A Way to Combat the Sub-optimality of Turbo Decoding of Product Codes

Nadine Chapalain[1], Arnaud Guéguen[1], Damien Castelain[1], Ramesh Pyndiah[2].

[1]Mitsubishi Electric ITE, Immeuble Germanium, 80, Avenue des Buttes de Coësmes, 35700 Rennes - FRANCE
[2]ENST Bretagne, Technopôle, Brest Iroise, BP.832, 29285 Brest - FRANCE

Abstract

In this paper, the sub-optimality of iterative decoding of BCH product codes also called Block Turbo Codes (BTC) is investigated. Lower bounds on Maximum Likelihood (ML) decoding performances for Packet Error Rate (PER) and Bit Error Rate (BER) are given in order to evaluate the optimality of the iterative decoding algorithm. On an AWGN (Additive White Gaussian Noise) channel, simulations show that the turbo decoding of product code is sub-optimal for long codes, even when the elementary codes are decoded in an optimal way. We propose to apply after turbo decoding a scheme to combat this sub-optimality. This scheme is described, the performance gain is then evaluated on Gaussian and Rayleigh channels.

1 Introduction

Turbo-codes have attracted much attention these years thanks to their performance approaching Shannon's theoretical limit. Their very good performance stem both from their good weight distribution and from their ability to be decoded in a nearly optimal way through an iterative decoding process called turbo decoding. Turbo codes are included in more and more standards such as the third generation of mobile systems [1] and more recently they have been chosen as an option in the IEEE 802.16 [2] and HIPERACCESS standards [3]. Convolutional Turbo Codes (CTC) were introduced in 1993 [4], they are based on the parallel concatenation of two convolutional codes separated by a pseudo-random interleaver and iterative decoding of the resulting code using Soft-Input-Soft-Output (SISO) decoders. The concept was extended to the iterative decoding of product codes also called Block Turbo Codes (BTC) [5] made of the serial concatenation of BCH codes. This paper deals with BTC which are particularly attractive for applications requiring high code rate and present good free distance even for small block sizes unlike CTC. The aim of this paper is first to quantify the sub-optimality of two iterative decoding algorithms of product codes compared to ML decoding. One is based on a sub-optimal BCH decoding algorithm and the other one is based on an optimal BCH decoding algorithm. Secondly, the aim is to provide a way to improve the turbo decoding process in those cases. In this view, after a recall on the iterative decoding of product codes in section II, a lower ML bound is given in section III that accurately predicts the ML performance of the studied product code at high Signal to Noise Ratio (SNR) on an AWGN channel. The performances of BTC with the two iterative decoding algorithms are compared to this lower ML bound. In section IV, a scheme that enables to partly overcome the sub-optimality of turbo decoding is then presented and evaluated on AWGN and Rayleigh channels, and the results are discussed.

2 Iterative decoding of product codes

2.1 Product codes

Let P be the product code resulting from the serial concatenation of two linear block codes $C_1(n_1,k_1,d_1)$ and $C_2(n_2,k_2,d_2)$, where n_i, k_i and d_i are respectively the code word length, the number of information bits per code word and the Hamming distance of code C_i, i=1,2. The product code $P=C_1 \otimes C_2$ is represented by a matrix obtained by encoding the k_2 rows of k_1 information bits by code C_1, then encoding the n_1 resulting columns of the matrix by code C_2. By construction, all the n_2 rows of the matrix are code words of code C_1 and all the n_1 columns are code words of code C_2. The product code parameters are the product of the elementary code parameters: $n=n_1.n_2$, $k=k_1.k_2$, $d_{min}=d_1.d_2$, and the code rate is given by $R=R_1.R_2$ where R_i is the code rate of code C_i, i=1,2.

2.2 Iterative decoding of product codes

Iterative decoding of product codes consists in decoding successively the rows and the columns of the matrix and iterating the procedure. To be efficient, the constituent code decoder has to work on soft inputs and deliver soft outputs that evaluate the reliability associated to the decision on each bit. From the soft outputs generated by the decoding of one dimension (the rows or the columns), an extra information called extrinsic information is extracted and used to modify the associated soft inputs - the a priori informations - of the next decoder. A sub-optimal and an optimal ML decoding algorithm of BCH constituent codes are evaluated in this paper as the elementary SISO decoders of the turbo decoder. These two algorithms are presented below.

