1-10-2005

On the performance of turbo codes with convolutional interleavers

Sina Vafi
University of Wollongong, sina@uow.edu.au

Tadeusz A. Wysocki
University of Wollongong, wysocki@uow.edu.au

On the performance of turbo codes with convolutional interleavers

Abstract

In this paper, some issues governing the block-wise performance of convolutional interleavers used in turbo codes are presented. Two different constructions of convolutional interleaver differing by the position of stuff bits in the interleaved data block are considered here. The performance assessment is based on the contribution of each weight to the overall code performance. For the given turbo code and each utilized interleaver, weight contribution is computed to finalize the code behavior in different signal to noise ratios. Simulations have been performed to verify the conducted analysis.

Keywords

turbo codes, convolutional interleavers, block-wise performance

Disciplines

Physical Sciences and Mathematics

Publication Details

This article was originally published as: Vafi, S, & Wysocki, TA, On the performance of turbo codes with convolutional interleavers, In K. Chung, S. Nordholm (Eds.), Asia-Pacific Conference on Communications, Perth, October 2005, 222-226. USA: IEEE. Copyright IEEE 2005.

2005 Asia-Pacific Conference on Communications, Perth, Western Australia, 3 - 5 October 2005.

On The Performance of Turbo Codes With Convolutional Interleavers

Sina Vafi, Tadeusz Wysocki
University of Wollongong Northfields Ave, Wollongong, NSW, Australia
{sv39,wysocki}@uow.edu.au

***Abstract*— In this paper, some issues governing the block-wise performance of convolutional interleavers used in turbo codes are presented. Two different constructions of convolutional interleaver differing by the position of stuff bits in the interleaved data block are considered here. The performance assessment is based on the contribution of each weight to the overall code performance. For the given turbo code and each utilized interleaver, weight contribution is computed to finalize the code behavior in different signal to noise ratios. Simulations have been performed to verify the conducted analysis.**

I. Introduction

Convolutional interleavers are introduced as the well-known non-block interleavers, which apply a lower number of memories compared to block interleavers and maintain synchronization with correspondence deinterleavers [1][2]. Because of their non-block characteristic, the continuous performance of the code is a vital assumption, when they are utilized as a turbo code component. In comparison to the block performance, this increases the complexity of the code analysis and decoding procedure with similar behavior to the block performance, especially for the Recursive Systematic Convolutional (RSC) codes with the low constraint length [3].

In order to contribute advantages of those interleavers in the conventionally considered turbo codes, a block-wise operation of the convolutional interleaver has been suggested through inserting a number of stuff bits at the end of each data block to return the interleaver memories to the zero state. However, certain number of stuff bits that appear in the end part of the interleaved data can be deleted to optimize overall number of stuff bits in the encoded data [4].

So far, we have verified the performance of turbo codes with non-optimized and optimized convolutional interleavers based on iterative decoding performance. Recently, we have presented a simple algorithm to calculate weight distribution of the code and its application to analyze the code behavior at the error floor region [5]. In this paper, we are utilizing the weight distribution of the turbo code obtained by this algorithm in investigating the effect of the optimized and non-optimized interleavers on the code performance.

To precisely determine the turbo code performance at the error floor region, it is important to know the percentage contribution of each weight of the code to the overall code performance. Similarly to most of the block interleavers, the non-optimized interleaver produces a free distance value with high multiplicities for the turbo code. In this case, a number of major contributing weights depend on the interleaver length and the considered signal to noise ratios. In contrast, applying an optimized convolutional interleaver provides a lower free distance value with low multiplicities such that only a few weights are dominated for the code performance. The effect of these low weights can be reduced by increasing the interleaver period. Therefore, it is expected that for an equal number of stuff bits, the optimized interleaver creates better performance for the code compared to the non-optimized interleaver. This has been confirmed by simulation results for different turbo codes.

The organization of the paper is as follows: Section 2 gives the analysis of turbo code with non-optimized and optimized convolutional interleavers based on computed weights contribution. In section 3, analytical results are verified by simulations of the iterative decoder performance. Finally, section 4, concludes the paper.

II. Turbo codes analysis with convolutional interleavers

A convolutional interleaver is constructed by T parallel lines which define its period. Generally, the difference between number of memories in two adjacent lines is considered constant and referred to as a space parameter of the interleaver. Figure 1.(a) shows the structure of the convolutional interleaver with the period $T = 4$ and the space value $M = 1$.

