Some UEP Concepts in Coding and Physical Transport

Werner Henkel, Neele von Deetzen, and Khaled Hassan
School of Engineering and Science
Jacobs University Bremen [1]
D-28759 Bremen, Germany
Email: {w.henkel, k.hassan, n.vondeetzen}@iu-bremen.de

Lucile Sassatelli and David Declercq
ETIS ENSEA/UCP/CNRS
F-95014 Cergy, France
Email: {sassatelli, declercq}@ensea.fr

***Abstract*— Unequal error protection is the key to future 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, a UEP bit-loading method 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 more flexible bit-rate adaptations. Pruning is also the tool that leads to a construction approach for check-irregular LDPC codes with UEP properties.**

I. Introduction

Source coded data, especially from scalable video and audio codecs, come in different importance levels. Thus, data has to be protected differently. We looked into different means of achieving unequal error protection (UEP) properties on the physical level and by different coding schemes. In physical transport, we concentrated on multicarrier modulation (OFDM, DMT), which leads to a new bit-allocation algorithm as a modification of the one by Chow, Cioffi, and Bingham [1]. This appeared to be the algorithm that allows for an easy implementation of UEP.

In Turbo coding the typical approach for implementing UEP properties as in standard convolutional codes would certainly be puncturing [4]. Puncturing is simply omitting some of the output bits according to some pattern, thereby changing the denominator of the rate $R = k/n$, i.e., reducing the n. Pruning as an alternative has not been discussed much except in [5], [6] but would allow for changing the code rate in the opposite direction, i.e., 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, e.g., zero. This will in turn modify the check degree of connected check nodes. This will serve as a tool for designing check-node degree distributions for a given UEP profile.

II. Achieving UEP with multicarrier bit loading

For non-UEP applications, there are manifold bit-allocation algorithms, e.g., the ones by Hughes-Hartogs, Campello, Chow-Cioffi-Bingham, Fischer-Huber, George-Amrani. Up to now, there has been only one proposal for UEP bit loading by Yu and Willson [2] based on the Fischer-Huber algorithm, which is quite complex and not robust in case of non-stationary channels with, e.g., impulse noise or fading. We selected the algorithm by Chow et al. [1], which is based on an iteratively modified margin γ in Shannon's capacity formula for the Gaussian channel, i.e.,

$$b_k = \log_2\left(1 + \frac{\text{SNR}_k}{\gamma}\right) \; . \tag{1}$$

The quantization error

$$\Delta b_k = b_k - \hat{b}_k \; , \tag{2}$$

when rounding to integer numbers of bits per carrier, is later used as a criterion for brute-force bit addition or reduction, when the iterative margin modifications do not yet lead to the desired target rate.

The generalization to many protection classes is almost obvious. γ is now made dependent on the protection level j, i.e., we use

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

with the quantization errors

$$\Delta b_{k,j} = b_{k,j} - \hat{b}_{k,j} \; , \tag{4}$$

instead. The iterative modification of γ_j is performed in the same way as in the original Chow et al. algorithm, namely applying

$$\gamma_{0,new} = \gamma_{0,old} \cdot 2^{\frac{B_{tot}-B_T}{N}} \tag{5}$$

to one of the margins, e.g., to γ_0. $B_{tot} = \sum_{k,j} b_{k,j}$ is the total actual number of bits, B_T denotes the total target number. The spacing between the margins is kept constant. This ensures a constant SNR separation between all the classes, whereas the absolute value is allowed to change.

The question arises how the classes should be mapped onto the carriers, i.e., which carriers should belong to which protection class. In principle, there are two sorting approaches:

- use the carriers with the highest signal-to-noise ratio for the most important class,
- use the carriers with the lowest signal-to-noise ratio for the most important class.

The first would be very intuitive, but fails under non-stationary disturbances. Nevertheless, the first is more efficient in the stationary case. Note that under non-stationary disturbances not taken into account during the bit-loading process, carriers with previously high SNR may experience severe degradation, whereas those with previously low SNR may not see much of a difference. Thus, the second choice is more robust against non-stationary disturbances. This results in extremely low usage of bad-SNR carriers, allocating them with just a few bits or not at all. This is, of course, very inefficient in the stationary case. Those carriers could still be used for the less important data. This leads to a possible third scheme, representing a compromise between both extremes. One may go initially for the second, more robust approach, filling up unused carriers in a second step with less important data. One could even allow for mixed allocation of different priority data on the same carrier, realizing that the quantization error determined before according to (4) allows for an additional placement of lower-priority data. This leads to hierarchical modulation schemes similar to the ones proposed in the DVB-T standards.

