Design of Flexible-Length S-Random Interleaver for Turbo Codes

Petar Popovski, *Student Member, IEEE*, Ljupco Kocarev, *Senior Member, IEEE*, and Aleksandar Risteski, *Member, IEEE*

***Abstract*—We introduce a method for generating a sequence of semi-random interleavers, intended to be optimally stored and employed in a turbo coding system that requires flexibility of the input block (i.e., interleaver) size N. A distinctive feature of this method is seen in the very simple rules for obtaining shorter/longer interleavers by pruning/adding positions to the interleaver currently used in the system. For each N, the obtained bit error rate (BER) is not higher than the BER for ordinary *S*-random interleaver of the same N. The method always converges and is suitable for obtaining interleavers of large lengths.**

***Index Terms*—Error correction coding, turbo codes, interleaver.**

I. Introduction

THE advent of turbo codes [1] motivated an extensive research aimed to explain and further improve their extraordinary error performance. A codeword in turbo codes is produced by concatenation of the outputs of two convolutional encoders and the decoding is performed by an iterative procedure. The requirement from the interleaver in turbo-coding scheme is twofold. First, it should produce codewords with good Hamming spectrum by avoiding low-weight codewords [2]. Second, the suitability of the interleaver can be assessed with respect to the suboptimal decoding algorithm [3]. To meet both requirements, the heuristics of obtaining interleaver with good spreading properties has been employed [2], [4]. The *S*-random interleaver [2] is considered to be a reference randomized interleaver with good spreading properties, which yields excellent BER performance.

Considering the envisioned applications and the system-level performance, it is essential that the turbo-coded subsystem can flexibly adjust the size of its interleaver, e.g. according to the requirements by the upper protocol layers. In this letter we propose an algorithm for design of a *Flexible-Length S-random (FLS-random)* interleaver. For a given interleaver length N, the FLS-random interleaver results in bit error rate (BER) that is not higher than the BER resulting from the *S*-random interleaver constructed for that particular N. The spread of the FLS-random interleaver grows as $O(\sqrt{N})$. The FLS-random interleaver has very simple rules for obtaining shorter/longer interleavers by pruning/adding positions to the interleaver currently used in the system, which makes it suitable for implementation.

Manuscript received November 23, 2003. The associate editor coordinating the review this letter and approving it for publication was Dr. I. Fair.

P. Popovski is with the Center for TeleInFrastructure (CTIF), Aalborg University, 9220 Aalborg, Denmark (e-mail: petarp@kom.auc.dk).

L. Kocarev is with the Institute for Nonlinear Science, University of California at San Diego, La Jolla, CA 92093-0402 USA (e-mail: lkocarev@ucsd.edu).

A. Risteski is with the Faculty of Electrical Engineering, University "Sts. Cyril and Methodius", 1000 Skopje, Macedonia (e-mail: acerist@cerera.etf.ukim.edu.mk).

Digital Object Identifier 10.1109/LCOMM.2004.832737

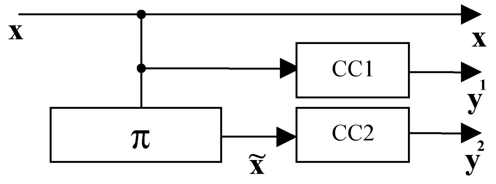

Fig. 1. Turbo-coder: CC1 and CC2 are recursive convolutional codes; π is a random permutation.