Due to the unequal number of memories in different lines, at the specific time, some data remains in the interleaver memories. In order to construct an isolated interleaved data block, enough zero stuff bits must be inserted at the end of each input bitstream returning the interleaver memories to the zero state. Figure 1.(b) shows the interleaved data obtained from an interleaver ($T = 4, M = 1$) with the length $L = 24$.

When the interleaver with the above structure is utilized in turbo codes, the existence of stuff bits reduces the channel bandwidth usage. Hence, an optimization can be carried out by

0-7803-9132-2/05/$20.00 ©2005 IEEE.

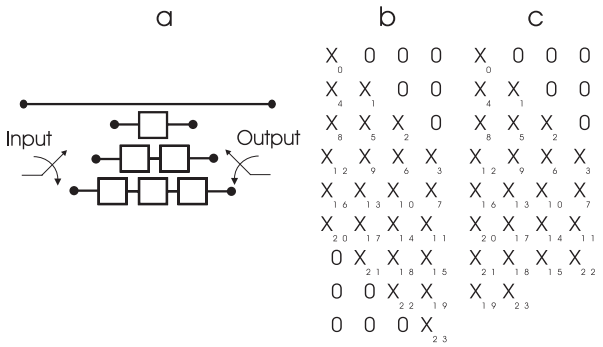

Fig. 1. Convolutional interleaver structure a) Interleaver with period $T = 4$ and space value $M = 1$ b) block interleaved data with length $L = 24$ c) Optimized interleaved data with the length $L = 24$.

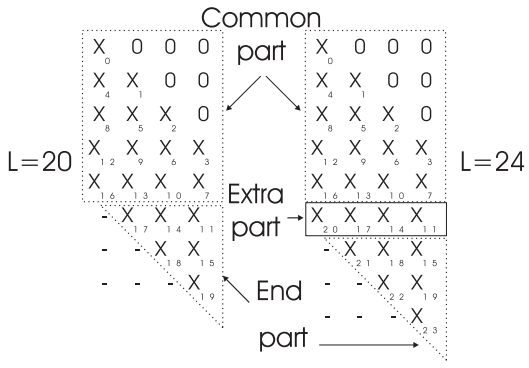

Fig. 2. Comparison of different parts of interleaved data at the output of the interleaver with different lengths and similar period and Rem(L, T) values,i.e. $T = 4$, Rem(20,4)=0, Rem(24,4)=0.

deletion of stuff bits located in the end part of the interleaved data [4].

Figure 1.(c) shows the optimized interleaved data of the Figure 1.(b). For turbo codes with the optimized convolutional interleaver, we have presented an algorithm to compute its weight distribution [5].

Since this interleaver has been basically constructed by the non-optimized convolutional interleaver, the obtained weights for the code with the non-optimized interleaver should also appear at the optimized interleaver with the similar multiplicities. Thus, the proposed weight distribution algorithm can be also applied for the code with the non-optimized interleaver. In both interleavers, depending on the interleaver period and the input bitstream length, interleaved data is terminated at one of the interleaver lines. The relevant line is determined by a Rem(L, T) value, which represents the reminder of $\frac{L}{T}$ operation. For different interleaver lengths with similar period and Rem(L, T) values, the interleaved data with the longer length L includes the interleaved data from the interleaver with the shorter length plus an extra

TABLE I

WEIGHT-2 DISTRIBUTION OF TURBO CODES (1,5/7) WITH THE OPTIMIZED AND NON-OPTIMIZATION INTERLEAVERS $(T = 10, M = 1)$ AND LENGTH $L = 512$.

weight	Optimized interleaver	Non-optimized interleaver
d	N_d	N_d
11	1	0
12	0	0
13	2	0
14	1	0
15	2	0
16	3	0
17	2	1
18	5	0
19	2	0
20	5	0
21	6	0
22	7	0
23	4	0
24	1	1
25	9	1
26	340	336
27	2	0
28	5	0
29	9	0
30	340	357

part, which corresponds to increasing of the length.

