Turbo decoding of product codes using adaptive belief propagation

Christophe Jego, Warren J. Gross

Turbo Decoding of Product Codes using Adaptive Belief Propagation

Christophe Jégo[†], *member IEEE* and Warren J. Gross[‡], *member IEEE*

***Abstract*—The Adaptive Belief Propagation (ABP) algorithm was recently proposed by Jiang and Narayanan for the soft decoding of Reed-Solomon (RS) codes. In this paper, simplified versions of this algorithm are investigated for the turbo decoding of product codes. The complexity of the Turbo-oriented Adaptive Belief propagation (TAB) algorithm is significantly reduced by moving the matrix adaptation step outside of the belief propagation iteration loop. A reduced-complexity version of the TAB algorithm that offers a trade-off between performance and complexity is also proposed. Simulation results for the turbo decoding of product codes show that belief propagation based on adaptive parity check matrices is a practical alternative to the currently very popular Chase-Pyndiah algorithm.**

I. Introduction

Belief Propagation (BP) decoding [1] is a SISO decoding algorithm for linear block codes that is based on the propagation of soft information along the edges of a graph defined by the parity check matrix associated with the code. The BP algorithm is considered to be the reference LDPC decoding algorithm and it exhibits a high degree of parallelism making it suitable for high data rate applications. It is commonly believed that the BP algorithm is not suitable for decoding codes with non-sparse parity check matrices such as BCH and RS codes. Recently, however, the Adaptive Belief Propagation (ABP) algorithm was proposed for the decoding of RS codes with high-density parity check matrices [2]. This method adapts the parity check matrix at each iteration of the BP algorithm according to the bit reliabilities in order to sparsify the columns of the parity check matrix associated with the unreliable bits.

The motivation of this work is to find a BP-based algorithm to be applied to linear block codes with a non-sparse matrix for use in turbo decoders of product codes. The ABP algorithm seems especially interesting for decoding product codes whose parity check matrix is not sparse. Indeed, the graph-based message passing step of the ABP algorithm is highly parallelizable, unlike the Chase-Pyndiah algorithm [3]. However, due to the adaptation step that performs the sparsification at every iteration, the complexity of ABP becomes prohibitive for hardware implementation. A possible solution is to run a small number of iterations of the BP algorithm on the same parity check matrix as suggested in [4]. We thus investigate a simplified version of the ABP algorithm for turbo product codes using BCH or RS component codes called the Turbo-oriented Adaptive Belief propagation (TAB) algorithm in Section II. In order to decrease the complexity of the TAB algorithm, a complexity reducing method for the BP process is proposed in Section III. This version, called TAB Simplified (or TABS), offers a good trade-off between performance and complexity for the iterative decoding of product codes. Some performance results for different product codes using BCH or RS component codes are given in Section IV. The results show that belief propagation based on adaptive parity check matrices can provide performance similar to that of Chase-Pyndiah algorithm.

[†]Institut Telecom/Telecom Bretagne, CNRS Lab-STICC UMR 3192, Electronic Engineering Department, 29238 Brest Cedex 3, Université Européenne de Bretagne, France
[‡]Department of Electrical and Computer Engineering, McGill University, Montreal, H3A 2A7 Canada
E-mail: [†]christophe.jego@enst-bretagne.fr, [‡]wjgross@ece.mcgill.ca

