An improved iterative decoding algorithm for block turbo codes

Massinissa Lalam, Karine Amis Cavalec, Dominique Leroux, Dongning Feng, Jinhong Yuan

An improved iterative decoding algorithm for block turbo codes

M. LALAM, K. AMIS, D. LEROUX
GET-ENST Bretagne, TAMCIC (UMR 2872 CNRS)
SC Department, Technopole Brest Iroise
CS 83818 - 29238 Brest Cedex 3, France
Email: massinissa.lalam@enst-bretagne.fr

D. FENG, J. YUAN
School of Electrical Engineer and Telecommunications
University of New South Wales
Australia
Email: d.feng@student.unsw.edu.au

***Abstract*— Since the introduction of the block turbo code (BTC) concept, several soft-input / soft-output (SISO) algorithms have been used in order to softly decode product codes. The classical Chase-Pyndiah algorithm seems to be one with the best trade-off between complexity and performance, especially for low error correction capability t (typically 1 or 2) where it is nearly optimal. However, as an algebraic decoding-based algorithm, the lack of codeword diversity is one of its weakness for BTCs with higher error correction capability and/or non binary BTCs. In this paper, we propose an improved iterative decoding algorithm for BTCs. We present a simple sliding encoding-window (SEW) based decoding algorithm which exploits the cyclic and systematic properties of RS or BCH codes. By adding the SEW algorithm to a classical algebraic decoding method, the proposed decoder can easily generate a list of codewords that are close to the decoded codeword. With the codeword diversity, we can compute more reliable soft output necessary in the turbo decoding process. Monto-Carlo simulations of binary and non-binary BTCs are carried out on Gaussian channels. The results show that the algorithm can improve the error performance up to 1.5 dB relative to the conventional Chase-Pyndiah decoder, while the increase in complexity due to the encoding is minor since it is a low-cost process compared to that of algebraic decoding. Compared to the other encoder-based decoding algorithms in the literature, the proposed algorithm has the advantage that there is no requirement to recompute the generator or parity-check matrix by using Gaussian elimination operations, thus a lower computational complexity.**

I. Introduction

Concatenated block coding technique was introduced by P. Elias in 1954 [1]. In 1994, R. Pyndiah introduced binary (BCH codes [2]) and non-binary (RS codes [3]) block turbo codes (BTCs) with a soft-input soft-output (SISO) iterative decoding algorithm [4] which offers a good trade-off between performance and complexity. Since BTC can achieve high data rates with a low decoding complexity, it has found widespread applications (LMDS, UMTS, IEEE 802.16) and has become a key ingredient in the telecommunication system design process.

Soft iterative decoding of BTC requires the generation of a large set of codewords to compute reliable soft output for each bit. Various approaches have been developed in the literature in order to create a codeword list for a linear binary block code, denoted $C(n, k)$, where n is the length of the code and k its dimension. However, we can distinguish them into three major classes.

The first class of algorithm uses encoding to decode the received sequence. In the OSD[5]-BMA[6] or in [7] or [8], the principle is to reorder the soft values of the received sequence according to their reliability. Error patterns of increasing Hamming weight are applied at the k most reliable independent positions (MRIPs) and encoding is performed on the thresholded resulting sequences. All produced codewords are stored in the list of candidates. However, a new code generator must be evaluated for the encoding process due to the reordering, which increases the decoder complexity.

The second class of algorithm uses an algebraic decoder to construct a list of candidates. In the GMD [9], the Chase [10] or in [11], the principle is to locate the least reliable independent positions (LRIPs) in the received sequence. After thresholding, error patterns are applied at these LRIPs and algebraic Peterson [12] or Berlekamp-Massey (BM) [13] algorithm is performed. If the result is a codeword, then it is stored in the list of candidates.

The third class of algorithm uses both encoding and decoding to decode the received sequence as in [14].

