International Journal of Engineering Research and Development
e-ISSN: 2278-067X, **p**-ISSN : 2278-800X, www.ijerd.com
Volume 5, Issue 2 (December 2012), PP. 06-13

A Novel Approach for FEC Decoding Based On the BP Algorithm in LTE and Wimax Systems

Kallu Sujatha[1], Subba Rao Katteda[2], S Amarnath Babu[3]
[1]PG Student, Department Of IT, St.Anns College of Engg. & Tech., Chirala , Prakasam Dist, AP, INDIA.
[2]Associate Professor, Department Of IT, St.Anns College of Engg. & Tech., Chirala , Prakasam Dist, AP, INDIA.
[3]Associate Professor & HOD, Department Of IT, St.Anns College of Engg. & Tech., Chirala , Prakasam Dist, AP, INDIA.

Abstract:- Many wireless communication systems such as IS54, enhanced data rates for the GSM evolution (EDGE), worldwide interoperability for microwave access (WiMAX) and long term evolution (LTE) have adopted low-density parity-check (LDPC), tail-biting convolutional, and turbo codes as the forward error correcting codes (FEC) scheme for data and overhead channels. Therefore, many efficient algorithms have been proposed for decoding these codes. However, the different decoding approaches for these two families of codes usually lead to different ardware architectures. Since these codes work side by side in these new wireless systems, it is a good idea to introduce a universal decoder to handle these two families of codes. The present work exploits the parity-check matrix (H) representation of tailbiting convolutional and turbo codes, thus enabling decoding via a unified belief propagation (BP) algorithm. Indeed, the BP algorithm provides a highly effective general methodology for devising low-complexity iterative decoding algorithms for all convolutional code classes as well as turbo codes. While a small performance loss is observed when decoding turbo codes with BP instead of MAP, this is offset by the lower complexity of the BP algorithm and the inherent advantage of a unified decoding architecture.

I. INTRODUCTION

Until recently, most known decoding algorithms for convolutional codes were based on either algebraically calculating the error pattern or on trellis graphical representations such as in the MAP and Viterbi algorithms. With the advent of turbo coding [1], a third decoding principle has appeared: iterative decoding. Iterative decoding was also introduced in Tanner's pioneering work [2], which is a general framework based on bipartite graphs for the description of LDPC codes and their decoding via the belief propagation (BP) algorithm.

In many respects, convolutional codes are similar to block codes. For example, if we truncate the trellis by which a convolutional code is represented, a block code whose codewords correspond to all trellis paths to the truncation depth is created. However, this truncation causes a problem in error performance, since the last bits lack error protection. The conventional solution to this problem is to encode a fixed number of message blocks L followed by m additional all-zero blocks, where m is the constraint length of the convolutional code [4]. This method provides uniform error protection for all information digits, but causes a rate reduction for the block code as compared to the convolutional code by the multiplicative factor $L/(L+m)$. In the tail-biting convolutional code, zero-tail bits are not needed and replaced by payload bits resulting in no rate loss due to the tails. Therefore, the spectral efficiency of the channel code is improved. Due to the advantages of the tail-biting method over the zero-tail, it has been adopted as the FEC in addition to the turbo code for data and overhead channels in many wireless communications systems such as IS-54, EDGE, WiMAX and LTE [5, 6, 7].

Both turbo and LDPC codes have been extensively studied for more than fifteen years. However, the formal relationship between these two classes of codes remained unclear until Mackay in [8] claimed that turbo codes are LDPC codes. Also, Wiberg in [9] marked another attempt to relate these two classes of codes together by developing a unified factor graph representation for these two families of codes. In [10], McEliece showed that their decoding algorithms fall into the same category as BP on the Bayesian belief network. Finally, Colavolpe [11] was able to demonstrate the use of the BP algorithm to decode convolutional and turbo codes. The operation in [11] is limited to specific classes of convolutional codes, such as convolutional self orthogonal codes (CSOCs). Also, the turbo codes therein are based on the serial structure while the parallel structure is more prevalent in practical applications.

