Leaking Interleavers for UEP Turbo Codes

Abdul Wakeel, David Kronmueller, Werner Henkel, and Humberto Beltrão Neto
EECS, Transmission Systems Group (TrSys)
Jacobs University Bremen
Campus Ring 1
D-28759 Bremen, Germany
Email: {a.wakeel, w.henkel}@jacobs-university.de

***Abstract*—Interleaving is a key to Turbo coding's exceptional performance. An interleaver provides bit-permutation designed to ensure deterministic randomness. When applying interleavers to unequal error protecting (UEP) Turbo codes, typically, priority classes protected with different rates are kept separate. This work, however, gives insight, how the performance changes, when block boundaries are relaxed, leaking bits to other classes. The effect of this leakage is an improvement of the average performance due to a virtually bigger interleaver size, with the drawback of a flooring of the individual performances to the worst protection (highest code rate) involved in a certain class.**

Index Terms: UEP, Unequal Error Protection, Turbo code, interleaver.

I. Introduction

In 1993, Berrou et al. introduced the concept of *Turbo coding*, achieving error correction characteristics close to the Shannon-limit at a BER of 10^{-5} (at E_b/N_0 0.7 dB and $R = 1/2$) [1]. Together with well-known concepts of puncturing [2] and pruning [3]–[5], unequal error protecting Turbo codes can be designed. However, these unequal error protecting (UEP) Turbo codes, where rates could easily be switched by choosing puncturing or pruning patterns, were thought of using separated interleavers for each priority class, not to mix up priority classes. Typically, so-called *Semi-Random* (S-random) interleavers [6] are used, ensuring a certain minimum spacing of interleaved bits. When separating bits from different classes into separate interleavers, given a certain delay, the individual interleavers may become too small, deteriorating Turbo performance. Allowing for some leakage between the classes by *relaxing* the interleaver boundaries creates a virtually bigger overall interleaver with possibly better average performance. However, drawbacks for the error performances of the individual classes have to be expected. In this work, we will outline the effects on the average and individual performances.

The paper is structured as follows. Section II presents an overview of random and S-random interleavers. Our approach is based on S-random interleavers with a slight modification in the algorithm. In Section III, an overview of the problem statement as well as an analysis of the *relaxed* block S-Random interleaver concept is provided. Section IV shows the BER performance of a Turbo code using a relaxed block interleaver as compared to the Turbo code with S-random interleaver. Section V concludes the paper.

II. Random and Fixed Block S-Random Interleaver

Interleaving is a process of permuting the order of a data sequence in a one-to-one deterministic format in order to create statistical independence between disturbances on parity symbols related to the same information symbol entering the encoders of a Turbo code. Thus, an interleaver π of size N defines the permutation $i \rightarrow \pi(i)$. The interleaving may be defined by a permutation matrix $\mathbf{P}$ with a single one in each row and column, hence the permuted sequence is $\mathbf{d} \cdot \mathbf{P}$ ($\mathbf{d}$ being the input data sequence). The inverse of this process is the de-interleaver π^{-1} which restores the received sequence to its original order, defined as $\mathbf{P}^T$ [7].

A random interleaver is simply a random permutation π [7]. An S-random interleaver is also a random interleaver with an extra constraint. The S-random interleaver or *Fixed Block S-Random interleaver* that we implemented is the conventional Turbo code interleaver, restricted to acting on a block of length N. A slightly modified version will generate the relaxed block interleaver. The general algorithm [6] for the S-random interleaver is as follows:

1) Given N integers, randomly select one out of the integer pool without replacement.
2) Check if integer is outside the range $\pm S$ of S past values. If outside of the range, keep the value, else, reject it and place it back into the integer pool.
3) Repeat the above two steps until no integers in range 1 to N are left unused.

