High-Performance Low-Memory Interleaver Banks for Turbo-Codes

Stewart Crozier and Paul Guinand

Communications Research Centre, 3701 Carling Ave., P.O. Box 11490, Station H, Ottawa, Canada
K2H 8S2, Ph: 978-448-0994, Fax: 978-448-4046, Email: stewart.crozier@crc.ca, Web: www.crc.ca/fec

Abstract

A new method of designing high-performance, low-memory, interleaver banks for Turbo-codes is presented. The new interleavers are called dithered relative prime (DRP) interleavers. Only a small number of parameters are required to both store and implement each interleaver in the bank. The error rate performance is similar to that achieved by other good interleaver designs that typically require the storage of all K indexes for each interleaver of length K.

1 Introduction

Turbo-codes [1,2] have received considerable attention since their introduction in 1993. This is due to their powerful error correcting capability, reasonable complexity, and flexibility in terms of accommodating different block lengths and code rates. The Turbo-code (TC) encoder considered here consists of two 8-state, rate 1/2 recursive systematic convolutional (RSC) encoders operating in parallel with the data bits, d_i, interleaved between the two RSC encoders, as shown in Figure 1. The (feedback, feedforward) polynomials are (13,15) octal, as specified by the 3GPP standard [3]. Without puncturing, the overall code rate is 1/3. Other code rates are obtained by puncturing the coded bits. Standard practice has been to only puncture the parity bits, p_i. However, it will be shown that a significant increase in (Hamming) distance can be achieved by also puncturing a small number of data bits. A high minimum distance is desirable for both lowering the so-called "error floor" or flare and for making the asymptotic flare performance as steep as possible.

Interleaving is a key component of Turbo-codes, as shown in Figure 1. Two interleaver types that have been commonly investigated are the "random" interleaver and the so-called "*S*-random" or "spread" interleaver [4,5,6]. It was recognized early on that good spreading properties are desirable for both fast convergence and good distance properties. More recent high-spread interleavers include the dithered golden interleavers introduced in [7], and the low extrinsic correlation interleavers described in [8]. An efficient method of generating high-spread random (HSR) interleavers was described in [9]. This method also uses a more natural and effective definition of spread that is closely related to the distance properties of Turbo-codes. The same spread definition is used here. The HSR method, along with distance spectrum testing and index shuffling to eliminate low-weight codewords (post-processing), has provided some of the best performance results to date. The HSR method is used herein as one performance benchmark.

The above interleaver design methods typically require that all K_b indexes be stored to implement a single interleaver of length K_b. This is not a major concern when only one interleaver is required, as the other memory requirements for the corresponding TC encoder and decoder are also order K_b. However, when a bank of B interleavers is required to accommodate B different block lengths, and B is on the order of the longest interleaver length, K_B, then the interleaver bank memory requirements become order K_B^2. This can be prohibitive, especially if K_B is many thousands of bits. This is the interleaver bank problem.

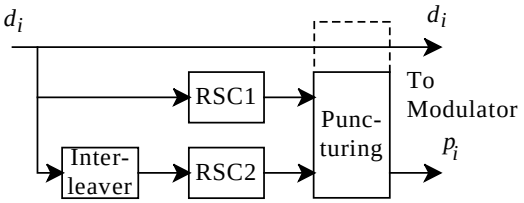

Fig. 1: TC encoder with two rate 1/2 RSC encoders.

In general, there are several criteria that a good interleaver bank should satisfy. The bank should provide a wide range of interleaver lengths, for example from a few tens of bits to many thousands of bits, depending on the application. The bank should have good resolution with convenient interleaver lengths. For example, the lengths could increase by 1 or 2 bits for short lengths (tens of bits), by a single byte (8 bits) for medium lengths (hundreds of bits), or by a few bytes for long lengths (thousands of bits). The amount of memory required to define and store each interleaver should be low. Ideally, there should only be a few parameters per interleaver length. The algorithm used to generate the interleaver indexes should also be simple. If the algorithm is simple enough, the indexes for a selected interleaver can be generated "on-the-fly", as needed by the encoder and decoder, saving even more memory. On-the-fly index generation is considered a bonus feature since the overall memory requirements remain order K_B, with or without this feature. However, this feature can still reduce the amount of memory required and simplify the initialization process when changing block lengths. Finally, the interleaver bank should provide good error rate performance for all block lengths.