II. The TAB algorithm applied to iterative decoding of product codes

Turbo decoding of product code involves sequentially decoding rows and columns using a SISO decoding algorithm. The turbo decoding process repeats this soft decoding for several iterations. Each decoding process computes soft information $\mathbf{y}'_{(\mathbf{it+1})}$ from the channel received information $\mathbf{y}$ and the information computed in the previous half-iteration, $\mathbf{y}_{(\mathbf{it})}$. The extrinsic information $\mathbf{w}_{(\mathbf{it})}$ is obtained by subtracting the soft input information $\mathbf{y}_{(\mathbf{it+1})}$ from the soft output information $\mathbf{f}_{(\mathbf{it})}$. The soft information $\mathbf{y}'_{(\mathbf{it+1})}$ is given by $\mathbf{y}'_{(\mathbf{it+1})} = \mathbf{y} + \alpha_{(\mathbf{it})}\mathbf{w}_{(\mathbf{it})}$ where $\alpha_{(it)}$ is a scaling factor that reduces the effect of the extrinsic information in the soft decoder during the first decoding steps. Each component code (N_i, K_i) of the product code has a parity check matrix $\mathbf{H}$ with $N_i - K_i$ rows and N_i columns. In this section, we propose the Turbo-oriented Adaptive Belief propagation algorithm (TAB), a simplified version of the standard ABP algorithm [2] to be used as the SISO decoding algorithm during the turbo decoding of product codes. Each iteration of the ABP algorithm consists of two sub-steps: matrix sparsification and belief propagation. The TAB algorithm is motivated by the high cost of the Gaussian elimination required for adaptation in each iteration of the ABP algorithm. We propose eliminating the adaptation from the iteration loop leaving only one adaptation in the initialization phase. The modified algorithm is composed of four steps. First, the received vector $\mathbf{y}'_{(\mathbf{it})}$ is ordered according to the absolute value of the soft input symbols in an ascending order (from the least reliable value to the most reliable value). Then, the $N - K$ columns of the original parity check matrix $\mathbf{H}$ corresponding to the least reliable bits in $\mathbf{y}'_{(\mathbf{it})}$ are reduced to obtain an identity matrix by applying Gaussian elimination. The objective is to decrease the

number of ones in the part of the parity check matrix that are associated with the least reliable symbols. Adapting the parity check matrix makes it suitable for the standard BP algorithm. The standard BP algorithm is applied in Step 3 to generate the soft output information $\mathbf{f_{(it)}}$. We note that few iterations (3 to 5) are necessary during the BP process in the proposed algorithm. Running a very small number of iterations on the same parity check matrix is especially effective in our turbo process. Indeed, the belief propagation is not exact due to the short cycles in the associated Tanner Graph. We call these local iterations to distinguish them from the iterations of the turbo process which we call global iterations. As the matrix updating stage is not in the local iterative loop, a significant decrease of the SISO decoding algorithm complexity is obtained. In addition, no damping coefficient [2] is necessary for the TAB algorithm. Instead, the reduction of the extrinsic information effect is done during the soft information computation.

III. A VARIANT OF THE TAB ALGORITHM: THE TABS ALGORITHM

In order to decrease the complexity of the standard Belief Propagation algorithm, simplified versions have been proposed. The best-known is the BP-based algorithm proposed by Fossorier et al. [5] In the BP-based algorithm, the parity check node processing is replaced by a selection of the minimum value for the magnitude. The memory and the complexity reductions are significant but the degradation in terms of BER can also be significant. For this reason, we propose a more accurate simplification. The computation of parity check nodes depends mainly on the smallest values of the messages $\boldsymbol{V_{n,m}}$ from variable nodes. These messages depend on the log-likelihood ratio $\boldsymbol{L_n}$ of the received bit n. Consequently, the BP decoding iterative process depends mainly on the least reliable bits. On the other hand, the first step of the TAB algorithm provides an ordering of the soft input symbols. For these reasons, we propose to apply the BP algorithm to a subset $\boldsymbol{S_n}$ of the soft input symbols of the codeword $\mathbf{y} = (y_1, y_2, ..., y_n)$ to decrease the memory and the computation complexity. $\boldsymbol{S_n}$ is obtained by taking into account the reliability of the symbols and the least reliable received bits are considered. Let $\boldsymbol{P(n)}$ and $\boldsymbol{Q(m)}$ denote the set of all the parity check nodes that are connected to the variable node n of the subset $\boldsymbol{S_n}$ and the set of all variable nodes of the subset $\boldsymbol{S_n}$ that are connected to the parity check node m of the subset $\boldsymbol{S_n}$ respectively. $\boldsymbol{P(n)/m}$ is the set of the parity check nodes that are connected to the variable node n without the parity check m and $\boldsymbol{Q(m)/n}$ is the set of variable nodes that are connected to the parity check node m without the variable n. The proposed simplified Belief Propagation (sBP) algorithm is carried out as follows:

- Define a subset of soft symbols in function of the reliability: $\boldsymbol{S_n}$
- Message $\boldsymbol{C_{m,n}}$ computation:

$$C_{m,n} = \prod_{n' \in N_{(m)/n}} sign\left(V_{n',m}\right) * 2tanh^{-1} \prod_{n' \in P(m)/n} tanh \left| \frac{V_{n',m}}{2} \right|$$

- Message $\boldsymbol{V_{n,m}}$ computation:

$$V_{n,m} = \sum_{m' \in Q(n)/m} C_{m',n} + L_n$$

- Extrinsic information $\boldsymbol{V_n}$ computation:

$$V_n = \sum_{m \in Q(n)} C_{m,n} + L_n$$

During the iterative process, the messages $\boldsymbol{C_{m,n}}$ are computed for the set $\boldsymbol{Q(m)}$. The check node update rule can be separated into the sign and the magnitude processing. The magnitudes of the messages $\boldsymbol{V_{n,m}}$ are computed for the set $\boldsymbol{P(n)}$ and the signs of the messages $\boldsymbol{V_{n,m}}$ are updated for the initial set $\boldsymbol{N(m)}$. A similar strategy was previously presented in [6]. In this paper, the authors investigated the performance of modified versions of adaptive BP algorithms for iterative soft-decision decoding of RS codes over magnetic recording channels. The reduced-complexity version of the standard ABP algorithm, called MABP, is based on the fact that unreliable bits are more likely to be erroneous. In the MABP algorithm, the received bit sequence is divided into two groups according to the absolute value of the soft input symbols: unreliable bits and reliable bits. The columns of the original parity check matrix $\mathbf{H}$ corresponding to the unreliable bits are reduced to obtain an identity matrix by applying Gaussian elimination. Then, the LLRs for the group of unreliable bits are updated by using the BP process and the LLRs of reliable bits are kept unchanged. The numbers of elements in the two groups are $N - K$ unreliable elements and K reliable elements respectively [7]. This means that the group sizes depend on the parameters of the component codes. In contrast, our method enables us to choose the size of the subset $\boldsymbol{S_n}$ of the soft input symbols. In practice, it is necessary to choose a number between $N/2$ and N to obtain an efficient ratio between performance and complexity. Moreover, as LLRs of reliable bits are kept unchanged in the MABP algorithm, this method can not be applied if the parity check matrix adaptation is done before the BP process like in the TAB algorithm. For theses reasons, our method that consists of applying the BP algorithm to a subset $\boldsymbol{S_n}$ of the soft input symbols is more suitable to obtain an efficient trade-off between performance and complexity for the turbo decoding of product codes. We call this variant the TABS (Turbo-oriented Adaptive Belief propagation Simplified) algorithm.

Table I gives complexity comparison in terms of operations between the proposed sBP algorithm and the classical BP algorithm. The memory requirements are also given. The complexity of the BP algorithm depends on the code length N, the number of information bits K, the average degree of the variable nodes Dv, the average degree of the check nodes Dc and the number of local iterations $Iter$. The complexity of the sBP algorithm depends on the same parameters and on another one S that defines the number of unreliable bits of the subset $\boldsymbol{S_n}$. First, multiplications are necessary for the initialization of the LLR $\boldsymbol{L_n}$. Computation of extrinsic information $\boldsymbol{V_n}$ is done through additions and subtractions. During the check node update, the processing of the sign and magnitude of the check to variable messages $\boldsymbol{C_{m,n}}$ is done separately. The computation of the signs of the $\boldsymbol{C_{m,n}}$ messages is processed through an XOR function. In contrast,

the magnitude processing is the bottleneck in the computation of $C_{m,n}$ messages. Performing in the logarithmic domain simplifies this computation greatly, since the multiplications become additions. In the logarithmic domain, the parity check node processing is simplified as follow:

$$C_{m,n} = \prod_{n' \in N_{(m)/n}} sign(V_{n',m}) * f^{-1}\left(\sum_{n' \in P_{(m)/n}} f(V_{n',m}) \right)$$