The theoretical S value limit which allows the algorithm to converge is $S < \sqrt{N/2}$. However, as the interleaver size increases the value of S may be significantly lower than the theoretical value of $\sqrt{N/2}$ [8]. The S constraint states that any two integers i and j which initially satisfy $|i - j| < S$, satisfy $|\pi(i) - \pi(j)| > S$ after interleaving [7]. Thus, as S grows, the random process of finding a number satisfying these criteria becomes more difficult, and thus the algorithm may not converge within suitable time (or not at all).

The S-random algorithm has proven to be well suited for the iterative decoding requirement of Turbo codes [9]. A random interleaver performs well for large interleaver sizes [7]. However, typically Turbo codes may be preferred for

shorter block sizes, even more so in the UEP case. ($S = 1$ will make an S-random to act as a random interleaver.)

III. Introducing leakage in interleaving

The performance of Turbo codes deteriorates with decreasing interleaver size. Moreover, in unequal error protection (UEP) Turbo coding the information block is divided into several blocks of different importance levels according to error correction requirements, with each block encoded with its own code and each possibly having its own interleaver, thereby avoiding a mix of bits of different priority classes. This splitting may lead to small interleaver sizes again resulting in unsatisfactory performances.

These individual interleavers may also be combined into one overall interleaver represented by a block-diagonal matrix, with entries randomly distributed within the block and each block being encoded at its own specific code rate according to its priority. We now preserve the code rate, but relax a certain fraction of bits into other (especially neighboring) blocks of different protection. The block-diagonal matrix structure of the interleaver matrix now becomes a random matrix, where depending on the fraction of external bits, a block-diagonal structure can still be recognized, with entries outside the block-diagonals as shown in Fig. 1.

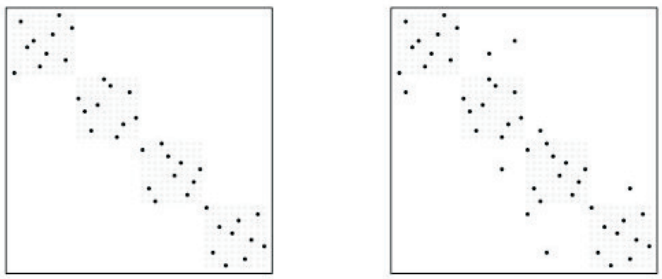

Fig. 1. Overall interleaver as a block-diagonal matrix (a) without and (b) with 'external' bits

We expect that the performance of a block can be kept constant if the interleaver spreads bits to both the less and the more protected adjacent blocks. If the fraction of these bits are chosen appropriately. For the end blocks of typically highest and lowest protection, the error rates will probably change because their interleavers can only spread bits into one direction. It is important to know if the gain due to a larger interleaver size has advantages and maybe even compensates possible performance losses of certain blocks.

In the following, we discuss the construction of a relaxed block interleaver based on the S-random interleaver.

We introduce some notations in Table I.

The notations are used to distinguish information and relaxed & interleaved blocks. The block-counting superscript $t \pm i$ also applies to all following notations. Next, the notation for *relaxing* and *interleaving* blocks is defined. Let X be an arbitrary source block sequence.

TABLE I
Relaxed block interleaver notations and definitions

Notation	Meaning
U	Sequence of information blocks
U^t	Information block at time t
Δ	The sequence of relaxed and interleaved blocks
Δ^t	Relaxed and interleaved block at time t
B^t	set of source bits within a block
B_I^t	subset of B^t restricted to X_γ^t (*internal bits*)
B_E^t	subset of B^t relaxed into $X_\gamma^{t\pm i}$ (*external bits*)
B_{E-}^t	subset of external bits B_E^t relaxed into preceding blocks X_γ^{t-i}
B_{E+}^t	subset of external bits B_E^t relaxed into succeeding blocks X_γ^{t+i}
N	$\lvert B^t\rvert$, number of source bits of block n
N_I	$\lvert B_I^t\rvert$, number of internal bits
N_E	$\lvert B_E^t\rvert$, number of external bits, $N_E = N - N_I$
N_{E-}	$\lvert B_{E-}^t\rvert$, number of external bits relaxed into preceding blocks
N_{E+}	$\lvert B_{E+}^t\rvert$, number of external bits relaxed into succeeding blocks, $N_{E+} = N_E - N_{E-}$