It is easy to design highly structured interleaver banks that satisfy all of the above criteria, except for the last one. The challenge is to get good performance too. For example, given a block length, K, a simple relative prime (RP) interleaver can be defined by just one other parameter, p, the modulo-K index increment [7]. These interleavers can easily achieve high spreads and thus can eliminate the worst-case low-weight codewords. In fact, these interleavers do provide excellent performance for short block lengths. However, the performance for medium and long block lengths is poor because of the large number of compound low-weight codewords generated by the repetitive structure. Another example of a low-memory interleaver bank is that specified in the 3GPP standard [3]. The 3GPP standard is used herein as one performance benchmark. The dithered-diagonal interleavers described in [9] are also candidates. In particular, excellent performance results have been obtained for the special block lengths of $K=2n^2$, where n is an integer, but not a multiple of 7 (the period of the feedback polynomial in the RSC encoders). These special interleavers can be stored and implemented using just n index increment values. This represents a significant reduction in the memory requirements. However, the bank resolution is rather coarse and the block lengths are not the most convenient (e.g. they are generally not multiples of bytes). Even so, it was the good error rate performance, and the low-memory requirements, of these special dithered diagonal interleavers that partly motivated the new approach presented here.

This paper describes a new family of interleavers, called dithered relative prime (DRP) interleavers, that provides a good solution to the interleaver bank problem for Turbo-codes. Section 2 reviews the interleaver and spread definitions. Section 3 describes the new DRP approach. Section 4 addresses distance testing and presents some example distance results. Section 5 presents example simulation results. Section 6 contains the conclusions.

2 Interleaver and Spread Definitions

Interleavers can be defined and implemented in a number of different ways. Figure 2 shows the definition used here. The interleaver reads from a vector of input symbols or samples, $\boldsymbol{v}_{in}$, and writes to a vector of interleaved or permuted output samples, $\boldsymbol{v}_{out}$. The output samples are written using the write indexes i=0…K-1, where K is the interleaver length. Vector $\boldsymbol{I}$ defines the order that the samples are read from the input vector. That is, the i-th output, written to location i in the output vector, is read from location $I(i)$ in the input vector. The interleaver is completely defined by read vector $\boldsymbol{I}$.

For example, letting $[x]_m$ denote x modulo-m arithmetic, a simple RP interleaver of length K is defined by

$$I(i) = [s + ip]_K, \quad i=0\ldots K\text{-}1 \tag{1}$$

where p and K are relative primes and s is the starting index. Note that $\boldsymbol{I}$ can also be computed recursively using

$$I(i) = [I(i-1) + p]_K, \quad i=1\ldots K\text{-}1 \tag{2}$$

where $I(0)=s$. Thus, an RP interleaver can be implemented using a single modulo-K index increment, p.

The new spread measure associated with two write indexes i and j, for any interleaver $\boldsymbol{I}$, is defined as [9]

$$S''_{new}(i, j) = |I(i) - I(j)| + |i - j| \tag{3}$$

The (minimum) spread associated with index i is then

$$S'_{new}(i) = \min_{j,\, j\neq i} [S''_{new}(i, j)] \tag{4}$$

The overall (minimum) spread is defined as

$$S_{new} = \min_{i} [S'_{new}(i)] \tag{5}$$

Proper termination of the TC's RSC constituent codes is very important for good performance at low error rates [7]. Some form of dual termination or dual tail-biting is recommended, as defined in [10,11] for example. With dual tail-biting, the absolute differences in (3) should be computed in a tail-biting sense. For these spread definitions, it can be shown that the theoretical maximum spread (with dual tail-biting) is $\text{floor}(\sqrt{2K})$. As an example, for a block length of K=512, the theoretical maximum spread is 32 (i.e. $S_{new}\leq 32$).