The techniques described above process at the binary level. Generally, these algorithms are also applied for non binary block codes, but to their binary representation, *e.g* the Chase-2 algorithm for RS codes in [15]. This approximation leads to suboptimal performance.

The conventional SISO iterative decoding algorithm using the Chase algorithm is considered very efficient for BTCs with high code rates such as Hamming-BTCs (Hamming codes as component codes). However, for an high error correction capability (typically $t \geq 2$) its performance is far from the capacity limit and new algorithms have to be found.

In this paper, we propose to use a sliding encoding-window (SEW) algorithm [16] to decode binary or non-binary BTC with systematic cyclic codes as its component codes (BCH or RS codes). For this type of codes, the SEW algorithm can easily generate a list of codewords, which are close to the received sequence in terms of the Hamming distance. By combining this list of codewords with the one generated by a classical algebraic-based decoder, such as the Chase algorithm, we can form an enlarged set of test patterns. Based on the new test patterns, the SISO iterative decoding algorithm can

produce a better log-likelihood ratio (LLR) of each decoded digit. This eventually improves the bit error probability in the iterative decoding of the BTC.

This paper is organized as follows. Part II briefly introduces the transmission scheme. Part III describes the SEW algorithm and its combination with the Chase or the BM algebraic decoding. Part IV presents simulation results with non-binary RS-based BTCs on Gaussian channel, while Part V deals with binary BCH-based BTCs. Part VI concludes the paper.

II. Transmission Scheme

Information symbols are coded with a BTC scheme at the transmitter side before modulation. A BTC can be constructed by concatenating two or more short block codes with a block interleaver to become a long powerful code with a large minimum Hamming distance. Let us consider two linear block codes, $C_1(n_1, k_1, d_1)$ and $C_2(n_2, k_2, d_2)$ respectively as its component codes, where n_i is the codeword length, k_i is the information length and d_i is the minimum Hamming distance of the code, $i = 1, 2$. The information symbols are put in a $k_1 \times k_2$ symbol matrix. The k_2 columns are encoded with the code $C_1(n_1, k_1, d_1)$, then the n_1 obtained rows are encoded with the code $C_2(n_2, k_2, d_2)$.

The parameters of the BTC $C = C_1 \otimes C_2$ are the products of the component code parameters such as the code length $n = n_1 \times n_2$, the information length $k = k_1 \times k_2$ and the Hamming distance $d = d_1 \times d_2$. Thus, a BTC has a larger minimum Hamming distance. All of the rows of the matrix $\mathbf{c}$ are codewords of C_2 and all of its columns are codewords of C_1 [17]. This is a very important property of BTCs since in the iterative decoding process, both information bits and parity check bits can exploit the extrinsic information from the received sequences.

Symbols can be either q-ary symbols (RS case) or bits (BCH case). For non binary codes, each q-ary symbol can be represented by an unique set of m bits, with $q = 2^m$.

Let $\mathbf{c}$ be the binary representation of one binary or non-binary codeword of C and $\tilde{\mathbf{c}} = 2\mathbf{c} - 1$ be the modulated representation of $\mathbf{c}$ using a BPSK. BPSK symbols are sent through a Gaussian channel and the received vector $\mathbf{r}$ is given by:

$$\mathbf{r} = \tilde{\mathbf{c}} + \mathbf{n} \tag{1}$$

where $\mathbf{n}$ is the additive white Gaussian noise (AWGN). The optimum decoded sequence can be obtained by using the maximum-likelihood (ML) decoding algorithm. It chooses a codeword $\hat{\mathbf{c}}$ from the code set $\mathcal{C}$, which has a smallest Euclidean distance from the received vector, $\mathbf{r}$:

$$\hat{\mathbf{c}} = \text{argmin}_{\mathbf{c}\in\mathcal{C}} \|\mathbf{r} - \tilde{\mathbf{c}}\|^2 = \text{argmin}_{\mathbf{c}\in\mathcal{C}} \|\mathbf{r} - 2\mathbf{c} + 1\|^2 \tag{2}$$

Unfortunately, it is infeasible to search all the codewords in the ML decoding due to an extremely large number of codewords up to 2^{mk} for a large value of k.