2.2.1 The sub-optimal algorithm

A SISO decoder was proposed by R. Pyndiah [5] that is based on the Chase algorithm [6] to compute the approximated Log A Posteriori Probability ratios (LAPP). The Chase algorithm intends to provide the code word that is the closest to the received sequence according to the Euclidian distance criterion. For that purpose, a subset of code words is generated, which in most cases will contain all the nearest code words from the received word. In order to compute the soft output, the two nearest code words with opposite component in position j are searched in this subset. The nearest code word will be the approximated ML decision D and the second one is the competing code word C. The soft output component in position j is calculated as the difference between the two following squared Euclidian distances: M^{D}, the squared Euclidian distance between the received sequence R=(r_0,…, r_j,…, r_{n-1}) and the ML decision D=(d_0,…, d_j,…, d_{n-1}), and M^{C}, the squared Euclidian distance between the received word R and the competing word C=(c_0,…, c_j,…, c_{n-1}). The relation below calculates the corresponding extrinsic information component w_j in position j.

$$w_j = \left(\frac{\left(M^{\mathrm{C}} - M^{\mathrm{D}}\right)}{4} \right) d_j - r_j \qquad (1)$$

In some cases there is no competing code word with opposite component in position j in the subset and the extrinsic information is then estimated by a predefined value, which is less accurate.

2.2.2 The optimal algorithm

The optimal decoding algorithm of the constituent block codes is a Maximum A Posteriori (MAP) algorithm that minimizes the symbol error probability and was proposed by L.E. Nazarov [7]. This algorithm uses Fast Hadamard Transform (FHT) for the calculation of the soft information with the dimension of the FHT basis being determined by the dimension of the dual code matrix. The decoding rule is exhaustive in the sense that every word in the constituent dual code is used in the decoding process.

3 On the optimality of the iterative decoding

3.1 The ML bound

Error bounds enable to predict the performance of a code at high SNR, at error rates that cannot be reached through simulations, and also to account for the efficiency of the decoding scheme.
The Union bound is an upper bound reflecting ML decoding performance that is very accurate at high SNR. The union bound on the Packet Error Rate (PER) assuming transmission over an AWGN channel is given by:

$$P_p(e) \le \frac{1}{2} \sum_{w=d_{\min}}^{n} A_w^P .erfc\sqrt{w\frac{RE_b}{N_0}} \qquad (2)$$

Where A_w^P is the number of product code words of weight w.
A product code has the particularity to have a large number of code words at minimum distance and to have the following non-zero multiplicities at weights quite far from the minimum distance. Taking into account this particularity, the first term is the one that influences the most the Union bound and the terms of higher orders can be neglected for large SNR. This results in the following lower bound [8, 9].

$$P_p(e) = \frac{1}{2} A_{d_{\min}}^P .erfc\sqrt{d_{\min}\frac{RE_b}{N_0}} \qquad (3)$$

$$\text{with } A_{d_{\min}}^P = A_{d_1}^{C_1} . A_{d_2}^{C_2} \qquad (4)$$

Where $A_{d_1}^{C_1}$ and $A_{d_2}^{C_2}$ are respectively the number of code words at minimum Hamming distance d_1 and d_2 of C_1 and C_2.
The corresponding lower ML bound on the Bit Error Rate (BER) is given by:

$$P_b(e) = \frac{d_{\min}}{n} P_p(e) = \frac{1}{2}\frac{d_{\min}}{n} A_{d_{\min}}^P .erfc\sqrt{d_{\min}\frac{RE_b}{N_0}} \qquad (5)$$

These two lower bounds are still very accurate at high SNR.