For both ordering schemes, we require an SNR sorting, i.e., the carriers need to be ordered according to their SNRs. In the more robust second alternative, we will usually start allocating the highest priority data to the lowest SNR carriers, moving the carrier number limit to the following classes in, e.g., a binary search until the desired rate is obtained, then the next class follows. We allow for a slight deviation in the lower-priority rate due to the discrete alphabets (integer number of bits per carrier) loaded onto the carriers. This will, however, only move some data bits into the better protected neighboring classes.

In Fig. 1, we show the performance curves of three classes originally spaced by 3 dB under the effect of non-stationary noise. For this example we used DMT transmission with almost ADSL2plus parameters and a NEXT environment with E1 and HDSL disturbers. The additional non-stationary noise was derived from real measured impulses. We observe that the ‘robust’ UEP bit loading with SNR sorting (high-priority class at low SNR carriers) still shows the BER curves in the desired sequence, whereas the more ‘intuitive’ bit allocation (high-priority class at high SNR carriers) turns the sequence upside down, i.e., suddenly, the most important data receives the lowest protection.

III. 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

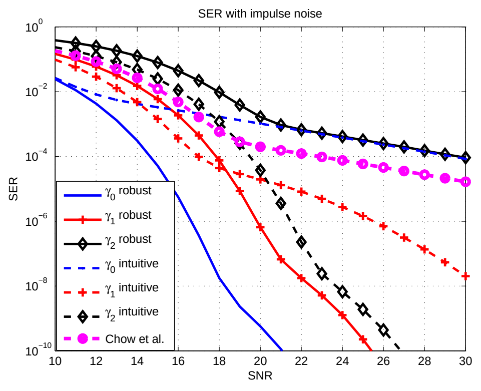

Fig. 1. The SER performance for UEP bit allocation with SNR-sorting, without SNR-sorting, and the non-UEP performance for a total target bit rate of 2304 bits per DMT symbol with 511 data carriers with real measured impulse noise as an additional disturber (bit allocation with T1/HDSL NEXT, only)

the performance of a convolutional code is puncturing, i.e., 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 method called pruning was presented in [5] and [6] for convolutional codes and Turbo codes, respectively. Pruning is based on modifying the number of input bits to the encoder k, i.e., the numerator of the code rate instead of the denominator. This approach will be revised and improved in the following.

A. Pruning - Conventional Approach

In [6], the authors presented a method called pruning for achieving low complexity variable rate convolutional and Turbo codes. The aim was to find a code family consisting of codes with code rates $R_i = k_i/n$. In order to keep the decoding complexity low, it was desired that the individual codes should have certain properties such that decoding can be performed by similar decoders. This requirement led to the construction of several sub-codes from a given mother code. The sub-codes of rates $R_{s,i} = k_{s,i}/n$, $1 \leq k_{s,i} < k_m$ were constructed by multiplying the polynomial generator matrix of the mother code of rate $R_m = k_m/n$, denoted by $G_m(D)$, by another polynomial generator matrix $G_p(D)$ called pruning matrix.

$$G_s(D) = G_p(D) \cdot G_m(D) \tag{6}$$

Choosing different pruning matrices and especially varying its dimensions, leads to different sub-codes $G_s(D)$ of the mother code. The dimensions of the pruning matrix are defined as $[k_p \times k_m]$ in order to guarantee proper matrix multiplication, leading to a sub-code generator matrix of dimensions $[k_p \times n]$. Thus, varying the number of rows of the pruning matrix varies the code rate of the sub-code. For a mother code, one can obviously construct sub-codes with $k_m - 1$ different code rates $1/n, \ldots, (k_m - 1)/n$. Given that the number of memory elements of the sub-encoder m_s is equal to (or smaller than) the number of memory elements in th mother encoder, the influence of building a sub-code from a mother code can