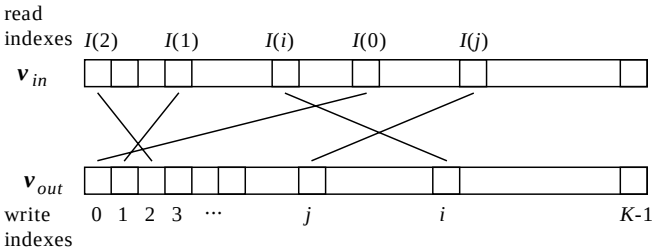

Fig. 2: Illustration of interleaver definition.

3 Dithered Relative Prime Interleavers

Figure 3 shows the approach used to design dithered relative prime (DRP) interleavers. The approach is well suited to dual tail-biting, which is the most difficult TC termination option to accommodate. The approach consists of three stages. First, the input vector, $\boldsymbol{v}_{in}$, is dithered (permuted locally) using a small read dither vector, $\boldsymbol{r}$, of length R. Vector $\boldsymbol{r}$ is a permutation of indexes 0 through R-1. Next, the resulting vector, $\boldsymbol{v}_a$, is permuted using an RP interleaver to obtain good spread. Finally, the resulting vector, $\boldsymbol{v}_b$, is dithered using a small write dither vector, $\boldsymbol{w}$, of length W, to generate the output vector $\boldsymbol{v}_{out}$. Vector $\boldsymbol{w}$ is a permutation of indexes 0 through W-1. The interleaver length, K, must be a multiple of both R and W. Note that short read and write dither vectors will not destroy the good spreading properties of an RP interleaver, but will tend to lower the spread somewhat. While a DRP interleaver could be implemented using the 3-stage process shown in Figure 3, this is not the recommended approach. The equivalent overall interleaver vector, $\boldsymbol{I}$, as illustrated in Figure 2, is determined next.

Let $\lfloor x \rfloor$ denote the floor(x) function and again let $[x]_m$ denote x modulo-m arithmetic. With these definitions, the

equations for the various DRP interleaver vectors shown in Figure 3 can be expressed as follows:

$$v_a(i) = v_{in}(I_a(i)), \qquad v_b(i) = v_a(I_b(i)),$$
$$v_{out}(i) = v_b(I_c(i)), \qquad i=0\ldots K\text{-}1 \qquad (6)$$

where

$$I_a(i) = R\lfloor i/R \rfloor + r(\lfloor i \rfloor_R), \quad i=0\ldots K\text{-}1 \qquad (7)$$

$$I_b(i) = \lfloor s + ip \rfloor_K, \qquad i=0\ldots K\text{-}1 \qquad (8)$$

$$I_c(i) = W\lfloor i/W \rfloor + w(\lfloor i \rfloor_W), \; i=0\ldots K\text{-}1 \qquad (9)$$

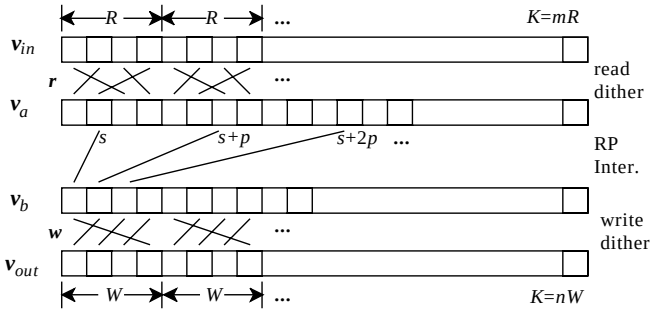

Fig. 3: Dithered relative prime (DRP) interleavers.

Thus, the input vector can be interleaved using

$$v_{out}(i) = v_{in}(I(i)), \quad i=0\ldots K\text{-}1 \qquad (10)$$

where the interleaver is completely defined by

$$I(i) = I_a(I_b(I_c(i))), \quad i=0\ldots K\text{-}1 \qquad (11)$$

All the indexes of $\boldsymbol{I}$ can be computed using equations (7), (8), (9), and (11).

