.

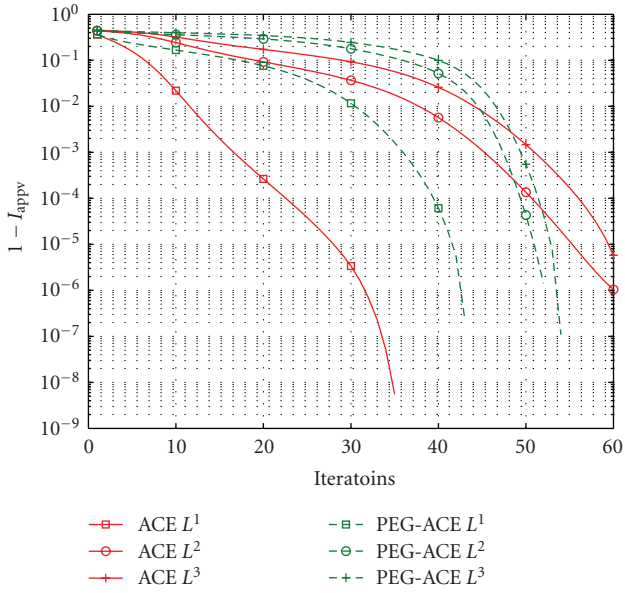


Figure 8: Distance of variable node *a posteriori* MI to the maximum MI as a function of the number of decoder iterations at $E_b/N_0 = 0.7$ dB.

shows the difference between the mutual information and its maximum value 1 on a logarithmic scale. For the UEP-capable ACE code, protection class L^1 converges much faster than the other protection classes. On the other hand, the different classes of the nonUEP PEG-ACE code have more equal convergence rates.

To confirm that the detailed check-node degree distribution is the key to the UEP capability of a code, a modification of the nonUEP PEG-ACE algorithm, which makes it UEP-capable, is presented. By constraining the edge selection

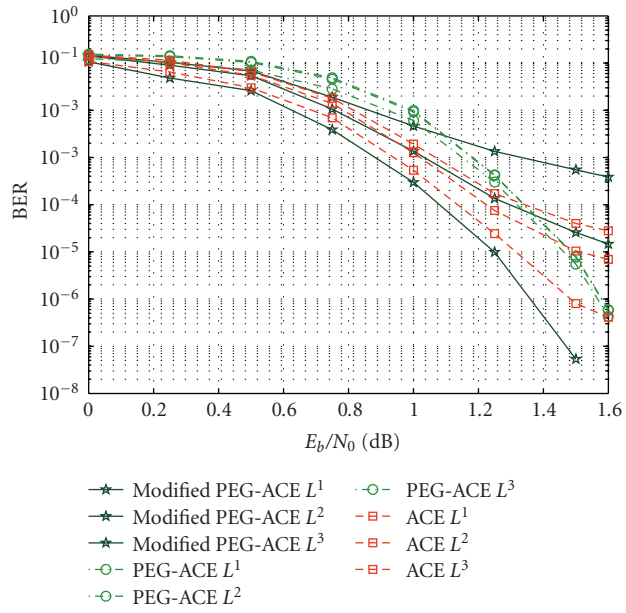


Figure 9: BER of the *modified* PEG-ACE code in comparison to the original PEG-ACE code and the ACE code.

procedure to allow only certain check-nodes to be connected, the resulting detailed check-node degree distribution is made similar to that of the ACE code. The bit-error rates of the codes constructed by the modified PEG-ACE, the original PEG-ACE and the ACE algorithm are shown in Figure 9. The figure shows that the original PEG-ACE code does not provide any UEP to its code bits, whereas the ACE code is UEP-capable. Surprisingly, the code constructed by the modified PEG-ACE algorithm offers even more UEP than the ACE code. The UEP capability provided by the modified PEG-ACE algorithm confirms that the detailed check-node degree distribution is crucial to the UEP capability of a code.

Further work is currently done along protograph constructions and and multi-edge type LDPC codes [38, 39].