- **Block Relaxing** determines which bits remain within the block (internal) and which will be mapped to neighboring blocks (external).

$$\gamma(X) \longrightarrow X_\gamma$$

- **Block Interleaving** permutes a given source sequence deterministically within a block of length N.

$$\pi(X) \longrightarrow X_\pi$$

A *relaxed block interleaver* can be separated into a two step process. The source sequence blocks must be interleaved (permuted) and relaxed (spread). Thus, possible generation schemes are listed below.

- *Relax/Interl.*: $U \xrightarrow{\gamma} X_\gamma \xrightarrow{\pi} X_{\gamma,\pi} = \Delta$, i.e., $U \xrightarrow{\gamma,\pi} \Delta$
- *Interl./Relax*: $U \xrightarrow{\pi} X_\pi \xrightarrow{\gamma} X_{\pi,\gamma} = \Delta$, i.e., $U \xrightarrow{\pi,\gamma} \Delta$

A relaxing parameter (μ_E) is introduced as the ratio of the number of external bits N_E to the total number of bits N of a block (block size), i.e., the fraction of relaxed bits,

$$\mu_E = \frac{|B_E^t|}{|B^t|} = \frac{N_E}{N} \ .$$

The fraction of internal bits is $\mu_I = 1 - \mu_E$. A reasonable relaxation will, of course, be limited to $50\,\%$. For the purpose of this paper μ_E = 0.1, 0.2, and 0.3 are used.

Block relaxing can be visualized as a two step process, separating B^t into internal and external sets (B_I^t and B_E^t) and then separating the external set into forward and backward sets ($B_E{-}^t$ and B_{E+}^t). The sets are then accordingly mapped to the respective blocks:

$$B_I^t \rightarrow X_\gamma^t \ ,$$

$$B_{E-}^t \rightarrow X_\gamma^{t-i} \ ,$$

$$B^t_{E+} \rightarrow X^{t+i}_{\gamma} \ .$$

The block relaxing function γ on a block X^t is thus defined as

$$\gamma(X^t) := \underbrace{B^t_{E-} \cup B^t_I \cup B^t_{E+}}_{X^t} \xrightarrow{\gamma} \underbrace{B^{t-1}_{E+} \cup B^t_I \cup B^{t+1}_{E-}}_{X^t_\gamma} \ .$$

Figure 2 provides an intuitive understanding of the block relaxing function on a block sequence leaking into the neighboring blocks.

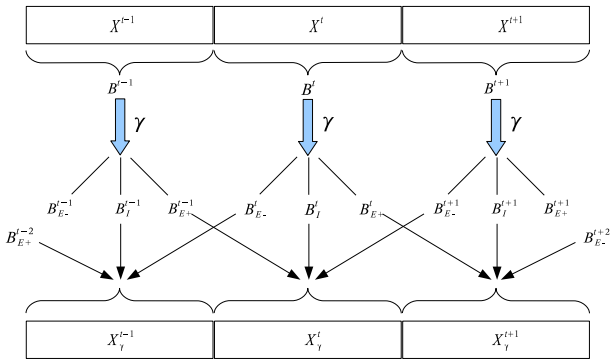

Fig. 2. Block relaxing function

One may define a block spread width w_γ as the extent to which the bits are relaxed. However, we only consider leaking to the neighboring block (class) in this paper, i.e., $w_\gamma = 3$. In the relaxation process, one has, of course, to ensure that external bits will finally be replaced by entries from other blocks. We call this property Self-Interlocking. Figure 3 gives an idea of self-interlocking. The interleave pattern is identical for the three blocks. A circle represents the block interleaving/relaxing of block X^t, square that of X^{t-1} and triangle that of X^{t+1}. It can be seen that the unique mapping of the permutation is upheld even with the relaxed interleaving.