where f(x) = $-ln\left[tanh\left(\frac{|x|}{2}\right)\right]$

The magnitude processing is computed by lookup tables (LUT) for the $f(x)$ function and additions. Received symbols, $V_{n,m}$ messages and $C_{m,n}$ messages have to be stored during the BP iterative process. Table I shows that the sBP algorithm is less complex than the standard BP algorithm. A decrease by a factor $(1 - S/N)$ is obtained for the computation of the extrinsic information V_n and the magnitude update of the check to variable messages $C_{m,n}$. Moreover, the numbers of $V_{n,m}$ messages and $C_{m,n}$ messages to store decreases by the same factor $(1-S/N)$. In summary, the sBP algorithm enables significant reductions of memory requirements and complexity in terms of operations.

Operations	BP algorithm	simplified BP (sBP) algorithm
Extrinsic information additions/subtractions	**Iter*(N*2Dv)**	**Iter*(S*2Dv)**
Check node additions	**Iter*((N-K)*(2Dc-1))**	**Iter*((N-K)*{2(Dc*(S/N))-1)}**
Exclusive OR	**Iter*((N-K)*(2Dc-1))**	**Iter*((N-K)*(2Dc-1))**
LUT for f(x)	**Iter*((N-K)*2Dc)**	**Iter*((N-K)*{2(Dc*(S/N))}**
Multiplication	**N**	**N**

Memories	BP algorithm	simplified BP (sBP) algorithm
received symbols	**N words**	**N words**
variable->check messages	**N*Dv words**	**S*Dv words**
check->variable message	**(N-K)*Dc words**	**(N-K)* (Dc*(S/N)) words**

TABLE I
COMPARISON IN TERMS OF OPERATION AND MEMORY COMPLEXITIES BETWEEN THE SBP AND BP ALGORITHMS.

IV. ITERATIVE DECODING OF PRODUCT CODES

In this section, the Bit Error Rate (BER) vs. Signal to Noise Ratio (SNR) performance of iterative decoding of product codes is presented. Concerning BCH codes, extended codes are considered because they are more efficient than non-extended codes. In the case of product codes using RS component codes, extended codes are not efficient because the impact on code rate and consequently on the Shannon limit is not significant as shown in [8]. Thus, turbo product codes using RS component codes are limited to non-extended codes in this paper.

A. Performance for several BCH product codes on a Gaussian channel

The performance of the TAB algorithm is compared with that of the Chase-Pyndiah algorithm for different product codes using BCH component codes. Three local iterations and 8 global iterations are chosen for the TAB algorithm. Eight global iterations and 16 error patterns are chosen for the Chase-Pyndiah algorithm. Bit error rate performance of turbo decoding of BCH product codes on Gaussian channel for two error correction powers (t=1 and t=2) are depicted in Fig. 1. For comparison, the uncoded BPSK is also plotted. The performance of the TABS algorithm for all the chosen BCH product codes is also given. For each case, a complexity decrease factor $(1 - S/N)$ is selected to offer a good trade-off between complexity and performance. No significant BER deviation is observed between the Chase-Pyndiah and the TAB algorithms. The Chase-Pyndiah algorithm outperforms the TAB algorithm by about 0.135 dB at BER of 10^{-6} for $(32, 26)^2$ BCH product codes. In contrast, the TAB algorithm outperforms Chase-Pyndiah algorithm by about 0.090 dB at 10^{-3} for a $(32, 21)^2$ BCH product code. These two cases are the greatest deviations observed in favor of each algorithm.