III. Decoding Algorithm

A. Chase Algorithm

The Chase algorithm [10] is a sub-optimum decoding algorithm, which belongs to the argument list decoding algorithm. Instead of reviewing all the codewords of a code, the Chase algorithm searches the codewords within a sphere of radius $d - 1$ about the received word $\mathbf{r}$ due to the fact that the ML codeword $\hat{\mathbf{c}}$ is located in the sphere of radius $d - 1$ with a very high probability at a high SNR. Thus, all possible error patterns of binary weight less than or equal to $d - 1$ are considered.

However, to decrease the complexity, we use the second form of the Chase algorithm applied at the binary level. Let $\mathbf{y}$ be the hard decision of $\mathbf{r}$. We search the n_d LRIPs of $\mathbf{r}$ (lowest absolute values) and then make all possible commutations (0 or 1) at these n_d positions in $\mathbf{y}$. The 2^{n_d} produced sequences are decoded using a classical algebraic decoder. Only valid codewords are added to the code subset $\mathcal{C}(n_d)$.

The decoded codeword $\hat{\mathbf{c}}$ is chosen according to (2), where we use the subset $\mathcal{C}(n_d)$ instead of the whole codeword set $\mathcal{C}$.

For non-binary codes, we use the same approach as in [15] where the algorithm is still applied at the binary level on the binary representation of a codeword.

In order to apply the Chase algorithm to the SISO iterative decoding, it is desirable that the code set $\mathcal{C}(n_d)$ should contain the decoded codeword $\hat{\mathbf{c}}$ and its concurrent (competitor) codewords for all the binary positions. However, for non-binary codes and high error correction capability binary codes (typically $t \geq 2$), it appears that the code subset generated by the Chase algorithm is too small to find the concurrent codewords for all bits. In this case, the SISO iterative decoder has to use a coefficient to approximate the soft decoder output. This will lead to a performance degradation. To deal with this problem, we consider a sliding encoding-window (SEW) algorithm [16] for decoding the systematic cyclic codes. In particular, the SEW algorithm can generate a large set of codewords used by the decoder to find the concurrent codewords in order to compute more reliable soft-decision outputs. This can eventually improve the iterative decoding performance.

B. Sliding encoding-window algorithm

We recall that in systematic codes the information part is visible in the codeword, generally at the beginning or the end of the codeword. With a cyclic q-ary code, any cyclic permutation over q-ary symbols of a codeword is still a codeword.

The SEW algorithm exploits both the systematic and cyclic properties of a code, but processes at the binary level, to easily generate candidates, which are in the neighbourhood of the received sequence. The SEW algorithm consists of two successive steps: the encoding phase (EP) and the sliding phase (SP) repeated until all bits are covered with codeword diversity. We describe the algorithm for q-ary RS codes, but its extension to binary codes ($m = 1$) is straightforward.

Let $\mathbf{W}$ be the block of the km soft values extracted from $\mathbf{r}$ corresponding to the information part, $n_e \leq km$ be the number of the LRIPs inside $\mathbf{W}$ and $\overline{\mathbf{W}}$ be the hard decision of $\mathbf{W}$. By permuting the bits (0 or 1) at the n_e LRIPs of $\overline{\mathbf{W}}$ and encoding the result, we produce 2^{n_e} codewords "close" to $\mathbf{y}$: at least $km - n_e$ bits are identical. Codewords generated during this encoding phase (EP) are stored in the set $\mathcal{C}_{n_e}$.

As we use systematic codes, we introduce codeword diversity only over the $(n-k)m$ bits of the redundancy part. To create diversity over all bits, we exploit the q-ary cyclic property of the RS code. We cyclically shift $\mathbf{W}$ of Δ q-ary symbols (km bits) over $\mathbf{r}$. After this sliding phase (SP), we perform the EP on the new window. We call $\mathbf{W}$ the sliding encoding-window and Δ the SEW step.