A DRP interleaver can be stored by just storing $\boldsymbol{r}$, $\boldsymbol{w}$, s and p. This represents a significant reduction in storage, as compared to storing all K indexes, but further simplifications and reductions are possible. Let M be the least common multiple (LCM) of R and W. It can be shown that

$$I(\lfloor i + M \rfloor_K) = \lfloor I(i) + Mp \rfloor_K, \quad i=0\ldots K\text{-}1 \qquad (12)$$

It follows that the interleaver indexes can be computed recursively by cycling through M index increments. That is,

$$I(i) = \lfloor I(i-1) + P(\lfloor i \rfloor_M) \rfloor_K, \quad i=1\ldots K\text{-}1 \qquad (13)$$

where $I(0)$ and the M index increments in vector $\boldsymbol{P}$ are defined by (11) and

$$P(i) = \lfloor I(i) - I(i-1) \rfloor_K, \quad i=0\ldots M\text{-}1 \qquad (14)$$

Thus, all the indexes of $\boldsymbol{I}$ can be computed using the simple recursion in (13), and the interleaver can be stored by just storing $\boldsymbol{P}$. ($I(0)$ is arbitrary.) Further, equation (13) is simple enough to accommodate "on-the-fly" index generation, saving even more memory. In particular, this method works well with the circular buffer feature provided by most modern digital signal processors.

A few important properties are now explained further. Dual tail-biting is assumed for convenience. A rotational (modulo-K) shift in $\boldsymbol{v}_{in}$ or $\boldsymbol{v}_{out}$ does not affect the spread or distance properties of the TC. However, a rotational shift in $\boldsymbol{v}_a$ or $\boldsymbol{v}_b$ can affect the spread and distance properties. It can be shown that any shift in $\boldsymbol{v}_a$ or $\boldsymbol{v}_b$ is equivalent to shifting $\boldsymbol{v}_{in}$ and/or $\boldsymbol{v}_{out}$ and using a different value for s in the RP interleaver. Thus, the s parameter is sufficient for testing different shifts when searching for good interleavers. Consider the special case where R and W are relative primes. In this case we have $M=R\times W$. Thus, a small amount of dither (small values for R and W) can still force a large number of index increments, M. This is undesirable since M is also the resolution of the interleaver bank (i.e. K must be a multiple of M). There is also no benefit derived from trying different s values since all relative shifts between dither vectors $\boldsymbol{r}$ and $\boldsymbol{w}$ will occur for every value of s. At the other extreme we have the special case where $M=R=W$. This case offers the largest amount of dither for the smallest number of index increments, M, and the finest interleaver bank resolution. In this case, different results can be achieved for all shifts $s=0\ldots M\text{-}1$, and thus all of the different shift values are worth considering. This second case is more convenient and has generally been found to give better distance results. This is the only case considered further below. As an example, with $M=R=W=8$, only 8 index increments are required to both store and implement each interleaver, and the interleaver bank resolution is conveniently in bytes.

4 Example Distance Results

The lowest weight TC codewords are constructed from combinations of low input-weight (IW) patterns that lead to low-weight RSC codewords in both RSC constituent codes. It is important to determine which combinations of low IW patterns need to be considered. For example, certain combinations do not need to be considered because of high spread. A number of distance lower bounds were derived. The presentation of these bounds is beyond the scope of this paper. From these bounds it was concluded that the most important cases to test, and to try and improve, are: "IW2:2,2", "IW3:3,3", "IW4:22,22", "IW6:33,222", "IW6:222,33", and "IW6:222,222". The meaning of these case labels is as follows. Each case label contains 3 numbers. The first number is the total IW. The second and third numbers indicate the base pattern combinations before and after interleaving, where each digit is the IW of a base pattern. By definition, all base patterns correspond to valid RSC codewords and a base pattern cannot be decomposed into a number of smaller base patterns.

Distance measurement routines have been developed for all of these cases. For completeness, and because it was feasible, routines were also developed to handle the other IW4 cases, namely "IW4:4,4", "IW4:4,22", and "IW4:22,4". With these extra IW4 cases included the minimum measured distances are guaranteed to be the true minimum distances for all possible IW2, IW3, and IW4 cases. While the minimum distances for IW5 and IW6 cannot be guaranteed in general, the minimum measured distance for IW6 is sure to be the true minimum distance (over IW5 and IW6) for long blocks

with large spread. This is because all the other IW5 and IW6 cases improve as the spread increases.