6. Achieving UEP with LDPC Codes with an Irregular Check-Node Profile

In Figure 6, we observed that a noncompressed *detailed* check-node distribution was an essential ingredient to obtain UEP properties, which are even preserved after many iterations, even if an overall compressed distribution was chosen to optimize the overall average performance (according to results in [32]). In the following, we even refrain from the overall concentrated form and design UEP properties by controlling the check-node degree distribution, possibly keeping a regular variable-node degree distribution. It is well-known that the quality of a variable-node is increased with the number of edges connected to it. Regarding the check-node side, a connected variable node profits from a lower connection degree of that check-node. Thus, the quality of variable nodes is increased by lowering the (average) check-node degree of all check-nodes connected to it.

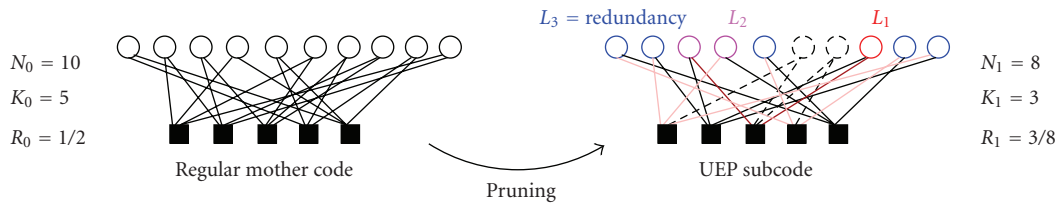


Figure 10: Pruning in the Tanner graph to exhibit UEP properties

We consider a check-node to belong to a certain bit-node (priority) class L_k if there is at least one edge of the Tanner graph connecting the check-node with one bit node of that class. By studying the mutual information at the output of a check-node of a priority class compared to the average mutual information, we get a measure of unequal protection of the priority class: the higher the difference, the more the class is protected compared to other bits in the codeword. It is also possible to link this difference in mutual information to the average check connection degree of class L_k,

$$\overline{d}^{(L_k)} = \sum_{d=d_{\min}^{(l_k)}}^{d_{\max}^{(L_k)}} d \cdot \tilde{\rho}_d^{(L_k)}, \tag{16}$$

where $d_{\min}^{(L_k)}$ and $d_{\max}^{(L_k)}$ are the minimum and maximum check connection degrees, respectively. $\tilde{\rho}_d^{(L_k)}$ is the relative portion of check-nodes with connection degree d that belong to class L_k. To maximize the performance of class L_k, $\overline{d}^{(L_k)}$ has to be minimized. In other words, the most protected classes have the lowest average check-node degrees.

Using the detailed representation of the LDPC code [40], we optimized the irregular check-node profiles for each priority class with Density Evolution. Once the irregularity profile has been optimized, there are some specific parity check matrix constructions that allow to follow the fixed profile. We depict in the following a method based on pruning, which has the advantage of being efficient and flexible, just as in the case of UEP Turbo codes in Section 4. With a single fixed (mother) encoder and decoder, the protection properties for different priority classes can be modified by suitable pruning. With pruning, we control the check-node distribution of the classes. Let (N_0, K_0) be the length and the number of information bits, respectively, of the mother code. Pruning in Section 4 meant simply omitting information bits according to some pruning pattern, that is, fixing them to some known values. Although this can be further generalized by adding a precoder to a mother code, which also offers suitable LDPC UEP solutions, we will stick to this simple pruning concept also in here. Presetting certain information to zero, means the creation of a subcode of dimension K_1 by eliminating $K_0 - K_1$ columns from the parity-check matrix $\mathbf{H}_m$. The subcode has length $N_1 = N_0 - (K_0 - K_1)$. This would be comparable to the length change in the case of pruning a systematic convolutional code. We use systematic LDPC codes, that is, LDPC codes for which the parity-check matrix has an upper triangular structure. The pruning is then performed by just omitting an information bit of the mother code, or equivalently, by removing the corresponding column in the information part of the parity check matrix (the part which is not upper triangular). By doing so, the dimensions of the subcode matrices $\mathbf{H}_S$ and $\mathbf{G}_S$ will be $M_0 \times N_0 - (K_0 - K_1)$ and $K_1 \times N_0 - (K_0 - K_1)$, respectively. The code rate is obtained as

$$R_1 = 1 - \frac{\operatorname{rank}(\mathbf{H}_S)}{N_0 - (K_0 - K_1)} = \frac{K_1}{N_0 - (K_0 - K_1)}. \tag{17}$$

Only the indices of the pruned columns of the mother code need to be known at the transmitter and the receiver in order to be able to encode and decode the pruned code. Thus, there is almost no complexity increase for realizing different UEP configurations with the same mother LDPC code. This shows that the specific matrix construction that we advice, based on a mother code and pruning, is very flexible and can be implemented in practice with low complexity.