II. The Role of the Interleaver

Fig. 1 depicts the considered turbo-coding system. The turbo codeword is obtained via concatenation of three component bit blocks: the input information bits $\mathbf{x}$ and the parity bits of each constituent convolutional code (CC), $\mathbf{y}^1$ and $\mathbf{y}^2$, respectively. We consider the CCs to be recursive convolutional codes with primitive polynomial in the feedback connection [5]. The input block to the CC2, denoted by $\tilde{\mathbf{x}}$, is obtained by permuting the sequence $\mathbf{x}$ according to the permutation law π implemented by the interleaver. Let N denote the interleaver length. Then the blocks $\mathbf{x} = (x_0, x_1, \cdots, x_{N-1})$ and $\tilde{\mathbf{x}} = (\tilde{x}_0, \tilde{x}_1, \cdots, \tilde{x}_{N-1})$ are consisting of N bits, i.e. $\mathbf{x}, \tilde{\mathbf{x}} \in \{0,1\}^N$. The interleaver is described by the permutation map[1] $\pi_N : Z_N \to Z_N$, where $Z_N = \{0, 1, \cdots, N-1\}$, such that $\tilde{x}_{\pi_N(i)} = x_i$ for each $i \in Z_N$. Hence, the i-th bit from the input block $\mathbf{x}$ appears as $\pi_N(i)$-th bit in the permuted block $\tilde{\mathbf{x}}$. The *cycle length (CL)* $C_{\pi_N}(i,j)$ for bits x_i and x_j with respect to the permutation π_N is defined as

$$C_{\pi_N}(i,j) = |i-j| + |\pi_N(i) - \pi_N(j)| \tag{1}$$

The *minimum cycle length (MCL)* for the permutation π_N is defined as

$$M(\pi_N) = \min_{i,j,i\neq j} C_{\pi_N}(i,j) \tag{2}$$

The quantity $\Lambda(\pi_N)$ is defined as

$$\Lambda(\pi_N) = \sum_{i,j,i\neq j} \mathrm{Ind}\left(C_{\pi_N}(i,j) = M(\pi_N)\right) \tag{3}$$

where $\mathrm{Ind}(\texttt{statement})$ is an indicator function, taking value 1 when `statement` is true and 0, otherwise.

The *S*-random permutation (interleaver) of length N is constructed inductively [2]. Pick at random $j_0 \in Z_N$

[1]Note that this permutation law is inverse of the standard one; we have adopted it to facilitate the description of the FLS interleaver construction.

and set $\pi_N(j_0) = 0$. Having defined each j_m such that $\pi_N(j_m) = m < k$, where $0 < k < N$, the value j_k for which $\pi_N(j_k) = k$, is obtained as follows. Pick at random $j \in Z_N$ such that $\pi_N(j)$ is still undefined. If there is j_m, such that $|j - j_m| \le S$ and $\pi_N(j) - \pi_N(j_m) \le S$, then reject that j. Otherwise, accept $j = j_k$ and go to determine j_{k+1} in the analogous way. Repeat until j_{N-1} is found.

In fact, MCL and S are two different measures of the interleaver's spreading [4]. In [6], MCL is referred to as triangular spread, while S as square spread. For given N, the theoretical upper bound [4], [6] on MCL is $\sqrt{2N}$, while for S it is $\sqrt{N}$. From (2) and (3), it can be easily shown that

$$S + 2 \le \mathrm{MCL} \le 2S + 1 \tag{4}$$

The heuristics behind the *S*-random interleaver is to achieve good *S*-factor, while keeping the interleaver essentially random. However, apart from the theoretical limit, only with $S < \sqrt{N/2}$ the above algorithm converges in reasonable time [2]. If for some S the algorithm does not yield convergence, S should be decreased. For large N, this decrease can lead to values of S which are much lower than than the heuristic recommendation $\sqrt{N/2}$. The original *S*-random construction creates interleaver for fixed N and if the coding system requires flexibility of N, then several permutation laws should be created and stored. A common way to avoid such storing is to obtain shorter interleavers by "pruning" larger ones, i.e. to leave the excess positions unused during the permutation. However, this operation destroys the interleaver properties and notably degrades the performance. *Prunable S-random (PS-random)* interleaver in [7] is built starting from a *S*-random interleaver with spread S of the shortest needed size and randomly inserting the new positions, while retaining the constant spread S. This algorithm inherits the convergence problems from the original *S*-random construction. If for given S the solution is not obtained after several runs, S should be decreased [7]. In addition, if the largest size is significantly larger than the starting size, then $S \lll \sqrt{N/2}$.