3.2 Performance evaluation

The performances are evaluated over an AWGN channel with the sub-optimal and the optimal BCH decoding algorithms, for product codes generated by two identical extended BCH codes (i.e. C_1=C_2). The BER versus E_b/N_0 after four decoding iterations is plotted for the BTC $(16,11,4)^2$ and $(64,57,4)^2$ and compared with their corresponding lower bound (Figures 1 and 2). Indeed very low gain is achieved beyond the fourth iteration.

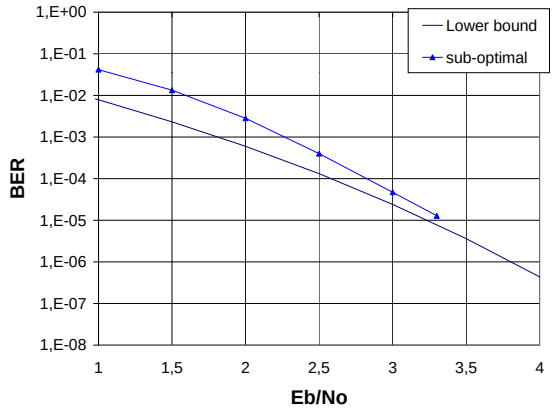

Figure 1: BER vs. E_b/N_0 for the BTC $(16,11,4)^2$ after four iterations with the sub-optimal Chase-Pyndiah algorithm.

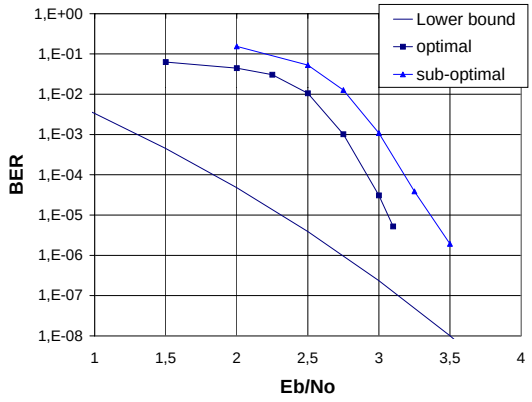

Figure 2: BER versus E_b/N_0 for the BTC $(64,57,4)^2$ after four iterations with the two BCH algorithms.

If the decoding process is optimal in the sense of Maximum Likelihood, the simulation curves shall stick to the ML bound. For the BTC $(16,11,4)^2$, iterative decoding with the sub-optimal Chase-Pyndiah algorithm is sufficient to reach the ML bound quite rapidly. For the BTC $(64,57,4)^2$, the E_b/N_0 must be sufficiently high for the 'turbo effect' to appear. Even if the BER decreases rapidly in the waterfall region for the optimal and sub-optimal algorithms, the simulation curves converge towards the ML bound only at high E_b/N_0, for very low BER. For the optimal algorithm the simulation curve seems to converge towards the ML bound for a BER less than 10^{-7} and for the sub-optimal algorithm it seems that the simulation curve will never reach it.

For BTC, even if the minimum distance is high, when the code rate increases, the iterative decoding appears to be sub-optimal since performances do not stick to the ML bound. When the product of $d_{\min}$ and R is high that is when the asymptotic gain G_a is high, the BTC iterative decoding is sub-optimal for low SNR and converges towards the ML bound only for low error rates at high SNR. The asymptotic gain is given by:

$$G_{a(dB)} = 10 * \log_{10}(R.d_{\min}) \qquad (6)$$

The sub-optimality of turbo decoding of BTC can be at least partly explained by graph theory [10, 11]. Indeed, representing a BTC on a Tanner graph, it shows that the resulting network has a large number of very short cycles. Therefore, the information produced by neighbouring nodes along the decoding process remains very correlated with each other and, as predicted by graph theory, the decoding algorithm will consequently fail to produce the true a priori probabilities (APP). Besides, as the number of short cycles increases with the BTC size, this may explain that the turbo decoding performances are all the more sub-optimal as the code is long as shown on Figures 1 and 2.