Figure 2 shows specified parts of two different interleaved data blocks with the similar period $T = 4$ and the Rem$(L, T) = 0$ value. The low weights are mainly produced by self-terminating input bitstreams, i.e. input bitstreams that automatically return the RSC encoder to the zero state without the effect of tail bits. Increasing the length of these patterns will not increase the weight of the codes. Indeed, increasing the length only increases number of cyclical shifts which consequently increases multiplicities of the code. As verified in [5], this issue is especially obvious for input bitstreams that simultaneously return both RSC encoders to the zero state. Therefore, it is expected that for different input bitstream lengths, interleavers with the equal period and $Rem(L, T)$ values generate similar weight distributions for the code. Based on this property we can compute the weight distribution of the code with short interleaver lengths and then extrapolate the achieved results for the assigned higher interleaver length.

Depending on the interleaver specifications, the minimum interleaver length applied in the algorithm would be different. This minimum value is achieved when all the interleaver memories have valid data. For non-optimized and optimized interleavers, it can be calculated by $(T(T-1)M) - Rem(L, T)$ and $\frac{T(T-1)M}{2}$ values, respectively. Table 1 gives weight-2 distribution of the 4-state turbo code (1, 5/7) with the non-optimized and the optimized interleaver $(T = 10, M = 1)$ and the length $L = 512$. The weights have been computed based on trellis termination and truncation for the first and the second RSC encoders, respectively.

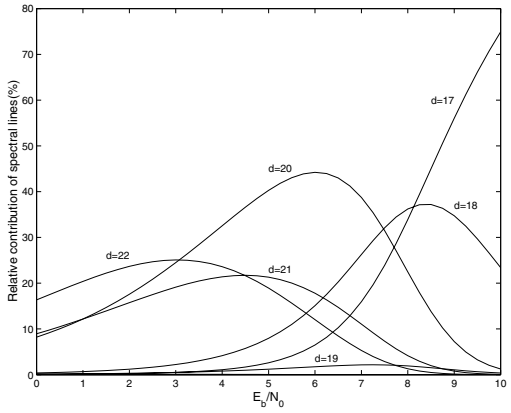

Fig. 3. Weight contributions to BER for the 4-state turbo code (1,5/7) with the non-optimized convolutional interleaver $(T = 10, M = 1)$ and length $L = 512$.

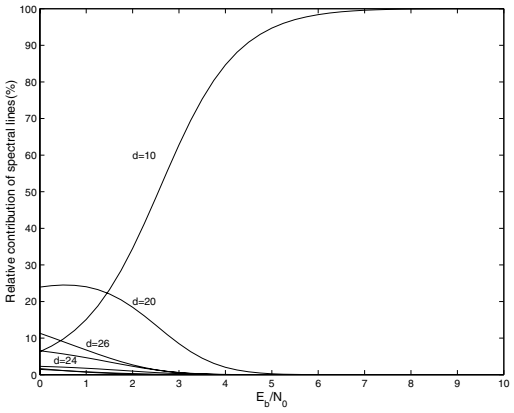

Fig. 5. Weight contributions to BER for the 4-state turbo code (1, 5/7) with the optimized convolutional interleaver $(T = 14, M = 1)$ and length $L = 512$.

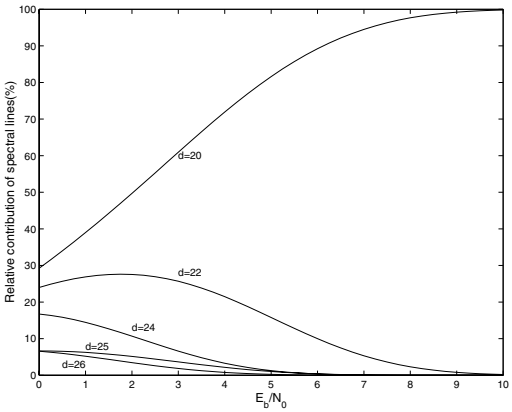

Fig. 4. Weight contributions to BER for the 4-state turbo code (1, 5/7) with the non-optimized convolutional interleaver $(T = 15, M = 1)$ and length $L = 1024$.

