Interleaver Design Using Backtracking and Spreading Methods

Marco Breiling[1], Stein Peeters[2], and Johannes Huber
Lehrstuhl für Nachrichtentechnik II, Universität Erlangen-Nürnberg, Cauerstr. 7, D-91058 Erlangen, Germany
Tel.: (+) 9131-85-27668, Fax: (+) 9131-85-28919, Email: breiling@LNT.de, http://www.LNT.de/~breiling

Abstract — **We propose a new algorithm for Turbo code interleaver design, which is based on the conventional s–random approach and whose complexity grows only linearly with the interleaver length.**

Designing the interleaver $\boldsymbol{\pi} = (\pi_1; ..; \pi_K)$ of length K of a Turbo code serves to increase the code's minimum distance $\delta_{\min}$ and hence to lower the *error floor* of the Word and Bit Error Rates (WER/BER). An efficient method was presented in [1]. Examinations show that for so–designed interleavers, the codeword at $\delta_{\min}$ is mainly caused by a combination of an input word $\mathbf{u}^{(1)}$ of the first component encoder (identical to the Turbo encoder input $\mathbf{u}$) and a second component input word $\mathbf{u}^{(2)}$ as shown in Fig. 1. In this example, "1001" represents an *error pattern*, i.e. an input sequence causing a **short** error event in a component code trellis. The s–random interleaver $\boldsymbol{\pi}$ does not avoid that the four "1"s in the *two* error patterns of $\mathbf{u}^{(1)}$ are mapped crosswise to *two* error patterns in $\mathbf{u}^{(2)}$, since the two "1"s belonging to each error pattern in $\mathbf{u}^{(1)}$ are spread to distant positions in $\mathbf{u}^{(2)}$, and hence the spreading condition of [1] is satisfied. However, this unlucky mapping of positions can be avoided and $\delta_{\min}$ can be increased by modifying the interleaver design algorithm.

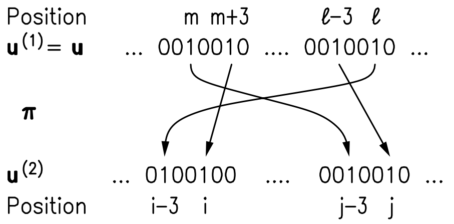

Figure 1: Unlucky mapping of positions

The proposed algorithm incorporates the s–random method of [1] and hence, it successively determines π_1 to π_K. In step l, we set up the set $\mathcal{A}_l \subset \{1; ..; K\}$ of possible values for π_l, which have not already been assigned to π_t in earlier steps $t < l$, and which satisfy the spreading condition of [1]. Moreover, step l consists in determining and discarding values of $\mathcal{A}_l$, which would cause an unlucky mapping like in Fig. 1.

Determining these unfavourable values in $\mathcal{A}_l$ can be done very efficiently using a recursive backtracking approach, which is shortly outlined using the example of Fig. 1. Our basic observation is that any "1" present in $\mathbf{u}^{(1)} = (u_1^{(1)}; ..; u_K^{(1)})$ or $\mathbf{u}^{(2)} = (u_1^{(2)}; ..; u_K^{(2)})$, respectively, must belong to an error pattern. Otherwise the associated codeword has large weight and can be ignored, since we consider and try to avoid only low weight codewords. In step l, we consider exclusively $\mathbf{u}^{(1)}$ with $u_l^{(1)} = 1$ and $u_t^{(1)} = 0$, $\forall t > l$. Our starting point for the backtracking is that the "1" in $u_l^{(1)}$ must belong to an error pattern (as reasoned above). Every possible error pattern must be considered, and for each of them, we must proceed in a backtracking manner. In our example of Fig. 1, we consider only the error pattern "1001" in $u_{l-3}^{(1)}$ to $u_l^{(1)}$. Since $\pi_{l-3} = j$ has already been determined, we know that $u_j^{(2)} = 1$. Following the above reasoning, the "1" in $u_j^{(2)}$ must belong to an error pattern, for which we must consider every possibility. In the Fig., we consider "1001" in $u_{j-3}^{(2)}$ to $u_j^{(2)}$. For the case that $j-3$ has earlier been assigned to π_m, $m < l$, we conclude that $u_m^{(1)} = 1$. Every possible new error pattern containing $u_m^{(1)} = 1$ must be considered in $\mathbf{u}^{(1)}$ (in the Fig. "1001" in $u_m^{(1)}$ to $u_{m+3}^{(1)}$). Finally, for $\pi_{m+3} = i$, we find that $u_i^{(2)} = 1$. We must thus discard $i-3$ from $\mathcal{A}_l$, since this prevents the assignment $\pi_l = i-3$, which would otherwise complete the unlucky mapping in Fig. 1. When all unfavourable values have been discarded from $\mathcal{A}_l$, then π_l is randomly chosen from the remaining values. The backtracking algorithm works also for error patterns of weight > 2. The complexity of a complete interleaver design grows linearly with K.

We verified the proposed algorithm by designing an interleaver of length $K = 200$ for a Turbo code of rate 1/2 employing $M = 2$ component codes (generator polynomials: $(1; 5/7)$). In the design, we used $s = 8$ and considered all error patterns of weight ≤ 3. For a termination of both component trellises, this Turbo code attains $\delta_{\min} = 14$. Fig. 2 shows the WER (upper curves) and BER (lower curves) for varying E_b/N_0 (received energy per information bit over the one–sided noise power spectral density) for a simulated transmission using coded BPSK over an AWGN channel. The performance is compared to using a pure s–random interleaver [1] with $s = 10$ (expected $\delta_{\min} \leq 12$) and a *uniform interleaver* [2] (mean $\delta_{\min} \leq 6$) of the same length. We can clearly see the improved BER and particularly WER for higher E_b/N_0.

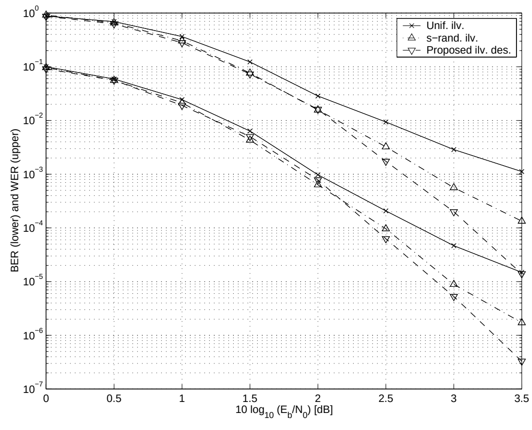

Figure 2: Simulation results

References

[1] S. Dolinar and D. Divsalar, "Weight Distributions for Turbo Codes Using Random and Nonrandom Permutations", *JPL–TDA Progress Report*, pp. 56–65, 1995.

[2] S. Benedetto and G. Montorsi, "Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding Schemes", *IEEE Trans. on Info. Th.*, pp. 409–428, 1996.

[1]M. Breiling is sponsored by the Fraunhofer Gesellschaft – Institut für Integrierte Schaltungen, Erlangen.

[2]S. Peeters is now working for BU Satellite Communications, Nortel Dasa Network Systems, Friedrichshafen/Germany.