Due to the systematic coding, the obtained codewords must be cyclically shifted in the reverse order so that their km-length information part matches the position of $\mathbf{W}$ in $\mathbf{r}$. We store shifted codewords in $\mathcal{C}_{n_e}$.

We repeat the EP and SP until all bits are covered with diversity. Obviously, the minimum step size is $\Delta = 1$, which means that every time, we slide the encoding window by one symbol position for a total of n EPs to process. Considering that the main objective of the SEW algorithm is to generate the codeword diversity for all n coded symbols and that each EP can potentially provide a codeword diversity over $n-k$ symbols, it is natural to set the step size of the sliding window operation $\Delta = n - k$. This is the step size which can be used to generate the codeword diversity for all coded symbols continuously with least overlapping. In this case, we need $\lceil \frac{n}{n-k} \rceil$ EPs to perform, where $\lceil . \rceil$ is the upper integer part. We can show that $2^{n_e} \leq \mathrm{card}(\mathcal{C}(n_e)) \leq \lceil \frac{n}{n-k} \rceil 2^{n_e}$.

It is interesting to note that a small step size Δ can provide better decoding performance but it suffers from a higher computational complexity. Therefore, it is possible to choose Δ between $\left[1, \lceil \frac{n}{n-k} \rceil\right]$ to achieve a good trade-off between the complexity and performance. Unless we precise a value of Δ, we will use $\Delta = n - k$ in the following.

The decoded codeword $\hat{\mathbf{c}}$ is chosen according to (2), where we use the subset $\mathcal{C}(n_e)$ instead of the entire code set $\mathcal{C}$.

The SEW algorithm is different from conventional encoder-based decoding algorithms in the way that no permutation or reordering of the received bits according to the reliability is required. Therefore, there is no need to recompute either the generator or parity check matrix, which is carried out by the Gaussian elimination method by checking the linear independence of the most reliable set of bits. Note that the complexity of encoding is much lower than that of the Gaussian elimination operation, one gets the performance enhancement at a slightly increased complexity.

C. Combination with decoding-based algorithm

The SEW algorithm always generates a codeword, but does not guarantee finding the most probable. In fact, if the thresholded input contains less than t errors, any algebraic decoding will produce the most probable codeword.

Therefore, we combine the SEW algorithm with at least one algebraic decoding, to ensure that if the input is successfully algebraically decoded, the proposed decoder produces the most probable codeword. In the following, we consider the Chase and the BM algorithms.

Depending on the algebraic decoding-based algorithm chosen, we combine the code set generated by the SEW $\mathcal{C}(n_e)$ with either the code set of the Chase $\mathcal{C}(n_d)$ or the decoded codeword if the BM succeeds. We denote $\mathcal{C}(n_e)^+$ this enlarged codeword set.

D. SISO decoder with combined decoding algorithm

The superior performance of BTC is due to the SISO iterative decoder. Decoding is performed either column-wise first and then row-wise or vice versa. For each bit of the selected codeword $\hat{\mathbf{c}}$, its LLR is computed according to:

$$\Gamma(\hat{c}_i) = \ln \frac{\Pr(\hat{c}_i = 1|\mathbf{r})}{\Pr(\hat{c}_i = 0|\mathbf{r})} \tag{3}$$

We use the subset $\mathcal{C}(n_e)^+$ instead of the entire set $\mathcal{C}$ to compute (3). Following the same approach as in [18], the LLR can be approximated by:

$$\Gamma(\hat{c}_i) \approx < \mathbf{c}^{i,1} - \mathbf{c}^{i,0}, \mathbf{r} > \tag{4}$$

where $< ., . >$ is the scalar product and:

$$\mathbf{c}^{i,j} = \mathrm{argmax}_{\mathbf{c} \in \mathcal{C}(n_e)^+ | c_i = j} < 2\mathbf{c} - 1, \mathbf{r} > \tag{5}$$