Figure 10 illustrates the pruning in the graph of a short code. Note that the protection level is determined by the average connection degree of the check-nodes connected to the variable nodes of a certain class.

In the following, we describe the iterative pruning procedure in some more detail.

Let the relative portion of bits devoted to a class L_k be denoted by $\alpha(k)$, with $\sum_{k=1}^{L} \alpha(k) = 1$. An iterative pruning is performed. The procedure is controlled by the two key parameters of the kth class, $\overline{d}^{(L_k)}$ and $d_{\min}^{(L_k)}$. The first is the average check connection degree of the kth class defined in (16). The proportion relation

$$\sum_{k=1}^{N_C} \alpha_k \sum_{j=2}^{d_{c_{\max}}} \tilde{\rho}_j^{(L_k)} = 1, \tag{18}$$

is obtained where the second sum starts at a connection degree of 2 since a check-node should at least be connected to two variable nodes. The upper limit $d_{c_{\max}}$ is the maximum possible check degree. The protection in class L_k can be improved by minimizing the average check connection degree $\overline{d}^{(L_k)}$, which requires to minimize $d_{\min}^{(L_k)}$, as well. For each considered class, $\overline{d}^{(L_k)}$ is lowered as much as possible minimizing $d_{\min}^{(L_k)}$ step by step, too. For a chosen $d_{\min}^{(L_k)}$, one would try to put a maximum number of check-nodes with the minimum degree $d_{\min}^{(L_k)}$ in order to decrease $\overline{d}^{(L_k)}$. Although this may be interpreted to keep the degree distribution concentrated inside a certain class, this is not necessary (cf. the results in Section 5). The reduction of

Table 1: Check-node profiles of concentrated and nonconcentrated codes used in Figure 11.

(a) Check profile of the almost concentrated code

j	2	3	4	5	6
Class 1	0	0	$9.04E-01$	$9.62E-02$	0
Class 2	0	0	$6.67E-01$	$3.33E-01$	0
Class 3	0	0	$3.56E-01$	$4.86E-01$	$1.58E-01$

(b) Check profile of the unconcentrated code

j	2	3	4	5	6
Class 1	$1.59E-01$	$1.97E-01$	$3.31E-01$	$2.70E-01$	$4.29E-02$
Class 2	$1.11E-02$	$4.89E-02$	$4.07E-01$	$4.60E-01$	$7.33E-02$
Class 3	$1.33E-03$	$8.67E-03$	$1.60E-01$	$4.82E-01$	$3.48E-01$

$\overline{d}^{(L_k)}$ may be realized in different ways. However the steps and the succession in pruning are chosen, including possible reallocations of variable nodes to classes, the following constraints need to be fulfilled and checked every time.

(1) Any pruned bit must not be linked with a check-node of degree already identical to the lower limit of a priori chosen degree distributions.

(2) Unvoluntary pruning shall be avoided, meaning that a column of the parity-check matrix $\mathbf{H}$ becomes independent from all the others and then it would not define a code any more.

(3) The chosen code rate K_1/N_1 must still be achieved given by the total number of checks NK and the number of bit nodes N.

(4) Convergence at a desired signal-to-noise ratio (near the Shannoncapacity limit) must be ensured, typically by investigating EXIT charts [24].

(5) A stability constraint [32] has to be ensured, which is formulated as a rule for λ_2, which is the proportion of edges of the graph connected to bit nodes of degree 2

$$\lambda_2 \leq \frac{e^{1/2\sigma^2}}{\sum_{j=2}^{d_{c\max}} \rho_j (j-1)}, \tag{19}$$

where ρ_j denotes the proportion of edges of the graph connected to check-nodes of degree j.