III. Flexible-Length S-Random (FLS-Random) Interleaver

The FLS-random interleaver is based on an iterative construction. Assume that we already have a *S*-random permutation of length K, denoted by π_K and let $L \gg K$ be the maximal interleaver length of interest. Starting from $N = K$, each permutation π_{N+1} of length $K < N + 1 \le L$ is obtained from the permutation π_N. Let π_N be defined by having $\pi_N(i)$ for each $i \in Z_N$. Then, the permutation $\pi_{N+1}(i)$ is obtained by **inserting** $N = Z_{N+1} \setminus Z_N$ at position j_N in π_N as follows:

$$\pi_{N+1}(i) = \begin{cases} \pi_N(i), & 0 \le i < j_N \\ N, & i = j_N \\ \pi_N(i-1), & j_N < i \le N. \end{cases} \tag{5}$$

The rule of producing permutation π_{N+1} from π_N, given by (5), will be shortly written as:

$$\pi_{N+1} = \mathbf{I}(\pi_N, j_N) \tag{6}$$

The algorithm starts by choosing the initial permutation π_K randomly. All permutations of lengths $(N+1)$ where $K < N+1 \le L$ are obtained through $(L-K)$ iterations, using (6) in each step. At the iteration in which π_{N+1} is obtained from π_N the objective is to select appropriate j_N. For given N, the value of j_N is selected using the following four steps.

Step 1) Create the set X_{N+1} that consists of $(N + 1)$ permutations $\pi^j_{N+1} = \mathbf{I}(\pi_N, j)$, obtained for all $j \in Z_{N+1}$.

Step 2) For each permutation in X_{N+1}, find $M(\pi^j_{N+1})$, as defined in (2), and let $\mu_{N+1} = \max_{j\in Z_{N+1}} M(\pi^j_{N+1})$. Then create the set $B_{N+1} \subset Z_{N+1}$, such that $m \in B_{N+1}$ iff $M(\pi^m_{N+1}) = \mu_{N+1}$.

Step 3) For each $m \in B_{N+1}$, find $\Lambda(\pi^m_{N+1})$, as defined in (3) and let $\lambda_{N+1} = \min_{m\in B_{N+1}} \Lambda(\pi^m_{N+1})$. Then create the set $D_{N+1} \subset B_{N+1}$, such that $k \in D_{N+1}$ iff $\Lambda(\pi^k_{N+1}) = \lambda_{N+1}$.

Step 4) Pick randomly, uniformly, $k \in D_{N+1}$, set $j_N = k$, and use (5) to create $\pi_{N+1} = \mathbf{I}(\pi_N, j_N)$.

Step 1 creates a set $\{\pi^j_{N+1}\}$ of $(N + 1)$ permutations that are obtained by selecting all possible values for $j \in Z_{N+1}$ and for each such value of j applying (5). The objective of Step 2 is to maximize the MCL of the permutation. Therefore, this step filters out from Z_{N+1} a subset of values B_{N+1} which result in maximal MCL. Note that the set B_{N+1} is always nonempty, since at least one permutation from $\{\pi^j_{N+1}\}$ has maximal MCL. Step 3 does not affect the MCL of the permutation π_{N+1} which is currently constructed. However, this is a crucial step which leads to fast increase of both MCL and S in the construction of the FLS-random interleaver. The heuristics behind Step 3 is to select permutation which will yield highest potential for the MCL to grow in the further iterations. The subset D_{N+1} obtained in this iteration is always nonempty due to the same reasons from Step 2. Finally, Step 4 introduces the randomness in the selection of j_N. In the simulation of the FLS-random construction we have noticed that $|D_{N+1}| > 1$ for many lengths $N + 1$, such that the Step 4 preserves the needed randomness in the structure of the obtained *S*-random interleaver. Since $|B_{N+1}| \ge |D_{N+1}| > 1$, the proposed greedy algorithm *always* converges. This variant of the algorithm is rather a brute-force, and the complexity of determining j_N is $O(N^3)$. Some optimizations can be introduced to speed-up the process, but we omit them due to the limited space.