From (4) we can estimate the extrinsic information w_i of the i^{th} bit:

$$\begin{aligned} \Gamma(\hat{c}_i) &\approx < \mathbf{c}^{i,1} - \mathbf{c}^{i,0}, \mathbf{r} > \\ &\approx r_i + \sum_{j \neq i} (c_j^{i,1} - c_j^{i,0}) r_j \\ &\approx r_i + w_i \end{aligned} \tag{6}$$

The extrinsic information from the horizontal (vertical) codes can be used to update the received vector $\mathbf{r}$ for decoding the vertical (horizontal) codes.

$$\mathbf{r}(m) = \mathbf{r} + \alpha(m)\mathbf{w}(m) \tag{7}$$

where m is the index of the iteration, $\mathbf{w}(m)$ is the extrinsic information computed from the previous decoding step and $\alpha(m)$ is a weighting factor [4]. The decoding procedure described above is then generalized by cascading the elementary decoders as illustrated in Fig 1.

A parameter β is used for the computation of w_i, when either $\mathbf{c}^{i,1}$ or $\mathbf{c}^{i,0}$ is not available in $\mathcal{C}(n_e)^+$ for the i^{th} position [4].

IV. Non-binary BTCs

A. Codeword diversity

The SEW algorithm naturally creates codeword diversity. For SEW ($n_e = 8$) and the code $RS(15, 9, 7)$, Fig. 2 represents an average histogram of all codewords of $\mathcal{C}_{n_e}$ sorted by their Hamming distance from the transmitted codeword. Distances are computed at the binary level over 1000 decodings.

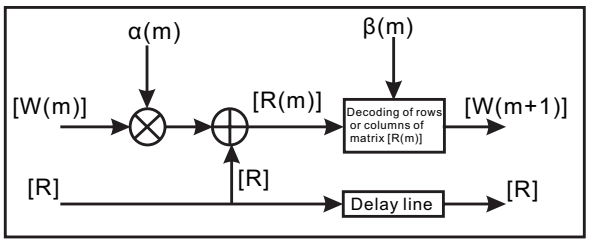

Fig. 1. SISO Decoder

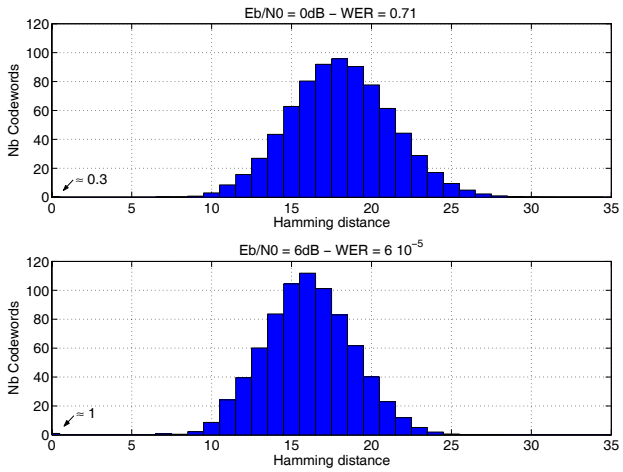

Fig. 2. SEW($n_e = 8$) codeword diversity at the binary level for $RS(15, 9, 7)$

As the signal to noise ratio increases, the SEW algorithm generates more codewords in the neighbourhood of the sent codeword. For the code $RS(15, 9, 7)$, the SEW ($n_e = 8$) performs 768 encoding operations, which can generate 767 and 766 different codewords at $E_b/N_0 = 0$ dB and 6 dB, respectively. However, in the radius of Hamming distance of 15 from the transmitted codeword, there are more codewords (325) at $E_b/N_0 = 6$ dB than that (163 codewords) at $E_b/N_0 = 0$ dB. The enlarged set of the codewords in the close neighbourhood of the correct codeword can help to improve the log-likelihood ratio of the decoder output.