Table 1 shows some example unpunctured (rate 1/3) distance results obtained for different block lengths, *K*, and number of index increments, *M*=*R*=*W*. The measured distances, *D*(IW), are a function of IW. Results are shown for input weights of 2, 3, 4, and 6. The spread, S_{new}, is also shown. As an example, consider the distance results with *K*=512. The *M*=8 interleaver is expected to perform the best for a code rate of 1/3, but the *M*=4 interleaver should also perform well when puncturing is used to achieve higher code rates.

Table 1: Example unpunctured (rate 1/3) distances for different block lengths, *K*, and number of increments, *M*.

K	***M***	S_{new}	***D***(2)	***D***(3)	***D***(4)	***D***(6)	min(***D***)
512	1	32	134	65	28	30	27*
512	2	32	134	61	36	38	36
512	4	29	66	65	52	38	38
512	8	24	58	57	44	46	44
1024	8	31	86	61	56	50	50
2048	8	57	90	93	60	54	54
4096	16	62	110	97	60	54	54
8192	16	107	>150	>100	72	58	58

* Upper bound for case "IW9:333,333" with *M*=1.

5 Simulation Results

Simulation results are presented for binary antipodal signaling (e.g. BPSK or QPSK) and a block length of *K*=512. Dual termination was used, as described in [10,11]. The TC used 8-state constituent codes, and the decoder used an enhanced maximum-log-*a-posteriori*-probability (max-log-APP) approach, with scaled extrinsic information, as described in [12,13,14]. It has been found that this decoding approach typically provides performance within 0.1 dB of true log-APP processing for 8-state codes. The maximum number of decoding iterations was set to 16. Early stopping was also used where the decisions before and after each half-iteration must agree 3 times in a row before stopping [14].

Figure 4 shows the packet error rate (PER) results for a block length of *K*=512 and a code rate of 1/3. (The nominal code rate is used for convenience. The exact code rate is slightly less due to the 6 termination bits included in the interleaver length, *K*.) Results are shown for the 4 DRP interleavers indicated in Table 1, with *K*=512 and *M*=1, 2, 4, and 8. For comparison, results are also shown for a random interleaver, the 3GPP interleaver, and a good HSR interleaver with post-processing to improve the distance spectrum. As expected, the random interleaver performs poorly, the 3GPP interleaver performs better, and the HSR interleaver performs the best. Not surprisingly, the DRP interleaver with *M*=1 (actually a simple RP interleaver) performs worse than the random interleaver (although it is expected to cross over at higher SNRs). There is a significant improvement with *M*=2, but performance is still a little worse than that for the 3GPP interleaver. Performance continues to improve with *M*=4 and *M*=8. Note that the performance with *M*=8 is essentially the same as that for the HSR interleaver. Figure 4 also shows Shannon's continuous-input sphere-packing bound for a rate 1/3 code [15]. Note that the HSR and DRP (*M*=8) interleavers both provide performance within 1 dB of this bound down to a PER of about 10^{-6}.

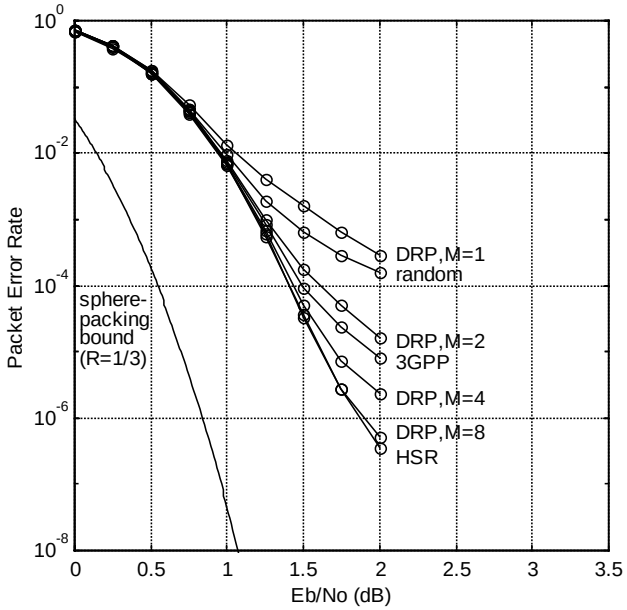

Fig. 4: PER results for *K*=512 and a code rate of 1/3.