As it will be seen, the values of both MCL and S for the FLS-random interleaver are increasing as $O(\sqrt{N})$. Hence, this algorithm can be used to construct very large *S*-random interleavers. Perhaps the advantageous feature of the FLS-random interleaver is reflected in the very simple rules for obtaining shorter/longer interleavers by pruning/adding positions to the interleaver currently used in the system. Let the communication system employ interleavers of lengths N for $K \le N \le L$. Then the system needs only to store π_K and the values $j_K, j_{K+1}, \ldots, j_{L-1}$ in order to employ any interleaver with length $K \le N \le L$. For example, let the coding system start with interleaver length N_1; then, to obtain π_{N_1}, start with π_K and insert successively at positions $j_K, j_{K+1}, \ldots, j_{N_1-1}$. Then, if during the system operation, the interleaver should be increased to length $N_2 > N_1$ (e.g. to decrease the bit error rate), make insertions at the the positions $j_{N_1}, j_{N_1+1}, \ldots, j_{N_2-1}$. If,

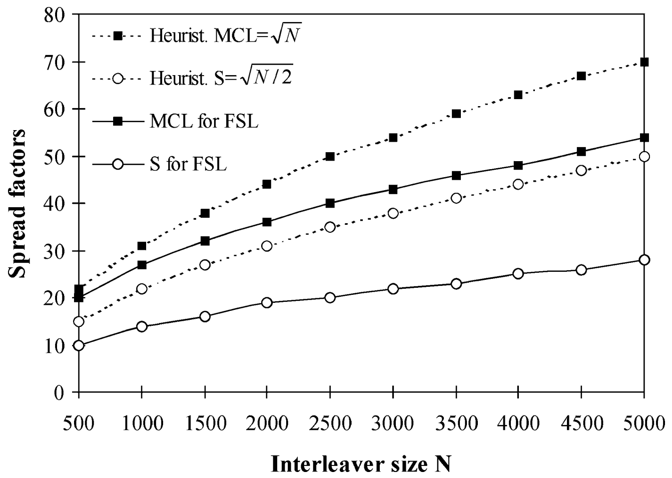

Fig. 2. Comparison of spread factor achieved by the FLS-random interleaver with the heuristic upper bound on the spread factor for *S*-random interleavers, for different interleaver length N.

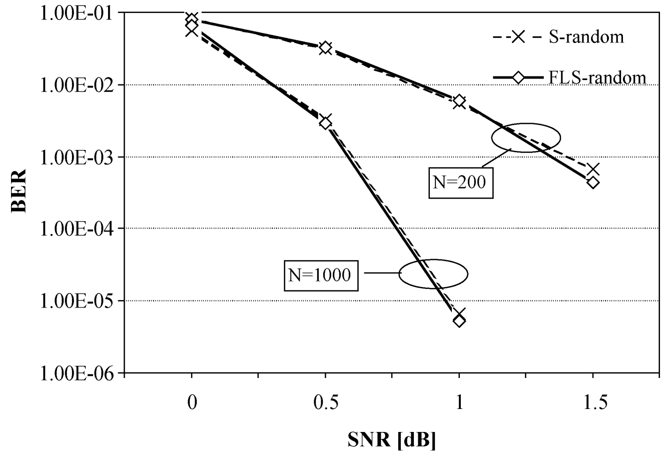

Fig. 3. Bit error rate offered by the FLS-random and *S*-random interleaver at different signal to noise rations. Comparison is made for two different lengths $N = 200$ and $N = 1000$.

otherwise, the interleaver length should be decreased from N_1 to N_0, then a pruning is applied iteratively according to the following relation:

$$\pi_{N-1}(i) = \begin{cases} \pi_N(i), & 0 \le i < j_N \\ \pi_N(i+1), & j_N \le i \le N-2. \end{cases} \tag{7}$$