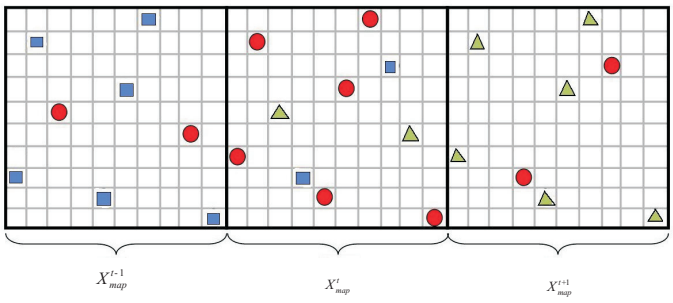

Fig. 3. Self-interlocking property of interleaver map

A. Relaxed Block Interleaver Schemes

The generation of a relaxed block interleaver based on the S-Random algorithm can be approached in two ways as described earlier. For recap purposes they are listed again below:

- *Relax/Interleave*: $U \xrightarrow{\gamma,\pi} \Delta$
- *Interleave/Relax*: $U \xrightarrow{\pi,\gamma} \Delta$

Even though the steps appear independent of each other, the order may affect the actual S value of the generated map. Before we describe the two variants, the target map is defined. Since the block spread bits to the adjacent neighboring blocks, the output relaxed block map will have a length of $3N$, where N is the size of an information block. The map is divided into three target mapping segments:

$$B^t_{E-} : [1, N] \longrightarrow \Delta^{t-1} \ ,$$
$$B^t_I : [N+1, 2N] \longrightarrow \Delta^t \ ,$$
$$B^t_{E+} : [2N+1, 3N] \longrightarrow \Delta^{t+1} \ .$$

As the process of selecting the random integers from the source integer pool of $[1, N]$ without replacement is random, it does not matter whether we first choose values for the set B^t_I or B^t_E. When choosing the integers from B^t_E for B^t_{E-} and B^t_{E+}, both are assigned $N_E/2$ integers. It would be possible to alter the balance of bits per set, yet for the purpose of this paper, each set is assigned an equal number of bits (if possible due to rounding).

Another aspect to keep in mind are the beginning and end blocks of a transmission. It is apparent that bits cannot spread in both directions in contrast to middle block. The initial information block X^0 can only spread into block Δ^0 and Δ^1, while the last information block Δ^L can only spread into the previous block Δ^{L-1} and Δ^L. Hence, for these blocks the fraction of external bits is $\mu_E/2$.

B. Interleave/Relax Algorithm Variant

First we describe the *Interleave/Relax* algorithm as it is a logical extension of the fixed block S-random interleaver.

1) **Generate S-Random Interleaver Map** The standard fixed block S-Random algorithm is used to generate an interleaver map of size N.
2) **Select internal and external bits.** Choose N_I random integers (mapping values) without replacement from integer range $[1, N]$ and add to set B^t_I. The remaining N_E integers are added to set B^t_E.
3) **Separate external bits.** Choose $N_E/2$ random integers without replacement from set B^t_E and designate as B^t_{E-}. The remaining $N_E/2$ integers are added to set B^t_{E+}.
4) **Relax Bits** The bit locations chosen for the individual sets are relaxed into their target blocks.

Figure 4 shows the Interleave Relax algorithm variant. It is clear that the relaxed interleaver map will inherit the S constraint of the input map. The interleaver input map is simply separated into B_I, B_{E-}, and B_{E+}, which are then shifted into their appropriate ranges in the output map, without changing the internal relationship regarding the S constraint.