Figure 5 shows the PER results for the same interleavers but with a code rate of 2/3. Most of the results were obtained without any data puncturing using the puncture masks (data, par1, par2) = (1, 0100, 0010), where a "0" indicates a punctured bit. The DRP (*M*=1) interleaver is as good as the 3GPP interleaver. The DRP (*M*=2) interleaver is better than the HSR interleaver, and performance continues to improve with *M*=4. Note that the *M*=4 result is slightly better than the *M*=8 result. This is not surprising given the unpunctured distance results shown in Table 1. The low IW cases (IW2, IW3, and IW4) are clearly dominating the performance. It should be noted that the HSR interleaver was not designed with puncturing in mind, but the DRP interleavers were.

A small amount of data puncturing, in exchange for more parity bits, can significantly improve the flare performance. This works because most of the distance, especially for the low IW cases, tends to come from the parity bits. It follows that the better the interleaver the better data puncturing works. There is a practical trade-off, however, as too much data puncturing can significantly degrade the convergence performance up top. Figure 5 shows results with 1/6 data puncturing for the two DRP interleavers with *M*=4 and *M*=8. The puncture masks were (data, par1, par2) = (111110, 001, 001). As can be seen, the flare performance is improved with only a small degradation up top. Note that the *M*=8 result is now slightly better than the *M*=4 result, reversing the trend

without data puncturing. This is expected because a decrease in the amount of parity puncturing tends to shift the emphasis away from the lowest IW cases. Even so, there is very little to choose from between these two DRP interleavers.

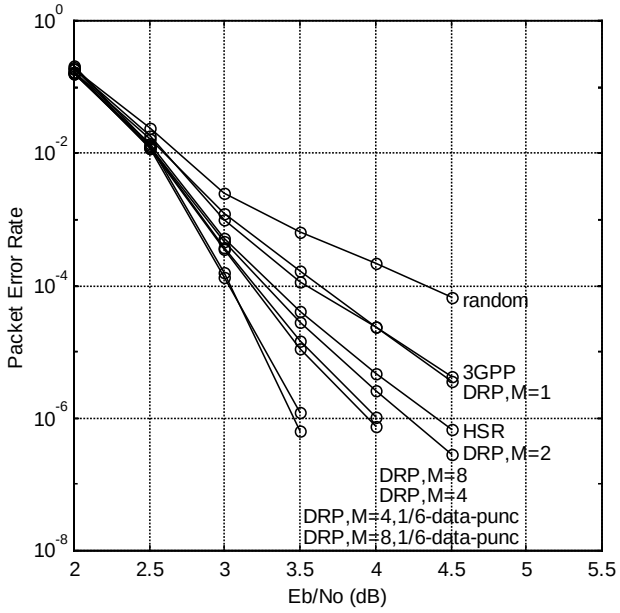

Fig. 5: PER results for K=512 and a code rate of 2/3.

6 Conclusions

Dithered relative prime (DRP) interleavers were introduced. These interleavers provide a good solution to the interleaver bank problem for Turbo-codes. The design is based on using a small read dither vector, ***r***, of length R, a high-spread RP interleaver with starting index s and index increment p, and a small write dither vector, ***w***, of length W. Distance testing is used to help select the dither parameters. A DRP interleaver can be stored by just storing ***r***, ***w***, s and p.

A DRP interleaver can also be stored using a vector, ***P***, containing M index increments, where M is the least common multiple of R and W. The interleaver is generated by repeatedly cycling through these M index increments. This method is simple enough to accommodate "on-the-fly" index generation, and works well with the circular buffer feature provided by most modern digital signal processors. The special case of M=R=W offers the largest amount of dither for the smallest number of index increments. This is important because M is also the resolution of the interleaver bank. As an example, with M=8, only 8 index increments are required to both store and implement each interleaver, and the interleaver bank resolution is conveniently in bytes.