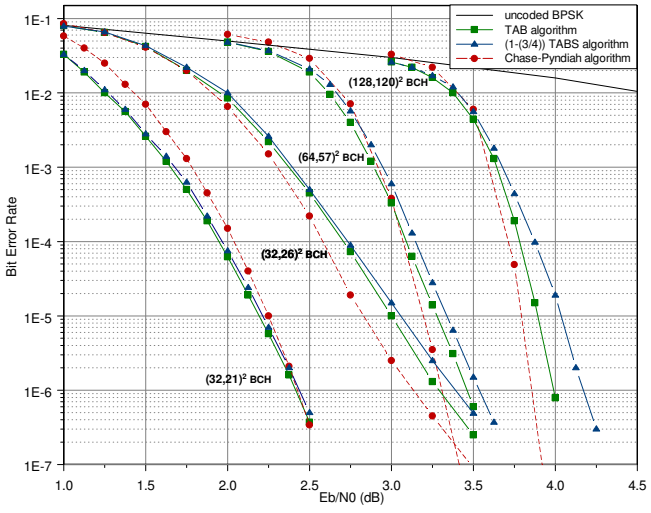

Fig. 1. Bit error rate performance of turbo decoding of BCH product codes (t=1 and t=2) on a Gaussian channel.

B. Iterative Decoding of Reed-Solomon Product Codes

The TAB and TABS algorithms are applied to the binary image expansion of the parity check matrix of the RS component codes of the product codes. It has been shown that RS codes can be decomposed into BCH subfield subcodes [9]. The performance of the TAB algorithm is compared with that of the Chase-Pyndiah algorithm for different product codes using RS component codes. The performance of the TABS algorithm is also given. The complexity factors $(1-(11/15))$, $(1-(25/31))$ and $(1-(57/63))$ were chosen for RS product codes with code lengths N=15, 31 and 63 respectively. Five local iterations and eight global iterations are chosen for the TAB algorithm. Three local iterations and eight global iterations are sufficient for the TABS algorithm. Eight global iterations and 16 error

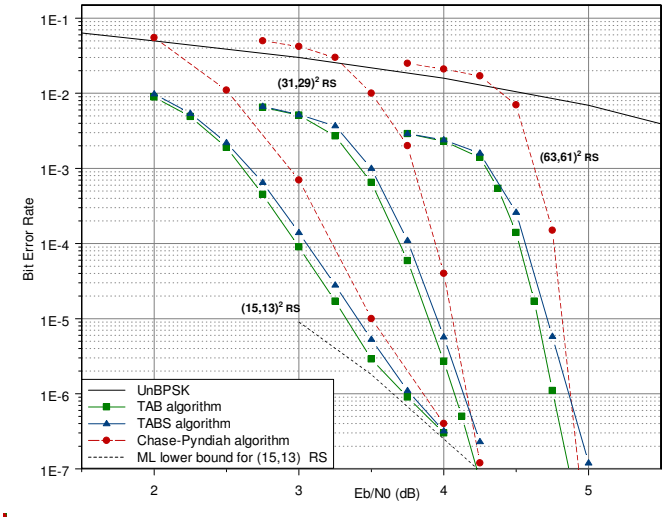

Fig. 2. Bit error rate performance of turbo decoding of Reed-Solomon product codes (t=1) on a Gaussian channel.

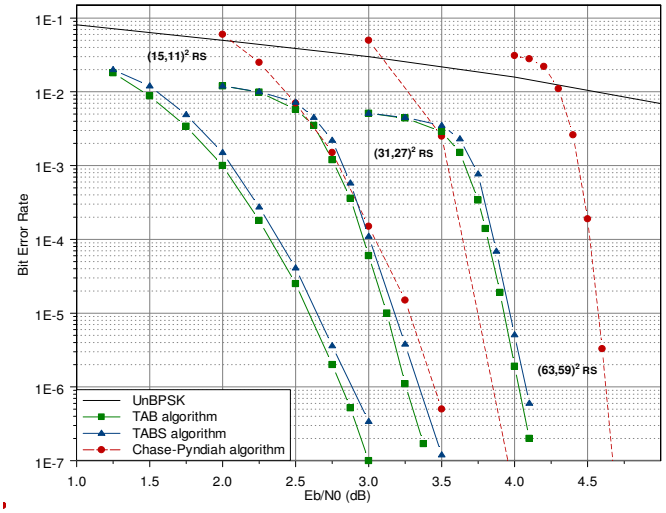

Fig. 3. Bit error rate performance of turbo decoding of Reed-Solomon product codes (t=2) on a Gaussian channel.