This figure shows that the SEW algorithm provides sufficient codeword diversity to be used in a SISO context in order to compute soft output reliability. Due to this codeword diversity, the choice of the β parameter, used when no concurrent codewords are found for a given position [4], is less determining.

B. Simulation results

We use 4 iterations in the simulations and the parameters of the BTC [4] defined for each half-iteration are:

- $\alpha = \{0.0, 0.25, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5\}$
- $\beta = \{0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0\}$

For the BTC $RS(15, 9, 7)^2$ the decoding algorithms are : Chase($n_d = 4$), BM + SEW($n_e = 8$) and BM + SEW ($n_e = 12$). Fig. 3 presents the bit error rate (BER) on a Gaussian channel.

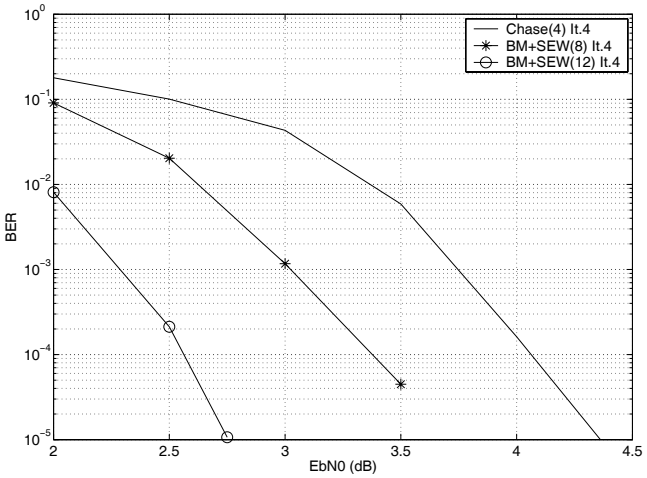

Fig. 3. BER vs Eb/N0 - BTC-$RS(15, 9, 7)^2$ on a Gaussian channel - 4 iterations

We observe a gain of 0.75 dB and 1.5 dB at a BER of 10^{-4} with the addition of the SEW ($n_e = 8$) and the SEW ($n_e = 12$) algorithm respectively.

For the BTC $RS(31, 27, 5)^2$, Fig. 4 shows the BER and the word error rate (WER) performance of the code with various decoding algorithms on a Gaussian channel: Chase($n_d = 5$), BM + SEW($n_e = 8$), Chase($n_d = 5$) + SEW($n_e = 8$), BM + SEW($n_e = 8$, $\Delta = 1$) and BM + SEW ($n_e = 12$).

We can observe that while the BM + SEW ($n_e = 8$) achieves the same BER performance as the Chase ($n_d = 8$) algorithm, the BM + SEW ($n_e = 8$) performs 0.15 dB better than the Chase ($n_d = 5$) algorithm in WER. In addition, we see that the BM + SEW ($n_e = 8$, $\Delta = 1$) and Chase ($n_d = 5$) + SEW ($n_d = 8$) have the same performance and outperform the Chase ($n_d = 5$) algorithm by 0.2 and 0.25 dB in BER and WER, respectively. Finally, we note that the BM + SEW ($n_e = 12$) is 0.5 and 0.6 dB better in BER and WER respectively than the classical Chase ($n_d = 5$). However, its complexity is greater than all the previous algorithms, but the SEW process can be parallelized as the 8 EPs performed for the $RS(31, 27, 5)$ are independent from each other.

V. Binary BTCs

The BTC $BCH(31, 16, 7)^2$ has been investigated over an AWGN channel with BPSK modulation by using Monte-Carlo simulation. The number of LRIPs of Chase algorithm is set as $n_d = 5$. The number of LRIPs of the SEW algorithm is set as $n_e = 6$ to maintain a reasonable complexity. The number of the iterations of the SISO iterative decoder is set to 4. The BER performance comparison of the combined and the conventional algorithms is shown in Fig. 5.