In an iterative procedure, $d_{\min}^{(L_k)}$ may be further reduced after ensuring that the listed constraints are fulfilled (if the lower limit of allowed degrees is not yet reached). A further pruning process is used to reduce $\overline{d}^{(L_k)}$.

Figure 11 shows an exemplary result obtained by iterative pruning. The curves are based on a regular LDPC mother code of length $N_0 = 2000$ and a code rate of $R_0 = 1/2$. The subcode has a length of $N_1 = 1000$ and code rate $R_1 = 1/3$. The L classes to be optimized are defined by the proportions $\alpha(k)$ for $k \leq L-1$ (the number of info bits in the class L_k is $\alpha(k) \cdot R_1 \cdot N_1$ if $k \leq L-1$, and $\sum_{k=1}^{N_c-1} \alpha(k) = 1$, and $(1-R_1) \cdot N_1 = (1-R_0) \cdot N_0$ in the last one which then contains the whole redundancy). The optimization is done for $L = 3$ classes with $\alpha(1) = 0.1$, $\alpha(2) = 0.9$. The mother code has parameters (2000,3,6).

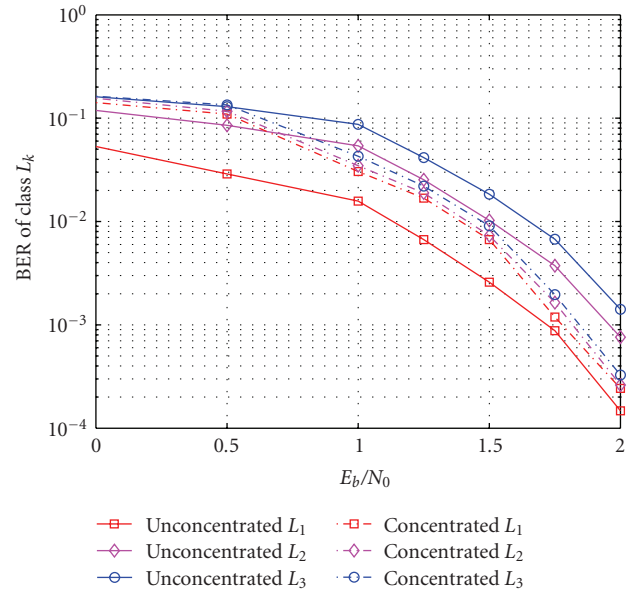


Figure 11: BERs of concentrated and nonconcentrated pruned check-irregular codes of rate 1/3 and length 1000 after 30 iterations for $\alpha(1) = 0.1$, $\alpha(2) = 0.9$.

Optimizations to obtain unconcentrated (degrees for checks between 2 and 6) and almost concentrated (degrees for checks between 4 and 6) degrees codes were performed to compare the performances.

The decoder is using the pruned parity-check matrix of the mother code. The check-node profiles are given in Table 1. The variable-node degree was three.

7. UEP in Physical Transport or in Coding?

This paper has pointed out manifold options for realizing unequal error protection, especially new concepts developed recently. UEP in multicarrier physical transport is very easy to realize and the design is very flexible allowing for

Table 2: List of good codes with constraint length $L_c = \{3, 4, 5\}$ and rate-1/2 convolutional mother code. Given are the code rates R_{CC} and R_{TC} of the corresponding convolutional and Turbo code, the pruning pattern, the SNR (dB) where the Turbo code converges, the offset δ from the Shannon limit (dB), the area between the EXIT curves, and the minimum distance of the convolutional code $d_{free,CC}$.