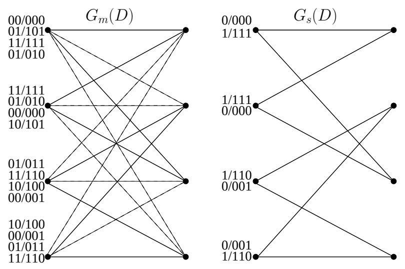

Fig. 2. Trellis sections of mother code (left) and sub-code (right)

directly be depicted in the trellis diagram of the convolutional codes. Figure 2 shows a trellis section of a mother code (left) and a sub-code (right) given by the following generator matrices.

$$\begin{aligned} G_s(D) &= G_p(D) \cdot G_m(D) \qquad (7) \\ &= (1 \;\; D) \cdot \begin{pmatrix} 1 & 1+D & 1+D \\ 1+D & D & 1+D \end{pmatrix} \\ &= \begin{pmatrix} 1+D+D^2 & 1+D+D^2 & 1+D^2 \end{pmatrix} \end{aligned}$$

The labels to the left of the states denote the input and output of the transitions leaving the states.

When comparing the two trellis sections, one can see that the set of paths in the right trellis is a subset of the set of paths in the mother trellis. This means that one does not have to store a separate decoder for all codes of a family but the decoder structure stays basically the same. When switching between different decoders, only the transition probabilities between the states have to be adapted. With this procedure, we are able to construct a code family consisting of codes with k_m different code rates.

B. Pruning - Improved Approach

In this section, we improve the concept of pruning for convolutional and Turbo codes. A drawback of pruning is the low number of available code rates in a code family, especially for small mother codes, i.e., mother codes with small k_m. Particularly, the number of available code rates is dependent on the dimensionality of the mother code. For example, a mother code of rate $R_m = 2/3$ only allows for one sub-code rate, namely $R_s = 1/3$. For applications like multimedia, where unequal error protection is desired, usually more levels of protection are required. This calls for an approach, where the number of available code rates is independent of the code itself.
This leads to the idea of time variant pruning. The corresponding measure in the field of puncturing is employing puncturing matrices defining a pattern of successive puncturing rules. The same can now be done for pruning, such that for each time instant, a sub-code is chosen according to some pattern. Let us define the pruning period L_p to be the length of such a pattern which is repeatedly applied. Let the code applied at time instant l be denoted by $G_l(D) \in \{G_m(D), G_{s,i}(D)\}$, $1 \le i \le k_m - 1$, $0 \le l \le L_p - 1$ and k_l be the number of encoder input bits corresponding to $G_l(D)$. The overall code rate of the scheme is then defined by

$$R = \frac{1}{L_p \cdot n} \sum_{l=0}^{L_p - 1} k_l \; , \qquad (8)$$

which is bounded by $1/n \le R \le k_m/n$.
Imagine a rate $R = 2/3$ mother code. A very easy case of such an above proposed pruning matrix can be defined by choosing one component of the pruning matrix to be 0 and the other one to 1. This would select only one of the inputs as active in order to construct a sub-code. Switching between the mother code and the sub-code then means switching between code rates $R = 1/3$ and $R = 2/3$. This procedure can aso be described as simply 'switching off' one of the inputs of the mother encoder according to the pruning pattern. Switching off one input is equivalent to feeding the corresponding input with zeros. Consequently, one does not even have to switch between the two encoders but only has to feed the mother encoder with zeros where the pruning pattern specifies to use the sub-code. The code rate can now be given as

$$R = \frac{L_p \cdot k - n_0}{L_p \cdot n} \; , \qquad (9)$$

where n_0 denotes the number of digits fixed to 0. Thereby, we have found a very simple way to adapt the overall code rate of the scheme by just fixing some encoder input digits.

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 of a 0 can be set to 1) and may help decoding the other bits reliably.

In the case of systematic convolutional encoders, switching off a certain input also fixes the corresponding output to be 0. In this case, the denominator of the code rate is also modified and leads to the overall code rate

$$R = \frac{L_p \cdot k - n_0}{L_p \cdot n - n_0} \; . \qquad (10)$$

One possible problem of the above proposed scheme might arise, when the number of fixed zeros is not negligible compared to the overall number of uncoded data. Especially for Turbo codes, the information bits are assumed to be equally distributed since the Turbo decoder suffers from statistical dependencies otherwise. If many zeros are fixed in the uncoded sequence, the probabilities of occurrence of a 0 and a 1 are not equal any more.
A possible solution for this problem is to not only insert zeros at the specified positions but also ones. This can be done in a predefined manner such that the receiver knows where zeros and ones are fixed. A very straightforward manner would be to choose the fixed digits to be 0 and 1 in an alternating fashion. This would directly preserve the equal distribution of zeros and ones.