It is obvious that the considered algorithm outperforms the conventional Chase-Pyndiah decoding algorithm for the BTC. In Fig 5, we also show the effect of step size Δ of the sliding operation on the code error performance. We can see that reducing the step size Δ from 15 to 8 can slightly improve the

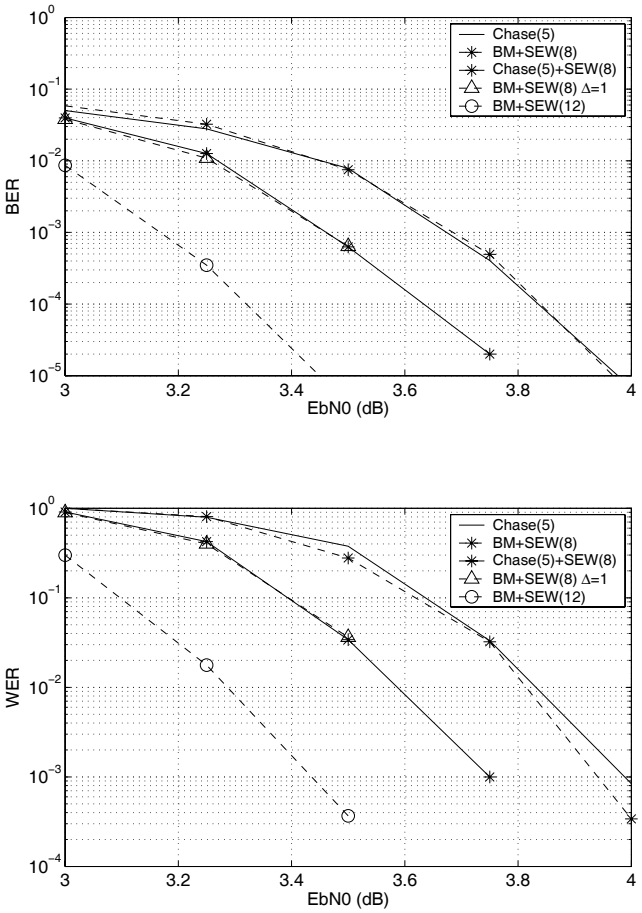

Fig. 4. BER vs Eb/N0 and WER vs Eb/N0 - BTC-$RS(31, 27, 5)^2$ on a Gaussian channel - 4 iterations

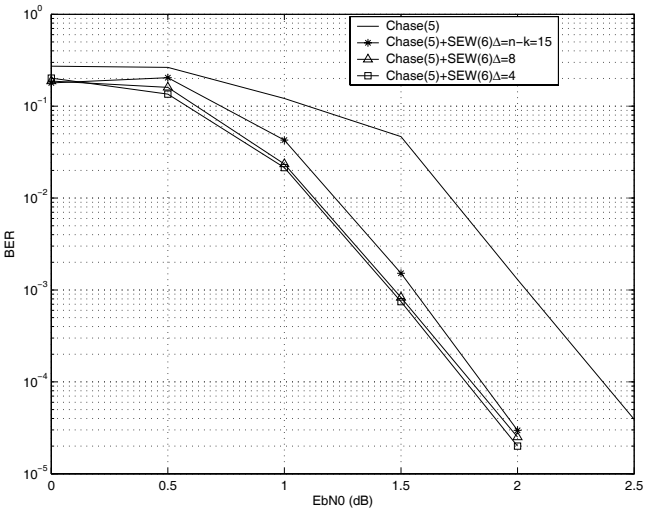

Fig. 5. BER vs Eb/N0 - BTC-$BCH(31, 16, 7)^2$ on a Gaussian channel - 4 iterations

bit error rate. However, the performance improvement obtained by reducing the step size Δ from 8 to 4 is negligible.