patterns are chosen for the Chase-Pyndiah algorithm. Bit error rate performance of turbo decoding of RS product codes on Gaussian channel for a correction power t=1 is depicted in Fig. 2. We note that turbo decoding based on the TAB algorithm outperforms turbo decoding based on the Chase-Pyndiah algorithm for the three considered RS product codes. In particular, the proposed algorithm enables a gain in terms of convergence performance at low SNR. It provides around 0.3 dB gain at BER= 10^{-3} for the three performances curves. For high SNR, performance results between the two algorithms are close. However the trade-off between performance and complexity is obtained with a slight degradation of BER performance compared with the TAB algorithm. On the other hand, a significant complexity gain and also a decrease from 5 to 3 of the local iterations are achieved. Figure 3 gives the bit error performance of turbo decoding of RS product codes on a Gaussian channel for correction power t=2. We observe that in terms of error performance the TAB algorithm clearly outperforms Chase-Pyndiah decoding solution for RS product codes with a correction power of t=2. The TAB algorithm enables a significant gain in terms of convergence performance at low SNR. For the TABS algorithm, complexity decrease factors $(1 - S/N)$ were chosen for each RS product codes to offer a good trade-off between complexity and performance. Fig. 3 shows the slight degradations of BER performance compared with the TAB algorithm.

V. Conclusion

In this paper, simplification versions of the ABP algorithm have been proposed for the turbo decoding of product codes using BCH or RS component codes. In order to further decrease the complexity of the TAB algorithm, a complexity reducing method for the BP process is introduced as the TABS algorithm. Simulation results have shown that the TAB algorithm is an alternative to the Chase-Pyndiah algorithm for product codes. Moreover, the proposed TABS algorithm offers a trade-off between performance and complexity. The major advantages of the two algorithms are their high degree of parallelism for high data rate applications and the possibility of applying to it a new decoding method called *stochastic decoding*. Stochastic decoding previously applied to the BP decoding of LDPC codes [10], has the potential to be applied to turbo decoding of product codes with the TAB algorithm.

References

[1] R. G. Gallager, "Low density parity check codes," *IRE Trans. Inform. Theory*, vol. IT, pp. 21–28, January 1962.

[2] J. Jiang and K. R. Narayanan, "Iterative soft-input-soft-output decoding of," *Information Theory, IEEE Transactions on*, vol. 52, pp. 3746–3756, 2006.

[3] R. Pyndiah, "Near-optimum decoding of product codes: block turbo codes," *Communications, IEEE Transactions on*, vol. 46, no. 8, pp. 1003–1010, Aug. 1998.

[4] M. El-Khamy and R. J. McEliece, "Iterative algebraic soft-decision list decoding of reed-solomon codes," *Selected Areas in Communications, IEEE Journal on*, vol. 24, pp. 481–490, March 2006.

[5] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low density parity check codes based on belief propagation," *Communications, IEEE Transactions on*, vol. 47, no. 5, pp. 673–680, May 1999.

[6] H. Xia and J. R. Cruz, "Performance of reliability-based iterative soft-decision reed-solomon decoding on magnetic recording channels," *Magnetics, IEEE Transactions on*, vol. 43, pp. 3320–3323, July 2007.

[7] ——, "Reliability-based reed-solomon decoding for magnetic recording channels," *Magnetics, IEEE Transactions on*, vol. 42, no. 10, pp. 2603–2605, October 2006.

[8] R. Zhou, R. Le Bidan, R. Pyndiah, and A. Goalic, "Low-complexity high-rate reed–solomon block turbo codes," *Communications, IEEE Transactions on*, vol. 55, no. 9, pp. 1656–1660, Sept. 2007.

[9] A. Vardy and Y. Be'Ery, "Bit-level soft-decision decoding of reed-solomon codes," *Communications, IEEE Transactions on*, vol. 39, pp. 440–444, March 1991.

[10] S. Tehrani, W. Gross, and S. Mannor, "Stochastic decoding of ldpc codes," *Communications, IEEE Letters*, vol. 10, pp. 716–718, October 2006.