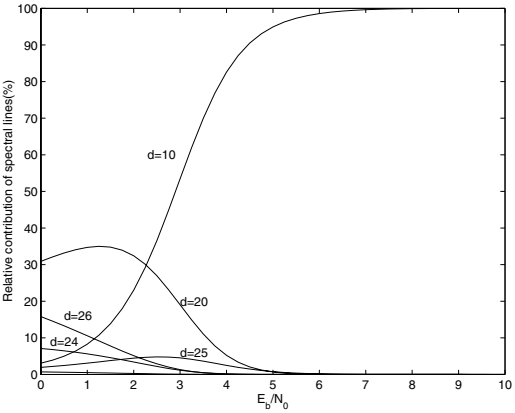

Fig. 6. Weight contributions to BER for the 4-state turbo code (1, 5/7) with the optimized convolutional interleaver $(T = 20, M = 1)$ and length $L = 1024$.

In the considered turbo codes structure, stuff bits have been inserted after trellis termination of the first RSC encoder since they do not have any effect on weights of the systematic and the first parity data. For the turbo code with the optimized interleaver, due to the deletion of zero stuff bits from the end part of the interleaved data, the distance between adjacent bits of the input bitstream in the interleaved data is reduced. Hence, the free distance value and some weights lower than the free distance value of the code with the non-optimized interleaver have been obtained. In fact, unlike non-optimized interleavers, optimized interleavers generate low weights with low multiplicities, while similar weights to the free distance value of the code with non-optimized interleavers are maintained. Comparing the obtained weights of the code from two different interleavers, it is concluded that the optimized interleaver will rearrange some of the input bitstreams related to the free distance value of the code with the non-optimized interleaver to other low weights with low multiplicities. In order to verify how the new generated weights affect the code performance, the contribution of each weight to BER in the code is computed and compared with the contribution of low weights in the code with the non-optimized interleaver. As indicated in [6], the contribution of each weight in the block-wise performance of the turbo code can be determined by the following equation:

$$P_d(\gamma_b) = \frac{N_d \tilde{\omega}_d}{L} Q(\sqrt{2dR\gamma}) \tag{1}$$

where R, $\gamma = \frac{E_b}{N_0}$, N_d and ω_d denote the code rate, the signal to noise ratio per information bit, number of multiplicities for weight d and average weight of information of weight d, respectively. Its relative contribution to the total BER is represented by:

$$\overline{P}_d(\gamma_b) = \frac{P_d(\gamma_b)}{\sum_d P_d(\gamma_b)} \tag{2}$$

Weight distribution for each code has been calculated by self-

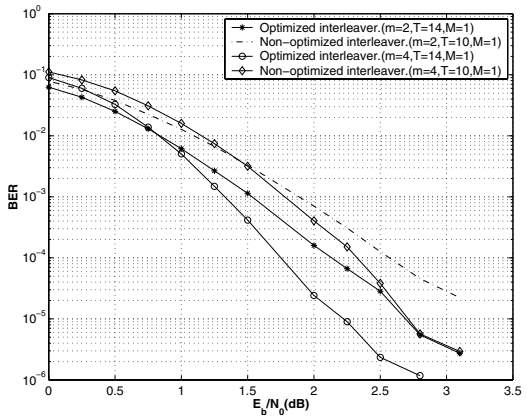

Fig. 7. Performance of full rate turbo codes with the interleaver length $L = 512$.

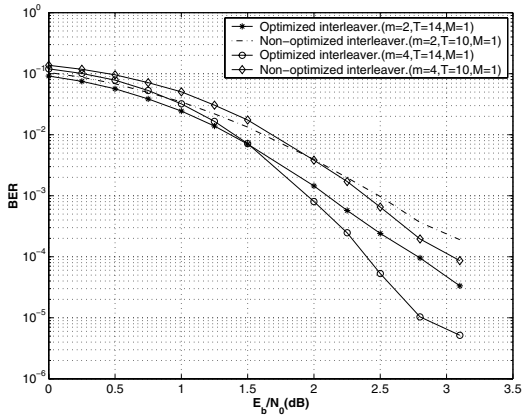

Fig. 8. Performance of half rate turbo codes with the interleaver length $L = 512$.

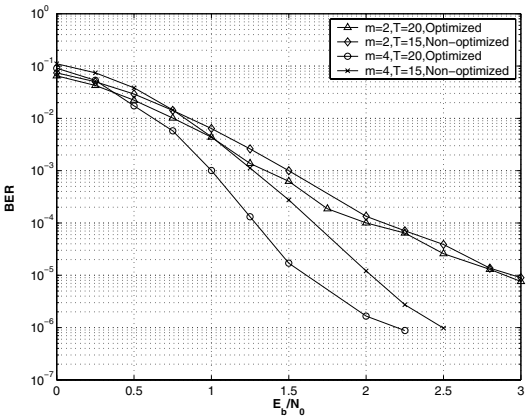

Fig. 9. Performance of full rate turbo codes with the interleaver length $L = 1024$.