conv. mother code	R_{CC}	R_{TC}	pruning pattern	E_s/N_0 [dB]	δ [dB]	area	$d_{free,CC}$
$\left(1 \quad \frac{D^2}{1+D+D^2}\right)$	0.5	0.333		−3.73	1.35	0.25	4
	0.417	0.278	(0 . . . . .)	−4.24	1.84	0.255	4
	0.4	0.267	(0 . . . .)	−4.27	2.05	0.263	4
	0.333	0.222	(0 0 . . . .)	−4.76	3.12	0.258	4
	0.2	0.133	(0 0 0 . .)	−8.53	1.47	0.257	6
	0.167	0.111	(0 0 0 . 0 .)	−8.78	2.14	0.266	6
$\left(1 \quad \frac{D+D^3}{1+D+D^2}\right)$	0.5	0.333		−4.73	0.35	0.077	5
	0.417	0.278	(0 . . . . .)	−6.04	0.05	0.168	5
	0.4	0.267	(0 . . . .)	−5.87	0.46	0.123	5
	0.333	0.222	(0 0 . . . .)	−7.16	0.767	0.219	5
	0.3	0.2	(0 0 . . .)	−7	0.92	0.194	6
	0.3	0.2	(0 . 0 . .)	−7	0.92	0.183	6
	0.2	0.133	(0 0 0 . .)	−9.13	0.87	0.226	7
	0.167	0.111	(0 . 0 0 0 .)	−9.37	1.547	0.236	8
$\left(1 \quad \frac{1+D+D^4}{1+D^2}\right)$	0.5	0.333		−4.73	0.35	0.086	5
	0.417	0.278	(0 . . . . .)	−5.64	0.44	0.228	5
	0.4	0.267	(0 . . . .)	−5.07	1.26	0.2	5
	0.375	0.25	(0 . . .)	−5.2	1.63	0.201	5
	0.333	0.222	(0 0 . . . .)	−6.76	1.167	0.263	5
	0.3	0.2	(0 0 . . .)	−5.80	2.12	0.202	5
	0.25	0.167	(0 0 . .)	−6.27	2.4	0.248	8
	0.2	0.133	(0 0 0 . .)	−6.73	3.27	0.306	8
	0.167	0.111	(0 0 0 0 . .)	−7.18	3.74	0.354	9
	0.1	0.067	(0 0 0 0 .)	−8.47	5.36	0.412	11

arbitrary SNR margins. In UEP Turbo or LDPC coding, the coding scheme has to be optimized in advance, that is, a code search is necessary and the performances have to be investigated beforehand (EXIT charts, simulations). Pruning and puncturing also offer quite some flexibility in choosing the code rate, but the actual performances are only obtained after the code-design and evaluation steps. However, in digital transport without access to the physical channel, the only option is UEP coding.

When the channel changes its frequency characteristic, the margins between the priority classes will be modified in UEP bit allocation, even if a more robust SNR sorting is used. In UEP Turbo or LDPC coding, the margins will more or less be preserved due to the large interleaver.

Appendix

Parameters of Figure 5

Generator matrix of the mother code:

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & \dfrac{1}{1+D+D^2} \\ 0 & 1 & \dfrac{1+D+D^3}{1+D+D^2} \end{pmatrix}. \tag{A.1}$$

The code rates given in the figure are the ones of the Turbo code, that is, the rate-2/3 convolutional code results in a rate-1/2 Turbo code. The interleaver size was 2160.

Puncturing and pruning pattern:

$R = 0.7$ (punctured)

$$\mathbf{P}_u = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}, \tag{A.2}$$

$R = 0.6$ (punctured)

$$\mathbf{P}_u = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}, \tag{A.3}$$

$R = 0.38$ (pruned)

$$\mathbf{P}_r = \begin{pmatrix} \cdot & \cdot & 0 & \cdot & \cdot & \cdot & \cdot & 0 \end{pmatrix}, \tag{A.4}$$

$R = 0.25$ (pruned)

$$\mathbf{P}_r = \begin{pmatrix} \cdot & 0 & 0 & \cdot & 0 & \cdot & \cdot & 0 \end{pmatrix}. \tag{A.5}$$

Acknowledgments

Some of this work was part of the FP6/IST Project *M*-Pipe and was cofunded by the European Commission. Furthermore, The authors appreciate funding by the German national research foundation DFG. Some results of this paper have been prepublished at conferences [10, 17, 41] or will appear in [36]. UEP LDPC codes for higher-order modulation, which were not presented in here, have recently been published in [42]; for results on UEP multilevel codes, the reader is referred to [22].

References

[1] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC Codes) and their applications," *IEEE Transactions on Communications*, vol. 36, no. 4, pp. 389–400, 1988.