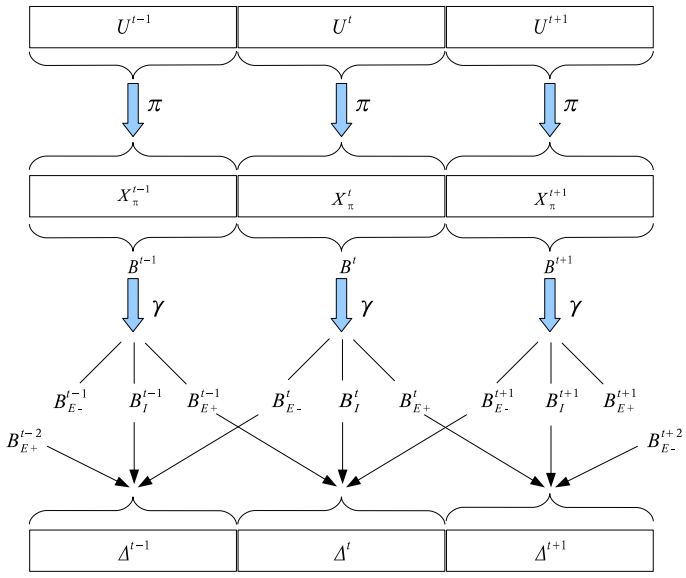

Fig. 4. Relaxed block interleaver; *interleave/relax* generation algorithm

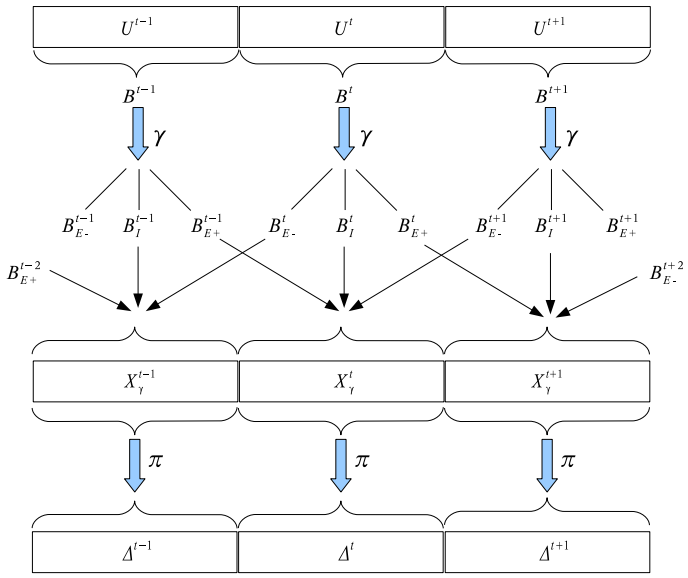

Fig. 5. Relaxed block interleaver; *relax/interleave* generation algorithm

C. Relax/Interleave Algorithm Variant

The Relax/Interleave variant first separates B into B_I, B_{E-}, and B_{E+}, then interleaves each set individually into a map of length N, then maps these blocks into their respective ranges within the output map. The algorithm is as follows.

1) **Select internal and external bits.** Choose N_I random integers (mapping values) without replacement from integer range $[1, N]$ and add to set B_I^t. The remaining N_E integers are added to set B_E^t.
2) **Separate external bits.** Choose $N_E/2$ random integers without replacement from set B_E^t and designate as B_{E-}^t. The remaining $N_E/2$ integers are added to set B_{E+}^t.
3) **Interleave internal bits.** A modified version of the S-Random algorithm interleaves B_I^t into Δ^t.
4) **Interleave external bits.** A modified version of the S-Random algorithm interleaves B_{E-}^t and B_{E+}^t into Δ^{t-1} and Δ^{t+1}, respectively.

Figure 5 shows Relax Interleave R/I algorithm scheme graphically.