4 A scheme to combat the sub-optimality of turbo decoding

4.1 Description of the post-processing scheme

The turbo decoding of product codes with high asymptotic gain G_a converges towards the ML bound at high SNR. The scheme that is proposed below aims at improving the turbo decoding to get closer to the ML bound at lower SNR. The basic principle is the following [12]: when errors are detected in the decoded block after a certain number of iterations, the decoded erroneous binary sequence is turbo encoded and modulated. The resulting sequence is then multiplied by a coefficient that is in the order of magnitude of 10^{-2}, and subtracted from the input sequence. The turbo decoding process is then applied over with this modified input sequence. Again, if residual errors are found in the decoded block, the same post-processing scheme of re-encoding, re-modulation and subtraction is applied, and so on until there is no error left or until a maximum number of

post-processing iterations is reached. The proposed scheme is depicted on Figure 3. The switch on the left-hand side is down when the received sequence is going to be turbo decoded for the first time, and up when post-processing is performed.

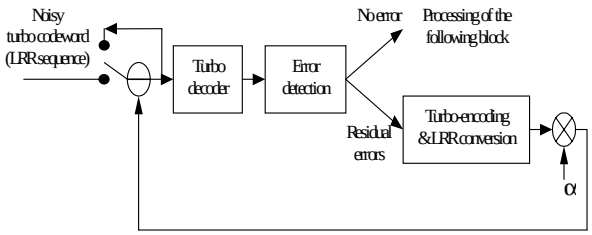

Figure 3: iterative post-processing scheme

The underlying idea is that the contribution of the erroneous decoded sequence is partly subtracted from the received input sequence if the turbo decoder fails to converge towards the transmitted code word. Consequently, when turbo decoding is performed on the modified sequence, the erroneous code word towards which the turbo decoder first converged gets farther away from the received sequence, so that the turbo decoder will tend towards another code word which will hopefully be the transmitted one. If the residual errors were due to the sub-optimality of the turbo decoding, the probability of error free convergence at the next step is increased.

Error detection [13] can be performed after the iterative decoding of product codes by only checking if all the rows and the columns of the decision matrix [D] are respectively code words of C_1 and C_2. If that is the case, [D] is a code word of P and no error is detected. Thus, no additional check bits are required for error detection. However in two situations, it happens that this scheme fails to correctly predict if errors remain. First, when the decision matrix [D] is not equal to the transmitted code word [E] but is a code word of P. In this case, the matrix [D] contains residual errors but those errors are not detected. Second, when [D] is not a code word of P, but the residual errors are only located on the redundancy part of the matrix. All errors on the information bits have been corrected but an alarm is generated by the detection scheme.

4.2 Performance evaluation

4.2.1 On Gaussian channel

The proposed scheme is applied to the BTC $(32,26,4)^2$ and $(64,57,4)^2$ with the sub-optimal and the optimal BCH algorithms. The detection scheme is the one presented above. The performances are evaluated on a AWGN channel with a maximum of 20 post-processing iterations. A post-processing iteration is the turbo decoding of the product code with four iterations. The results are compared with the lower ML bound in terms of BER and PER.

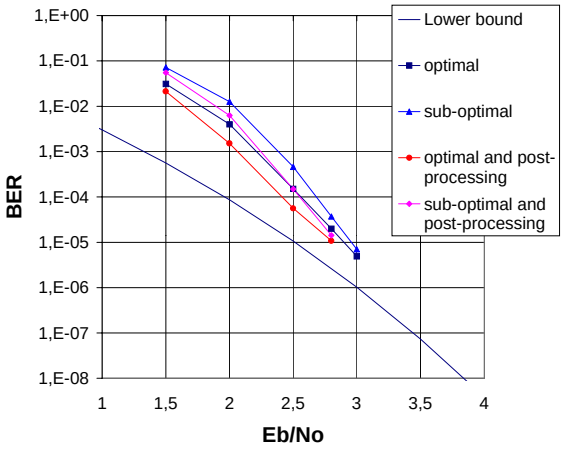

Figure 4: BER versus E_b/N_0 for the BTC $(32,26,4)^2$ after four iterations with post-processing.