[2] C.-H. Wang and C.-C. Chao, "Path-compatible pruned convolutional (PCPC) codes: a new scheme for unequal error protection," in *Proceedings of the International Symposium on Information Theory*, Cambridge, Mass, USA, February 1998.

[3] W. Henkel and N. von Deetzen, "Path pruning for unequal error protection turbo codes," in *Proceedings of the IEEE International Zurich Seminar on Digital Communications*, pp. 142–145, Zürich, Switzerland, February 2006.

[4] P. S. Chow, J. M. Cioffi, and J. A. C. Bingham, "Practical discrete multitone transceiver loading algorithm for data transmission over spectrally shaped channels," *IEEE Transactions on Communications*, vol. 43, no. 2, pp. 773–775, 1995.

[5] C.-L. Huang and S. Liang, "Unequal error protection for MPEG-2 video transmission over wireless channels," *Signal Processing: Image Communication*, vol. 19, no. 1, pp. 67–79, 2004.

[6] P. K. Vitthaladevuni and M.-S. Alouini, "A recursive algorithm for the exact BER computation of generalized hierarchical QAM constellations," *IEEE Transactions on Information Theory*, vol. 49, no. 1, pp. 297–307, 2003.

[7] "DVB-T standard: ETS 300 744, Digital Broadcasting Systems for Television, Sound and Data Services: Framing Structure, Channel Coding and Modulation for Digital Terrestrial Television," ETSI Draft, Vol. 1.2.1, No. EN300 744, 1999–2001.

[8] B. Barmada, M. M. Ghandi, E. V. Jones, and M. Ghanbari, "Prioritized transmission of data partitioned H.264 video with hierarchical QAM," *IEEE Signal Processing Letters*, vol. 12, no. 8, pp. 577–580, 2005.

[9] P. Hoeher, "Statistical discrete-time model for the WSSUS multipath channel," *IEEE Transactions on Vehicular Technology*, vol. 41, no. 4, pp. 461–468, 1992.

[10] W. Henkel and K. Hassan, "OFDM (DMT) bit and power loading for unequal error protection," in *Proceedings of the OFDM-Workshop (InOWo '06)*, Hamburg, Germany, August 2006.

[11] K. Hassan and W. Henkel, "Unequal error protection with eigen beamforming for partial channel information MIMO-OFDM," in *Proceedings of the IEEE Sarnoff Symposium (SARNOFF '07)*, pp. 1–5, Princeton, NJ, USA, May 2007.

[12] K. Hassan and W. Henkel, "UEP with adaptive multilevel embedded modulation for MIMO-OFDM systems," in *Proceedings of the OFDM-Workshop (InOWo '08)*, vol. August 2008, Hamburg, Germany, August 2008.

[13] D. Hughes-Hartogs, "Ensemble Modem Structure for Imperfect Transmission Media," U.S. Patents, no. 4,679,227, July 1987, no. 4,731,816 March 1988, and no. 4,833,706, May 1989.

[14] J. Campello, "A practical bit loading for DMT," in *Proceedings of the IEEE International Conference on Communications (ICC '99)*, vol. 2, pp. 801–805, Vancouver, Canada, June 1999.

[15] R. F. H. Fischer and J. B. Huber, "New loading algorithm for discrete multitone transmission," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '96)*, pp. 724–728, November 1996.

[16] F. Yu and A. N. Willson Jr., "DMT transceiver loading algorithm for data transmission with unequal priority over band-limited channels," in *Proceedings of the Annual Conference on Signals, Systems, and Computers*, vol. 1, pp. 685–689, Pacific Grove, Calif, USA, October 1999.

[17] W. Henkel, N. von Deetzen, K. Hassan, L. Sassatelli, and D. Declercq, "Some UEP concepts in coding and physical transport," in *Proceedings of the IEEE Sarnoff Symposium (SARNOFF '07)*, Princeton, NJ, USA, April 2007.

[18] K. Hassan, G. Sidhu, and W. Henkel, "Multiuser MIMO-OFDMA with different QoS using a prioritized channel adaptive technique," in *Proceedings of the IEEE International Conference on Communications Workshops (ICC '09)*, Dresden, Germany, 2009.