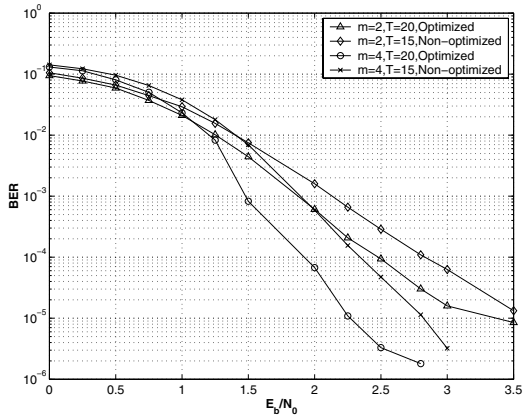

Fig. 10. Performance of half rate turbo codes with the interleaver length $L = 1024$.

terminating input bitstreams with weights no greater than 4. The interleaver specifications are designed in such a way that the overall number of generated stuff bits has no major influence on the code rate. As an assumption, the maximum number of stuff bits has been considered to be no more than 5% of the whole number of the encoded data. Figures 3 and 4 show contribution of the calculated weights for the turbo code (1,5/7) with the non-optimized convolutional interleaver lengths $L = 512$ and $L = 1024$, respectively. Similarly to previously obtained results of the turbo code performance with block interleavers, for the short interleaver lengths, many weights contribute to the code performance, while with increasing length, fewer weights are important.

The optimized interleavers are designed in a way that they produce a similar number of stuff bits with the applied non-optimized interleavers. Figures 5 and 6 show weights contribution of the 4-state turbo codes with the optimized interleaver $(T = 14, M = 1)$ and $(T = 20, M = 1)$ for the length $L = 512$ and $L = 1024$, respectively.

For the code with the interleaver length $L = 512$, the optimized interleaver $(T = 14, M = 1)$ has two major weights that contribute to the code performance, while the non-optimized interleaver has many weights. In this case, the obtained free distance value is 10 with 3 multiplicities, and its effect is almost dominant for all signal to noise ratios. In this code, weight 20 with 321 multiplicities has the second major contribution, while in the code with the non-optimized interleaver $(T = 10, M = 1)$ weight 20 has the highest contribution with 196 multiplicities for signal to noise ratios lower than 7dB.

For the higher interleaver length $L = 1024$, the code with the non-optimized interleaver $(T = 15, M = 1)$ has the free distance value 20 with 585 multiplicities whose effect is dominant for the code performance. Applying the optimized interleaver $(T = 20, M = 1)$ with higher period and a similar number of stuff bits, i.e. 190 bits, reduces the free distance to 10 with 3 multiplicities.

In addition, the code also has weight 20 with 846 multiplicities and lower contribution than the corresponding weights in Figure 4, especially in signal to noise ratios greater than 2 dB.

The graphs represent that although the code with the optimized interleaver generates some lower weights than the free distance value of the code with the non-optimized interleavers but due to their low multiplicities, they do not have a major influence on the code performance for low signal to noise ratios. Similar conclusions can be reached from the weight 20 of Figure 6. Since the distance of this weight to the free distance value is relatively high, its effect on the code performance is different from the identical weight in the code with the non-optimized interleaver.

The obtained graphs for the optimized interleaver imply that patterns $(00..011100..0)_L$ and $(00..0100100...0100100..0)_L$ with length L have major contributions to the code performance, which respectively produce the free distance value 10 and the weight 20.

From the obtained results, one can conclude that the optimized interleaver generates a low free distance value and the effect of other weights on the code performance is approximately voided, particularly in the medium to high signal to noise ratios. In addition, apart from the end part of the optimized interleaved data, where the free distance value is obtained, increasing the interleaver period will increase the distance between adjacent bits of the input bitstream in the interleaved data. This will create higher weights and reduce multiplicities of other low weights having major contributions to the code performance. Therefore, with a suitable selection of the interleaver characteristics, performance of the code can be improved.