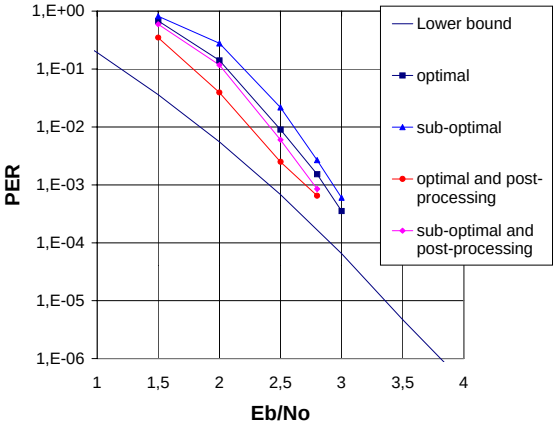

Figure 5: PER versus E_b/N_0 for the BTC $(32,26,4)^2$ after four iterations with post-processing.

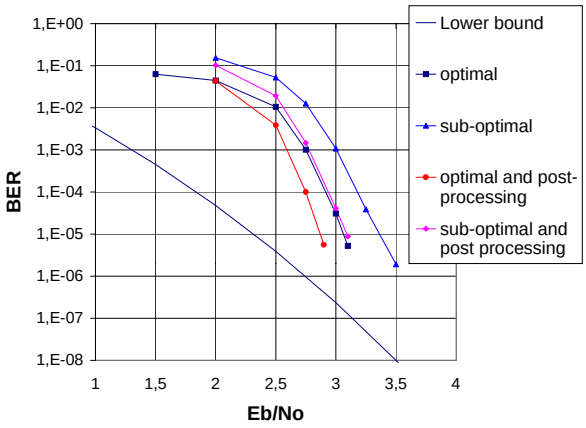

Figure 6: BER versus E_b/N_0 for the BTC $(64,57,4)^2$ after four iterations with post-processing.

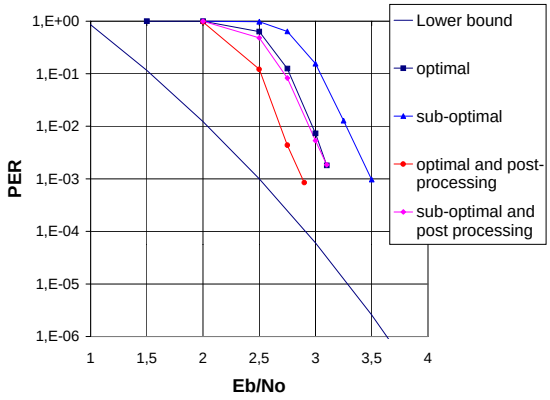

Figure 7: PER versus E_b/N_0 for the BTC $(64,57,4)^2$ after four iterations with post-processing.

The optimal BCH algorithm always performs better than the sub-optimal algorithm in terms of BER and PER. For the BTC $(32,26,4)^2$ at high E_b/N_0, the simulation curve of the Chase-Pyndiah algorithm tends towards the performance of the optimal algorithm while for the BTC $(64,57,4)^2$ the two curves are parallel. With the post-processing scheme, for the BTC $(32,26,4)^2$ the gain is in the order of 0.1 dB at a BER of 10^{-5} and 0.15 dB for a PER of 10^{-3}. When the code is longer, for the BTC $(64,57,4)^2$ the gain provided by the post-processing scheme increases and is in the order of 0.25 dB for the optimal algorithm and 0.3 dB for the sub-optimal algorithm. The tables below summarize the results.

	BER=10^{-5}	PER=10^{-3}
Optimal	0.1 dB	0.14 dB
Sub-optimal	0.12 dB	0.15 dB

Table 1: Gain provided by the post-processing algorithm with 20 iterations for the BTC $(32,26,4)^2$.

	BER=10^{-5}	PER=10^{-3}
Optimal	0.23 dB	0.23 dB
Sub-optimal	0.28 dB	0.33 dB

Table 2: Gain provided by the post-processing algorithm with 20 iterations for the BTC $(64,57,4)^2$.

Note that using this error detection scheme results in increasing the BER and PER as compared as using an ideal error detection scheme. However, regarding the gain obtained with the post-processing scheme, this error detection scheme seems accurate enough.