In the following, we will give a small example in order to illustrate the proposed schemes. We will both show the

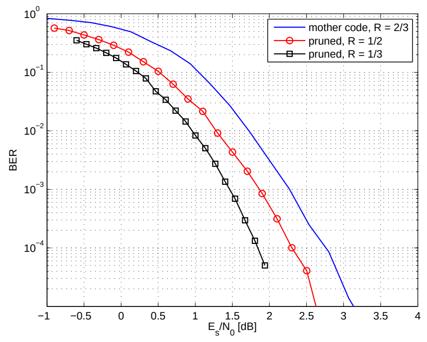

Fig. 3. Bit-error rate curves of a mother code and two sub-codes of rates $R = 2/3$, $R = 1/2$, and $R = 1/3$

insertion of fixed zeros only and of fixed zeros and ones. Assume an application that requires a code rate of $R = 0.4$ and the available codes are a mother code of rate $R_m = 2/3$ and a sub-code of rate $R_s = 1/3$. A possible pruning pattern would specify the following succession of employed codes leading to the desired code rate: $G_m(D)\ G_s(D)\ G_s(D)\ G_s(D)\ G_s(D)$. This sequence would encode 6 input bits and output 15 code bits, thus $R = 0.4$. When employing the mother code only and fixing the specified digits to zeros, the corresponding input sequence would be

$$\mathbf{u} = \begin{pmatrix} u_1 & 0 & 0 & 0 & 0 \\ u_2 & u_3 & u_4 & u_5 & u_6 \end{pmatrix}. \tag{11}$$

When inserting not only zeros but zeros and ones in an alternating fashion, the encoder input sequence is given by

$$\mathbf{u} = \begin{pmatrix} u_1 & 0 & 1 & 0 & 1 \\ u_2 & u_3 & u_4 & u_5 & u_6 \end{pmatrix}. \tag{12}$$

These small examples show how a code rate other than that of the mother code or one of the sub-codes can easily be achieved by the proposed method. Figure 3 shows a set of bit-error rate curves of a rate $R = 2/3$ mother code and two sub-codes of rates $R = 1/2$ and $R = 1/3$ constructed by fixing appropriate amounts of information bits to zero.

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, the states at joint trellis segments are the same as already required in rate-compatible punctured convolutional codes [4]. 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 sub-code, 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 an alternative to puncturing with comparable complexity and, as well, theoretically infinite range of achievable code rates. However, we see an advantage of pruning over puncturing when dealing with Turbo schemes. Whith puncturing, bits from the transmitted sequence are erased, such that there's now knowledge about them any more. When pruning, we add perfect knowledge about certain bits and may enhance the decoding performance in iterative decoding through increased extrinsic information.

IV. Achieving UEP with LDPC codes with an irregular check-node profile

UEP properties of LDPC codes have been studied in the literature, capitalizing on the unequal speed of convergence of the iterative Belief Propagation decoder. When a certain variable node has very high connection degree in the Tanner graph of the code, it collects a lot of information in a small number of iterations, and is thereby very well protected against the additive noise - some authors name bits with high connection *elite* bits. Generalizing this behavior to design specific UEP properties is feasible, although with growing number of iterations, the difference of protection between the bits diminished. However, the UEP that comes from irregular connection profile can be also solved on the other side of the Tanner graph, that is, with an irregular check node connection profile.

We consider a check node to belong to a certain bit-node (priority) class C_k if it 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 C_k,

$$\overline{\rho}^{(C_k)} = \sum_{d=d_{min}^{(C_k)}}^{d_{max}^{(C_k)}} \rho^{(C_k)}(d)d$$

To maximize the UEP property of class C_k, $\overline{\rho}^{(C_k)}$ has to be minimized. In other words, the most protected classes have the lowest average check-node degrees.