[19] H. E. Levin, "A complete and optimal data allocation method for practical discrete multitone systems," in *Proceedings of the IEEE Global Telecommunicatins Conference (GLOBECOM '01)*, pp. 369–374, San Antonio, Tex, USA, November 2001.

[20] S. Pfletschinger, *Multicarrier modulation for broadband return channels in cable TV networks*, Ph.D. thesis, University of Stuttgart, Stuttgart, Germany, 2003.

[21] B. Barmada, M. M. Ghandi, E. V. Jones, and M. Ghanbari, "Combined turbo coding and hierarchical QAM for unequal error protection of H.264 coded video," *Signal Processing: Image Communication*, vol. 21, no. 5, pp. 390–395, 2006.

[22] N. von Deetzen and W. Henkel, "On code design for unequal error protection multilevel coding," in *Proceedings of the 7th ITG Conference on Source and Channel Coding (SCC '08)*, Ulm, Germany, January 2008.

[23] O. M. Collins and M. Hizlan, "Determinate state convolutional codes," *IEEE Transactions on Communications*, vol. 41, no. 12, pp. 1785–1794, 1993.

[24] S. T. Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Transactions on Communications*, vol. 49, no. 10, pp. 1727–1737, 2001.

[25] A. Ashikhmin, G. Kramer, and S. ten Brink, "Extrinsic information transfer functions: model and erasure channel properties," *IEEE Transactions on Information Theory*, vol. 50, no. 11, pp. 2657–2673, 2004.

[26] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. IT-27, no. 5, pp. 533–547, 1981.

[27] 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.

[28] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 619–637, 2001.

[29] C. Poulliat, D. Declercq, and I. Fijalkow, "Enhancement of unequal error protection properties of LDPC codes," *Eurasip Journal on Wireless Communications and Networking*, vol. 2007, Article ID 92659, 9 pages, 2007.

[30] N. Rahnavard, H. Pishro-Nik, and F. Fekri, "Unequal error protection using partially regular LDPC codes," *IEEE Transactions on Communications*, vol. 55, no. 3, pp. 387–391, 2007.

[31] H. Pishro-Nik, N. Rahnavard, and F. Fekri, "Nonuniform error correction using low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 51, no. 7, pp. 2702–2714, 2005.

[32] 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 Transactions on Information Theory*, vol. 47, no. 2, pp. 657–670, 2001.

[33] T. Tian, C. R. Jones, J. D. Villasenor, and R. D. Wesel, "Selective avoidance of cycles in irregular LDPC code construction," *IEEE Transactions on Communications*, vol. 52, no. 8, pp. 1242–1247, 2004.

[34] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 386–398, 2005.

[35] D. Vukobratović and V. Šenk, "Generalized ACE constrained progressive edge-growth LDPC code design," *IEEE Communications Letters*, vol. 12, no. 1, pp. 32–34, 2008.

[36] N. von Deetzen and S. Sandberg, "On the UEP capabilities of several LDPC construction algorithms," to appear in *IEEE Transactions on Communications*.

[37] T. Richardson and R. Urbanke, *Modern Coding Theory*, Cambridge University Press, Cambridge, UK, 2008.

[38] T. J. Richardson and R. L. Urbanke, "Multi-edge Type LDPC Codes," http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.106.7310&rep=rep1&type=pdf.

[39] G. Liva, S. Song, L. Lan, Y. Zhang, S. Lin, and W. Ryan, "Design of LDPC codes: a survey and new results," *Journal of Communications Software and Systems*, vol. 2, no. 2, 2006.

[40] K. Kasai, T. Shibuya, and K. Sakaniwa, "Detailedly represented irregular LDPC codes," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E86, no. 10, pp. 2435–2444, 2003.

[41] L. Sassatelli, W. Henkel, and D. Declercq, "Check-irregular LDPC codes for unequal error protection under iterative decoding," in *Proceedings of the 4th International Symposium on Turbo Codes & Related Topics in connection with the 6th International ITG Conference on Source and Channel Coding*, Munich, Germany, April 2006.

[42] S. Sandberg and N. von Deetzen, "Design of bandwidth-efficient unequal error protection LDPC codes," *IEEE Transactions on Communications*, vol. 58, no. 3, pp. 802–811, 2010.