4.2.2 On Rayleigh channel

The post-processing scheme is applied to the $(64,57,4)^2$ with the sub-optimal and the optimal BCH algorithms. The performances are evaluated on a Rayleigh channel with maximum 20 post-processing iterations and also with 4 and 2 post-processing iterations in order to reduce the decoding delay. The detection scheme is the one presented above.

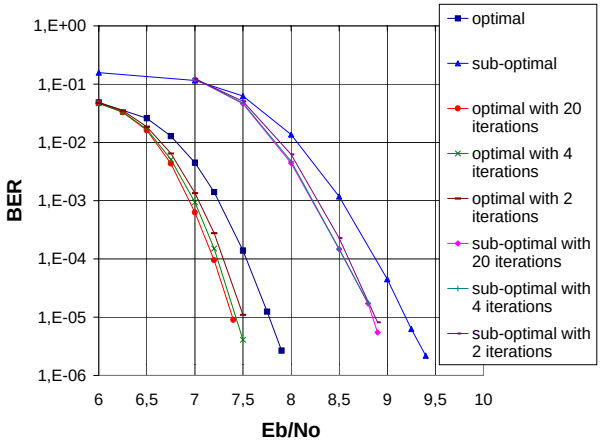

Figure 8: BER versus E_b/N_0 for the BTC $(64,57,4)^2$ with 20, 4 and 2 post-processing iterations and compared with classical turbo decoding.

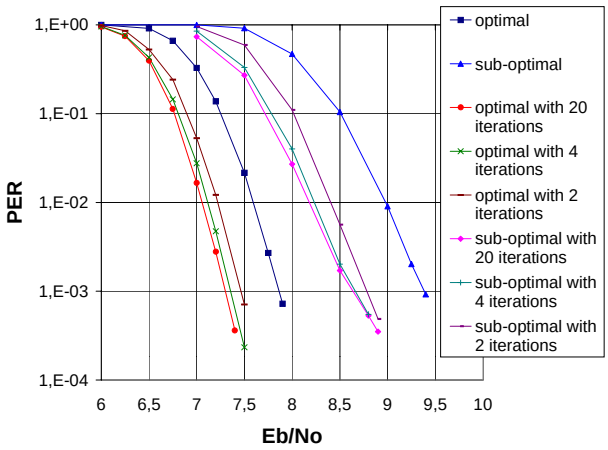

Figure 9: PER versus E_b/N_0 for the BTC $(64,57,4)^2$ after 20, 4 and 2 post-processing iterations and compared with classical turbo decoding.

On the Rayleigh channel, the turbo decoding of the product code $(64,57,4)^2$ with the optimal BCH algorithm provides a significant gain of around 1.5 dB in terms of BER and PER as compared with the sub-optimal BCH algorithm.
For the BTC $(64,57,4)^2$ the gain provided by the post-processing scheme with 20 iterations is higher on the Rayleigh channel than on the Gaussian channel. It is in the order of 0.35 dB in terms of BER and exceeds 0.56 dB in terms of PER.
For low error rates, this post-processing scheme is most of the time performed and does not always converge towards the transmitted code word after the 20 iterations. For high error rates, the post-processing scheme converges most of the time towards the transmitted code word with only 2 or 4 iterations. For

some applications, the decoding delay introduced by the post-processing scheme with 20 iterations could be too high, but 2 to 4 post-processing iterations could be acceptable. The performance evaluation of the BTC$(64,57,4)^2$ on the Rayleigh channel with the post-processing scheme with 4 and 2 iterations show that there is only a negligible degradation for 4 iterations and a small degradation for 2 iterations for both algorithms. The table 3 below summarizes the results.

	Nber of iterations	BER=10^{-5}	PER=10^{-3}
Optimal	20	0.37 dB	0.56 dB
Sub-optimal	20	0.34 dB	0.75 dB
Optimal	4	0.34 dB	0.51 dB
Sub-optimal	4	0.34 dB	0.72 dB
Optimal	2	0.24 dB	0.4 dB
Sub-optimal	2	0.31 dB	0.62 dB

Table 3: Gain provided by the post-processing algorithm with 20, 4 and 2 iterations for the BTC $(64,57,4)^2$ on the Rayleigh channel.