III. Simulation results

Simulations have been conducted for the 4- and 16-state turbo codes $(m = 2, 1, 5/7)(m = 4, 1, 35/23)$, where m represents number of memories for the RSC encoders. The encoded data have been decoded using Soft Output Viterbi Algorithm (SOVA) as selected iterative decoding method by 8 iterations in the presence of Additive White Gaussian Noise (AWGN). Figure 7 shows performance of full rate 4- and 16-state turbo codes with the interleaver length $L = 512$. It can be observed that the optimized interleaver $(T = 14, M = 1)$ has improved performance of both codes by 0.5 dB. In addition, the 4-state code with the interleaver $(T = 14, M = 1)$ has better performance than the 16- state code with the interleaver $(T = 10, M = 1)$ in low signal to noise ratios with less complexity in the encoder structure and decoding process. Similar results have been achieved for the half rate turbo codes with similar interleavers and are presented in Figure 8. These results show that, the 4-state code with the optimized interleaver has 0.25 dB better performance than 16- state code with the non-optimized interleaver in all signal to noise ratios.

Similar turbo codes to the above examples have been examined with higher interleaver length, $L = 1024$. For the full rate 4-state turbo codes, as shown in Figure 9, the optimized interleaver $(T = 20, M = 1)$ slightly improves the code performance compared to the non-optimized interleaver $(T = 15, M = 1)$ for $\frac{E_b}{N_0} \leq$ 2dB, while for $\frac{E_b}{N_0} > 2dB$, both interleavers have similar performance. This behavior can be easily explained by the weight contributions presented in Figures 4 and 6. In these figures, the weight $\omega = 20$ has the highest contribution to BER. In the mentioned signal to noise ratio ranges, this weight in the code with the optimized interleaver has the slightly lower contribution than for the code with the non-optimized interleaver. Hence, better performance for the code with the optimized interleaver is expected. After $\frac{E_b}{N_0}$=2dB, the contribution of this weight is replaced by weight 10 with low multiplicity, which creates similar performance to the weight 20 with high multiplicity. However, for the 16- state turbo code, the optimized interleaver has improved the code performance by 0.5 dB. Simulations of the half rate code have been illustrated in Figure 10. In contrast to the results obtained for the full rate codes, the optimized interleavers have improved their performances from 0.25 to 0.5 dB.

From the conducted simulations, it can be concluded that in the case of similar numbers of stuff bits, the optimized interleaver leads to better performance of the turbo code. This is mainly achieved due to the existence of a free distance value with low multiplicities, that contributes the most to the code performance. The proposed optimized interleaver can be designed such that the higher free distance value for the code is produced, while other properties are maintained. For this purpose, some modifications to this interleaver have been suggested in [7][8].

IV. Conclusions

In this paper, an analysis of the turbo codes with two different models of convolutional interleavers has been presented based on the contribution of each weight to the code performance. For different codes, the results obtained from the proposed analysis have been verified by several simulations to select the suitable interleaver for the required code.

References

[1] G.D.Forney, "Burst-correcting codes for the classic bursty channel," *IEEE Trans. on Commun.*, vol. COM-19, pp. 772–781, Oct. 1971.

[2] E.K.Hall and G.Wilson, "Stream-oriented turbo codes," *IEEE Trans. on Inform.Theory.*, vol. 47, no. 5, pp. 1813–1831, July 2001.

[3] S. Benedetto and G. Montorsi, "Performance of continuous and blockwise decoded turbo codes," *IEEE Communications Letters*, pp. 77–79, May 1997.

[4] S. Vafi and T. Wysocki, "Iterative turbo decoder design with convolutional interleavers," *4th International Symposium on CSNDSP,Newcastle,UK.*, pp. 124–127, July 2004.

[5] S.Vafi and T.Wysocki, "A simple method for approximation of weight distribution of turbo codes with convolutional interleavers," *Submitted to IEE Proceedings Communications.*

[6] W.Feng, J.Yuan, and B.Vucetic, "A code-matched interleaver design for turbo codes," *IEEE Trans. on Commun.*, vol. 50, no. 6, June 2002.

[7] S.Vafi and T.Wysocki, "Performance of convolutional interleavers with different spacing parameters in turbo codes," *Australian Communications Theory Workshop*, pp. 8–13, Feb. 2005.

[8] S.Vafi and T.Wysocki, "Generalized convolutional interleaver and its performance in turbo codes," *Submitted to IEEE Communications Letters.*