The S-random algorithm is slightly modified. In the regular S-Random algorithm, the mapping vector is filled one by one sequentially. However, if this was the case for *relax/interleave*, then the sets B_I^t, B_{E+}^{t-1}, and B_{E-}^{t+1} would always be arranged in a fixed order within block Δ^t. Thus, it is necessary to also randomly choose which mapping target location to process at a given time, such that the bit sets are randomly distributed throughout Δ^t. Note that the S-random constraint only needs to be fulfilled within each set, i.e., B_I^t, B_{E+}^{t-1}, B_{E-}^{t+1}. The latter observation will probably allow for interleaver maps resulting from the *relax/interleave* algorithm to achieve higher S constraint values, as the position mapping is less dense.

IV. Simulation Results

This section provides simulation results for UEP Turbo coding using the Relaxed Block Interleaver (RBI) compared with the Fixed Block S-random interleaver (FBI). Three blocks, each of size $N = 1000$, were used as an example. The blocks are assigned priorities and are encoded at different code rates R_i ($R_1 = 1/2$, $R_2 = 3/8$, $R_3 = 1/4$). Figure 6 shows the performance curves for UEP Turbo codes using a fixed block S-random interleaver map with a block size of $N = 1000$. Since a relaxed block interleaver is designed on the basis of S-random interleaving, the performance curves are used as reference curves for comparison with the performance of relaxed block S-random interleaving.
We did not see significant performance differences for the two possible orders of relaxing and interleaving.

Figure 7 shows the BER curves for UEP Turbo codes using the Relaxed block interleaver with $\mu_E = 0.2$ relative to Turbo coding with FBI. Bits from the mid-priority block ($R_2 = 3/8$) are relaxed into neighbors high ($R_3 = 1/4$) as well the less priority ($R_1 = 1/2$) blocks. The performance of the less important block must improve as its external bits are encoded at a lower rate, while the performance of the high priority block must degrade as its external bits are encoded at a higher rate. Moreover, since the middle block relaxes bits to both the high as well as the less important blocks, its performance should roughly remain constant. Indeed such results become visible from Fig. 7. Furthermore, we realize a flooring towards the performance of the worst protection involved in a certain block. In Fig. 8, average BER curves for UEP Turbo coding using the RBI map with $\mu_E = 0.1, 0.2, 0.3$ are shown relative to UEP Turbo coding using the FBI map. As foreseen, the average performance improves with increasing μ_E, due to the increase of the utilized interleaver size.