The memory can be reduced further by selecting a small number of "good" dither combinations (***r***, ***w***, and s) and then just optimizing over p for each interleaver length. Good distance results have been obtained with as few as 8 dither combinations. With this approach, each interleaver in the bank can be stored by just storing 2 integers, the number of the best dither combination and the corresponding best p value found. In this case, the memory that is required to store a large bank of B interleavers is only about 2B integers.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes", Proceedings of ICC'93, Geneva, Switzerland, pp. 1064-1070, May, 1993.

[2] C. Berrou, and A. Glavieux, "Near Optimum Error Correcting Coding and Decoding: Turbo-Codes", IEEE Trans. On Comm., Vol. 44, No. 10, October 1996.

[3] 3GPP Document, 3G TS 25.212 v4.00 (2000-12), "3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Multiplexing and Channel Coding (FDD)".

[4] D. Divsalar and F. Pollara, "Multiple Turbo Codes for Deep-Space Communications", JPL, TDA Progress Report 42-121, May 15, 1995.

[5] D. Divsalar and F. Pollara, "Multiple Turbo Codes", MILCOM'95, pp. 279-285, November 6-8, 1995.

[6] S. Benedetto and G. Montorsi, "Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding Schemes", IEEE Trans. on Inform. Theory, Vol. 42, No. 2, pp.409-428, March 1996.

[7] S. Crozier, J. Lodge, P. Guinand, and A. Hunt, "Performance of Turbo Codes with Relative Prime and Golden Interleaving Strategies", Sixth International Mobile Satellite Conference (IMSC'99), Ottawa, Canada, pp. 268-275, June 16-18, 1999.

[8] J. Hokfelt, O. Edfors, and T. Maseng, "Interleaver Design for Turbo Codes Based on the Performance of Iterative Decoding", IEEE International Conference on Communications (ICC'99), Vancouver, BC, Canada, June 6-10, 1999.

[9] S. Crozier, "New High-Spread High-Distance Interleavers for Turbo-Codes", 20th Biennial Symposium on Communications, Kingston, Ontario, Canada, pp.3-7, May 28-31, 2000.

[10] P. Guinand and J. Lodge, "Trellis Termination for Turbo Encoders", Proc. 17th Biennial Symp. On Communications, Queen's University, Kingston, Canada, pp. 389-392, May 30-June 1, 1994.

[11] S. Crozier, P. Guinand, J. Lodge, and A. Hunt, "Construction and Performance of New Tail-Biting Turbo Codes", 6-th International Workshop on Digital Signal Processing Techniques for Space Applications (DSP'98), ESTEC, Noordwijk, The Netherlands, paper 1.3, September 23-25, 1998.

[12] S. Crozier, A. Hunt, K. Gracie, and J. Lodge, "Performance and Complexity Comparison of Block Turbo-Codes, Hyper-Codes, and Tail-Biting Convolutional Codes", 19-th Biennial Symposium on Communications, Kingston, Ontario, Canada, pp.84-88, May 31-June 3, 1998.

[13] K. Gracie, S. Crozier, A. Hunt, and J. Lodge, "Performance of a Low-Complexity Turbo Decoder and its Implementation on a Low-Cost, 16-Bit Fixed-Point DSP", The 10-th International Conference on Wireless Communications (Wireless'98), Calgary, Alberta, Canada, pp.229-238, July 6-8, 1998.

[14] K. Gracie, S. Crozier, and A. Hunt, "Performance of a Low-Complexity Turbo Decoder with a Simple Early Stopping Criterion Implemented on a SHARC Processor", International Mobile Satellite Conference (IMSC'99), Ottawa, Canada, June 16-18, 1999.

[15] S. Dolinar, D. Divsalar and F. Pollara, "Code Performance as a Function of Block Size", JPL, TMO Progress Report 42-133, May 15, 1998.