5 Conclusion

The lower ML bound of product codes is used to investigate the optimality at high SNR of block turbo decoding with a sub-optimal and an optimal BCH decoding algorithm. The performances loss is quantified for the BTC $(16,11,4)^2$, $(32,26,4)^2$ and $(64,57,4)^2$. The BTC $(16,11,4)^2$ with code rate R=0.47 converges rapidly towards the ML bound while the BTC $(64,57,4)^2$ with code rate R=0.79 converges at only high SNR for very low error rates. Iterative decoding is sub-optimal for product codes with a high code rate, and more precisely with a high asymptotic gain. A scheme based on error detection and re-encoding is proposed to partly overcome this loss. The error detection scheme does not require transmitting any additional bits. For the BTC $(64,57,4)^2$ with 20 post-processing iterations, the performance gain at a PER of 10^{-3} is up to 0.33 dB on a Gaussian channel and up to 0.75 dB on a Rayleigh channel. On a Rayleigh channel, the turbo decoding of the product code $(64,57,4)^2$ with the optimal BCH decoding algorithm performs around 1.5 dB better in terms of BER and PER compared to the turbo decoding with the sub-optimal BCH decoding algorithm. For applications with delay constraints, the number of post-processing iterations can be reduced to 4 or 2 still leading to interesting performance gain.

6 Literature

[1] 3GPP TSG WG1 25.212, v3.1.0: “Multiplexing and Channel Coding”.

[2] Draft physical layer specification for the 802.16.1 air interface standard, IEEE802.16.1p-00/07r2.

[3] Minutes of the HIPERACCESS PHY group 19.5th meeting of the ETSI project BRAN, Aug. 2000.

[4] C. Berrou, A. Glavieux, A. and P. Thitimajshima: “Near Shannon limit error-correction coding: Turbo codes”, in Proc. IEEE ICC’93, Geneva, Switzerland, pp.1064-1070, May 1993.

[5] R. Pyndiah, A. Glavieux, A. Picart and S. Jacq: "Near optimum decoding of product codes", in Proc. of IEEE GLOBECOM'94, San Francisco, USA, vol.1/3, pp. 339-343.

[6] D. Chase: “A class of algorithm for decoding block codes with channel measurements information”, IEEE Trans. inform. Theory, vol IT-18, pp. 170-182, Jan 1972.

[7] L.E. Nazarov, V.M. Smolyaninov: "Use of Walsh-Hadamard transformation for optimal symbol-by-symbol binary block code decoding", Electronics Letters, vol. 34, n°3, pp. 261-262, Feb. 1998.

[8] L.M.G.M. Tolhuizen and C.P.MJ. Baggen: “On the weight enumerator of product codes”, Discrete mathematics, 106/107, PP. 483-488, 1992.

[9] L. Tolhuizen, S. Baggen, E. Helstra-Nowacka: “Union bounds on the performance of product codes”, in Proc. IEEE ISIT’98, Cambridge, MA, USA, pp. 267, Aug. 1998.

[10] R. J. McElliece, D.J.C. McKay, J.F. Cheng: “Turbo decoding as an instance of Pearls’s, Belief propagation algorithm”, IEEE J. on selected areas in Communications, V.16 Number 2, Feb. 1998.

[11] E. Fabre, A. Guyader: “Dealing with short cycles in graphical codes”, paper submitted to ISIT 2000 (Session: Codes on Graph).

[12] A. Guéguen, D. Castelain: “On the suboptimality of turbo decoding in the case of short frame turbo codes and a way to improve the performance”, in Proc. European Wireless 2000, Dresden, Germany, pp. 307-312.

[13] R. Pyndiah: “Iterative decoding of product codes: Block Turbo Codes”, in Proc. IEEE Int. Symp. on Turbo codes & related topics, pp.71-79, Sept. 1997.