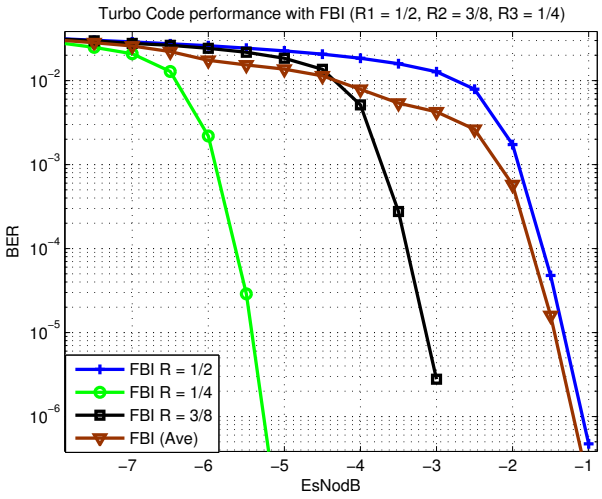

Fig. 6. Turbo code performance with FBI map

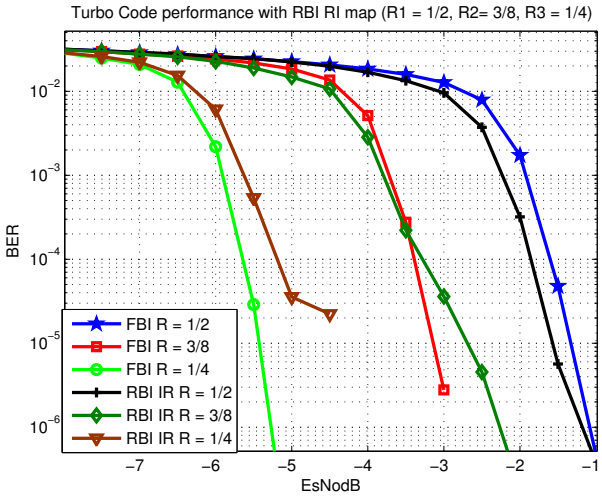

Fig. 7. Turbo code performance with RBI vs. FBI map

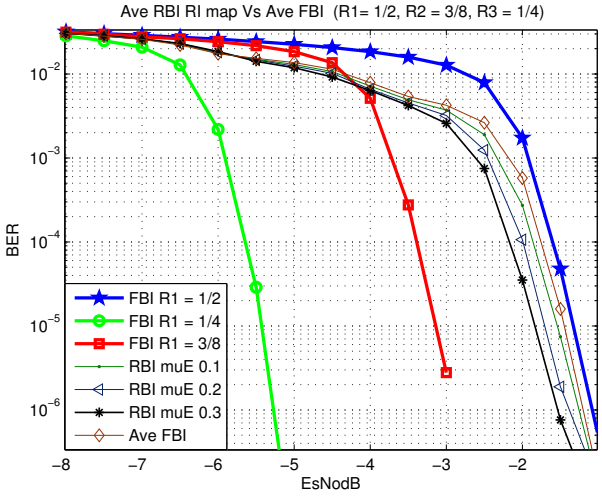

Fig. 8. Turbo code average performance with RBI map vs. FBI ave. Performance

V. Conclusions

In this paper, we have shown an interleaver design that relaxes the borders between priority classes of UEP Turbo codes. We observed that due to effectively utilizing a bigger interleaver, the average performance is improved. However, a flooring of the error-performance curves is investigated directed towards the lowest performing code addressed by leaked bits. The improved average performance is especially due to the improvements obtained in the lowest-priority class, which dominates the average performance. After all, a leaking interleaver design is just another measure for modifying the performance curves of UEP Turbo codes, additional to puncturing and pruning.

Acknowledgment

This work has been supported by the German National Science Foundation (DFG) and the Higher Education Commission (HEC), Pakistan.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-correcting Coding and Decoding: Turbo Codes. 1," proc. *IEEE International conference on Communication, 1993. ICC 93. Geneva,* Vol. 2, pp. 1064-1070, May 23-26, 1993.

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

[3] C.-H. Wang and C.-C. Chao. "Path-compatible Pruned Convolutional (PCPC) Codes," *IEEE Transactions on Communications*, Vol. 50, No. 2, Feb. 2002, pp. 213-224.

[4] W. Henkel and N. von Deetzen, "Path Pruning for Unequal Error Protection Turbo Codes," proc. *2006 International Zurich Seminar on Communication*, February 22-24, 2006, pp. 142-145.

[5] N. von Deetzen, *Modern Coding Schemes for Unequal Error Protection*, PhD Thesis, Shaker, March 2009.

[6] C. Fragouli and R. Wesel, "Semi-random Interleaver Design Criteria," *Global Telecommunications Conference, 1999. GLOBECOM '99*, Vol. 5, pp. 2352-2356, 1999.

[7] H. Sadjadpour, N. Sloane, M. Salehi, and G. Nebe, "Interleaver Design for Turbo Codes," *Selected Areas in Communications, IEEE Journal on*, Vol. 19, No. 5, pp. 831 -837, May 2001.

[8] P. Popovski, L. Kocarev, and A. Risteski, "Design of Flexible Length S-random Interleaver for Turbo Codes," *IEEE Communication Letters*, Vol. 8, No. 7, pp. 461-463, July 2004.

[9] J. Hokflet,, O. Edfors, and T. Maseng, "A Turbo Code Interleaver Design Criterion Based on the Performance of Iterative Decoding," *Communications Letters, IEEE*, Vol. 5, No. 2, pp. 52-54, Feb. 2001.