Using a detailed representation of the LDPC code [8], we have 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, and we depict in the following a method based on the pruning approach, which has the advantage of being efficient and flexible. We use the pruning approach discussed in Section III also to 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 III

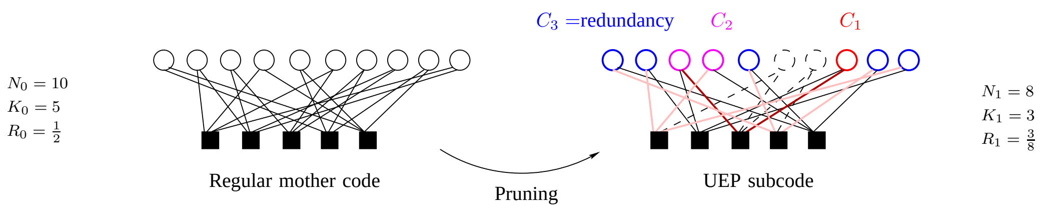

Fig. 4. Pruning in the Tanner graph to exhibit UEP properties

meant either applying a pre-coder to reduce the number of information dimensions or simply to omit information bits according to some pruning pattern. We do not discuss the first option for LDPC codes in here. The latter means the elimination of a bit node and replacing it by a known value, e.g., by zero. A subcode of dimension K_1 is obtained by eliminating $K_0 - K_1$ columns from the parity-check matrix $\mathbf{H}_m$. The subcode has length $N_1 = N_0 - (K_0 - K_1)$. Thus, the length is reduced in a similar way as in 10 for the case of systematic convolutional encoders when pruning the information bits. 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

$$R_1 = 1 - \frac{\text{rank}(\mathbf{H}_S)}{N_0 - (K_0 - K_1)} = \frac{K_1}{N_0 - (K_0 - K_1)} . \qquad (13)$$

Only the indices of the pruned columns of the mother code need to be known at the transmitter and 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 the pruning, is very flexible and can be implemented in practice with low complexity.

Figure 4 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.

V. UEP in Physical Transport or in Coding?

This paper has pointed out 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 arbitrary SNR margins. In UEP Turbo or LDPC coding, the coding scheme has to be optimized in advance, i.e., 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 (correlation properties for the equivalent binary channel), the margins between the priority classes will be modified in UEP bit allocation, even if the 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.

Acknowledgment

Some of this work is part of the FP6 / IST project M-Pipe and is co-funded by the European Commission.

Previous conference publications of the authors are [3] on UEP bit loading, [6] on pruning, and [7] on UEP LDPC codes.

References

[1] Chow, P.S., Cioffi, J.M., Bingham, J.A.C., "A practical discrete multitone transceiver loading algorithm for data transmission over spectrally shaped channels," *IEEE Transactions on Communications*, Vol. 43, No. 234, Feb-Mar-Apr 1995, pp. 773 - 775.

[2] Yu, F., Willson, A., "A DMT transceiver loading algorithm for data transmission with unequal priority over band-limited channels," proc. *Signals, Systems, and Computers, 1999*, Pacific Grove, CA, USA, Vol. 1, Oct. 24-27, 1999, pp. 685 - 689.

[3] Henkel, W. and Hassan, K., "OFDM (DMT) Bit and Power Loading for Unequal Error Protection," *OFDM-Workshop 2006*, Hamburg, Aug. 30 - 31, 2006.

[4] Hagenauer, J.: "Rate-Compatible Punctured Convolutional Codes (RCPC Codes) and their Applications", *IEEE Trans. on Comm.*, Vol. 36, No. 4, April 1988, S. 389-400.

[5] Wang, C.-H., Chao, C.-C., "Path-Compatible Pruned Convolutional (PCPC) Codes: A New Scheme for Unequal Error Protection", *International Symposium on Information Theory 1998*, Cambridge, MA, USA, Feb.1998.

[6] Henkel, W., von Deetzen, N., "Path Pruning for Unequal Error Protection Turbo Codes," *International Zürich Seminar on Communications 2006*, Zürich, Switzerland, Feb. 2006.

[7] Sassatelli, L. , Henkel W., and Declercq, D., "Check-Irregular LDPC Codes for Unequal Error Protection under Iterative Decoding," *4th International Symposium on Turbo Codes & Related Topics in connection with the 6th International ITG-Conference on Source and Channel Coding*, Munich, April 4-7, 2006.

[8] K. Kasai, T. Shibuya and K.Sakaniwa, "Detailedly Represented Irregular LDPC Codes," *IEICE Trans. Fundamentals*, Vol. E86-A(10) pp. 2435-2443, Oct. 2003.