IV. Results

We have constructed FLS-random interleavers starting with interleaver of length as small as $N = 6$ and getting $N = 5000$ by successive insertions. A great advantage of the algorithm is that the starting interleaver can be very short. For the results on Figs. 2 and 3 we have used the starting permutation $\pi_6 = \{5, 1, 3, 0, 4, 2\}$. By simulating the construction of the FLS-random interleaver, we have verified that the short starting permutation can be pseudorandom, without affecting the properties of FLS-random interleavers with $N > 50$. Fig. 2 shows the values of the two spreads, MCL and S, as function of N. For each permutation of size N, MCL is sampled from the algorithm for interleaver construction, while S is calculated as a maximal value for which the conditions of the *S*-random interleaver are satisfied for each pair of positions (i, j). These spreads are compared with the heuristical limits for MCL and S, which are $\sqrt{N}$ and $\sqrt{N/2}$, respectively. Although the values of MCL and S for FSL are always lower than the heuristical limits, they both grow as $O(\sqrt{N})$. For fixed MCL, (4) implies that the minimal value for S is $S = (\mathrm{MCL} - 1)/2$ and Fig. 2 shows that our interleaver produces S close to the minimal value. However, note that for large N, the *S*-random construction may not produce a result if S is kept at the heuristical limit, such that S should be decreased. More importantly, the decrease in the spreads for FLS-random interleaver does not deteriorate its BER performance, as Fig. 3 shows. Fig. 3 depicts BER for *S*-random and FLS-random interleaver at two lengths $N = 200$ and $N = 1000$. The *S*-random interleaver has been constructed by taking the heuristical value $S \approx \sqrt{N/2}$. Channel with additive Gaussian white noise has been used for several signal-to-noise ratios (SNRs). The constituent CC's are identical, each with generator matrix (1,17/15). It can be seen that the BER of the FLS-random interleaver is lower or equal to the BER of S-random interleaver for each SNR.

V. Conclusions

In this letter we have proposed the design of a flexible-length *S*-random (FLS) interleaver. The algorithm always converges and produces a solution, while inherently adapting the spread factors to the interleaver length N. The spread factors MCL and S of the FLS-random interleaver grows as $O(\sqrt{N})$ and its structure embeds randomness. Although the obtained spread factors are lower than the heuristical limits of MCL and S for each N, the BER that results from an interleaver obtained by this method is never higher than the BER offered by the *S*-random interleaver of the same length.

The FLS-random interleaver provides optimized storage of a sequence of interleavers in a turbo-coding system. The proposed interleaver construction includes very simple rules for obtaining larger/shorter interleavers from the interleaver that is currently employed in the system, which is a major practical implication of the FLS-random interleaver.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE Int. Commun. Conference (ICC)*, 1993, pp. 1064–1070.
[2] S. Dolinar and D. Divsalar, Weight distribution for turbo codes using random and nonrandom permutations, in TDA Progress Rep. 42-122, Aug. 1995.
[3] J. Hokfelt, O. Edfors, and T. Maseng, "Assessing interleaver suitability for turbo codes," in *Nordic Radio Symp.*, Saltsjobaden, Sweden, Oct. 1998.
[4] S. Crozier, "New high-spread high-distance interleavers for turbo-codes," in *Proc. 20th Biennial Symp. on Communications*. Kingston, ON, Canada, May 2000, pp. 3–7.
[5] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Trans. Commun.*, vol. 44, no. 5, May 1996.
[6] L. Dinoi and S. Benedetto, "Design of prunable *S*-random interleavers," in *Proc. Int. Symp. on Turbo Codes*, Brest, France, 2003, pp. 279–282.
[7] M. Ferrari, F. Scalise, and S. Bellini, "Prunable S-random interleavers," in *Proc. IEEE Int. Commun. Conf. (ICC)*, vol. 3, 2002, pp. 1711–1715.