VI. Conclusion

We considered an improved iterative decoding algorithm for turbo product codes. The algorithm exploits the code properties of the cyclic codes to generate an enlarged set of test patterns close to the received word, which is combined with the result of an algebraic decoding based algorithm, such as Chase or BM. The addition of the SEW algorithm can improve the reliability of extrinsic information hence to improve the bit and the word error performance of BTCs.

We show that the proposed algorithm can improve the BER of BTCs up to 1.5 dB. When combined with the Chase algorithm, the SEW algorithm produces an enlarged codeword set at the expense of a slightly increased complexity. Compared to the other encoder-based decoding algorithms, the proposed algorithm has the advantage that there is no requirement to recompute the generator or parity-check matrix by using Gaussian elimination operations.

References

[1] P. Elias. Error-free coding. *IRE Trans. on Inform. Theory*, 4(4):29–37, September 1954.

[2] R.C. Bose and D.K. RayChaudhury. On a class of error correcting binary group fields. *Inform. Contr.*, 3:68–79, March 1960.

[3] I.S. Reed and G. Solomon. Polynomial codes over certain finite fields. *J. Soc. Ind. Appl. Math*, 8:300–304, June 1960.

[4] R. Pyndiah, A. Glavieux, A. Picart, and S. Jacq. Near optimum decoding of product codes. *Global Telecommunications Conference*, 1:339–343, November 1994.

[5] M.P.C. Fossorier and Shu Lin. Soft-decision decoding of linear block codes based on ordered statistics. *IEEE Trans. on Inform. Theory*, 41(5):1379–1396, September 1995.

[6] P.A. Martin, A. Valembois, M.P.C. Fossorier, and D.P. Taylor. On soft-input soft-output decoding using "box and match" techniques. *IEEE Trans. on Communications*, 52(12):2033–2037, December 2004.

[7] D. Gazelle and J. Snyders. Reliability-based code-search algorithms for maximum-likelihood decoding of block codes. *IEEE Trans. on Inform. Theory*, 43(1):239–249, January 1997.

[8] P.A. Martin, D.P. Taylor, and M.P.C. Fossorier. Soft-input soft-output list-based decoding algorithm. *IEEE Trans. on Communications*, 52(2):252–262, February 2004.

[9] G.Jr. Forney. Generalized minimum distance decoding. *IEEE Trans. on Inform. Theory*, 12(2):125–131, April 1966.

[10] D. Chase. A class of algorithms for decoding block codes with channel measurement information. *IEEE Trans. Inform. Theory*, 18(1):170–182, January 1972.

[11] T. Kaneko, T. Nishijima, H. Inazumi, and S. Hirasawa. An efficient maximum-likelihood-decoding algorithm for linear block codes with algebraic decoder. *IEEE Trans. on Inform. Theory*, 40(2):320 – 327, March 1994.

[12] W.W. Peterson and E.J. Weldon Jr. *Error-correcting codes*. MIT Press, second edition, 1972.

[13] E.R. Berlekamp. *Algebraic Coding Theory*. Mc Graw Hill, 1968.

[14] T.H. Hu and S. Lin. An efficient hybrid decoding algorithm for reed-solomon codes based on bit reliability. *IEEE Trans. on Communications*, 51(7):1073–1081, July 2003.

[15] O. Aitsab and R. Pyndiah. Performance of reed-solomon block turbo code. *Global Telecommunications Conference*, 1:121–125, November 1996.

[16] M. Lalam, K. Amis, and D. Leroux. On the use of reed-solomon codes in space-time coding. *IEEE International Symposium on Personal Indoor and Mobile Radio Communications PIMRC'05*, September 2005.

[17] F.J. Macwilliams and N.J.A. Sloane. *The Theory of Error Correcting Codes*. North-Holland Publishing Company, 1978.

[18] M. Lalam, K. Amis, and D. Leroux. Sliding encoding-window for reed-solomon code decoding. *International Symposium on Turbo Codes ISTC'06*, May 2006.