In LTE and WiMAX systems, the proposed decoders for the tail-biting convolutional codes and turbo codes are based on the Viterbi and MAP algorithms, respectively. However, many other efficient algorithms have been proposed to decode tail-biting convolutional codes as well as turbo codes. For example, in [3], the reduced complexity wrap-around Viterbi algorithm was proposed to decode tail-biting convolutional codes in the WiMAX system to reduce the average number of decoding iterations and memory usage. In addition, other decoding algorithms such as double traceback and bidirectional Viterbi algorithms were also proposed for tail-biting convolutional codes in LTE [4]. Finally in [5], the design and optimization of low-complexity high performance rate-matching algorithms based on circular buffers for LTE turbo codes was investigated.

In this paper, we focus on the direct application of the BP algorithm used for LDPC codes to decode the tail-biting convolutional codes and turbo codes in WiMAX and LTE systems, respectively. Based on that, we propose a decoder with drastically lower implementation complexity than that proposed in the latest releases for these systems [5-7]. The rest of this paper is organized as follows. In Section II and III, the graphical representation of the tail-biting convolutional and turbo codes with the necessary notations and definitions used throughout this paper are introduced, followed by an investigation of the coding structures in WiMAX and LTE systems in Section IV. In Section V, simulation results for the performance of tail-biting convolutional and turbo codes using the proposed algorithm are introduced followed in Section VI by a complexity comparison between the proposed algorithm and the traditional ones. Finally, the paper is concluded in Section

II. CONVOLUTIONAL CODES

First introduced by Elias in 1955 [15], binary convolutional codes are one of the most popular forms of binary error correcting codes that have found numerous applications [16]. A convolutional code is called tail-biting when its codewords are those code sequences associated with paths in the trellis that start from a state equal to the last m bits of an information vector of k data bits. Many efficient algorithms have been proposed for decoding tail-biting convolutional codes such as the Viterbi and MAP algorithms. As shown below, we represent the tail-biting convolutional code by its generator and parity-check matrices in order to apply the BP algorithm directly.

A. Parity-check matrix of tail-biting convolutional codes

To be able to represent a tail-biting convolutional code by a Tanner graph and then apply the BP algorithm to its decoding, a prerequisite is to obtain its generator matrix **G** and its paritycheck matrix **H**. We introduce the matrix representation by a simplified example as follows:

Example 1: Consider the convolutional code with rate $R = {}^{n}\!/_{k} = 1/2$, where *k* represents the number of input bits and *n* the output bits. Assume that the information sequence is **x** = $(x_0, x_1, x_2, \dots)$**.** The encoder will convert this to the sequences $\boldsymbol{y}^{(\mathbf{0})} = (y_0^{(0)}, y_1^{(0)}, y_2^{(0)}, \dots)$ and $y^{(1)} = (y_0^{(1)}, y_1^{(1)}, y_2^{(1)}, \dots)$.

Note that if there are multiple input streams, we can refer to a single interleaved input **x** = $(x_0^{(0)}, x_1^{(1)}, \dots)$ Also, the output streams are multiplexed to create a single coded data stream **y**= $(y_0^{(0)}, y_0^{(1)}, y_1^{(0)} y_1^{(1)}, \dots)$ where **y** is the convolutional codeword. In addition, each element in the interleaved output stream **y** is a linear combination of the elements in the input stream **x** = $(x_0^{(0)}, x_0^{(1)}, \dots, x_0^{(k-1)}, x_1^{(0)}, x_1^{(1)}, x_1^{(1)}, \dots, x_1^{(k-1)}, \dots)$.

An impulse response $g^{(j)}$ is obtained from the encoder output by applying a single 1 at the input followed by a string of zeros, then strings of zeros are applied to all the other inputs (in the case of multiple inputs). The impulse responses for the encoder in our example are

$$g^{(0)} = (1011),$$
$$g^{(1)} = (1101).$$

The impulse responses are often referred to as generator sequences, because their relationship to the codewords generated by the corresponding convolutional encoder is similar to that between generator polynomials and codewords in a cyclic code. The generator sequences can be expressed in the following general form:

$$y_t^{(j)} = \sum_{l=0}^{m-1} x_{t-1} g_l^{(j)}. \qquad (1)$$

Each coded output sequence $y^{(j)}$ in a rate ${}^{1}\!/_{n}$ code is the convolution of the input sequence **x** and the impulse response $g^{(j)}$,

$$y^{(j)} = x * g^{(j)}. \qquad (2)$$

In vector form, this is expressed

$$y^{(j)} = \sum_{i=0}^{k-1} x^{(i)} * g_i^{(j)}, \qquad (3)$$

which can be developed thus

$$y_t^{(j)} = \sum_{i=0}^{k-1} \left[\sum_{l=0}^{m_i-1} x_{t-1}^{(i)}\, g_{i,l}^{(j)} \right]. \qquad (4)$$

We can express these forms as a matrix multiplication operation, thus providing a generator matrix similar to that developed for block codes. In fact, the primary difference arises from the fact that the input sequence is not necessarily bounded in length, and thus the generator and parity check matrices for convolutional codes are semi infinite. However, herein we introduce the **G** and **H** matrices as equivalent to a tail-biting convolutional code having finite length. Therefore, the generator and parity-check matrices will be as follows [4]:

$$G = \begin{bmatrix} G_m & \cdots & \cdots & G_0 & G_1 & \cdots & G_{m-1} \\ G_{m-1} & G_m & \cdots & \cdots & G_0 & \cdots & G_{m-2} \\ G_{m-2} & G_{m-1} & G_m & \vdots & \cdots & \ddots & \ddots \\ \ddots & \ddots & \ddots & \vdots & G_m & \cdots & \cdots \\ \cdots & G_1 & \vdots & G_{m-1} & G_m & \vdots & G_0 \\ G_0 & G_1 & \vdots & \vdots & \vdots & G_m & \vdots \\ \vdots & G_0 & G_1 & \vdots & \vdots & \ddots & \vdots \\ \cdots & \ddots & \ddots & G_1 & \cdots & \cdots & G_m \\ & \cdots & G_0 & & & & \end{bmatrix} \qquad (5)$$

where

$$G_l = \begin{bmatrix} g_{1,l}^{(1)} & g_{1,l}^{(2)} & \cdots & g_{1,l}^{(n)} \\ g_{2,l}^{(1)} & g_{2,l}^{(2)} & \cdots & g_{2,l}^{(n)} \\ \vdots & \vdots & \vdots & \vdots \\ g_{k,l}^{(1)} & g_{k,l}^{(2)} & \cdots & g_{k,l}^{(n)} \end{bmatrix} \tag{6}$$

Note that each block of *k* rows in the **G** matrix is a circular shift by *n* positions of the previous such block. In general, the parity-check matrix of a rate *k/n* tail-biting convolutional code with constraint length *m* is

$$H = \begin{bmatrix} P_0^T|\mathrm{I} & & \cdots & \cdots \; P_m^T|0 & P_{m-1}^T|0 & \cdots & P_1^T|0 & \\ P_1^T|0 & P_0^T|\mathrm{I} & \cdots & \cdots & P_m^T|0 & \cdots & P_2^T|0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ \cdots & P_1^T|0 & P_0^T|\mathrm{I} & \cdots & \cdots & \cdots & P_0^T|0 \\ P_m^T|0 & P_{m-1}^T|0 & \vdots & P_1^T|0 & P_0^T|\mathrm{I} & \vdots & \cdots \\ \ddots & \ddots & & \cdots & \ddots & P_1^T|0 & \ddots \\ & P_m^T|0 & P_{m-1}^T|0 & \cdots & \cdots & & P_0^T|\mathrm{I} \end{bmatrix} \tag{7}$$

where **I** is the $k \times k$ identity matrix, **0** is the $k \times k$ all zero matrix, and P_i, $i = 0, 1, ..., m$, is a $k \times (n - k)$ matrix whose entries are

$$P_i = \begin{bmatrix} g_{1,i}^{(k+1)} & g_{1,i}^{(k+2)} & \cdots & g_{1,i}^{(n)} \\ g_{2,i}^{(k+1)} & g_{2,i}^{(k+1)} & \cdots \; \vdots & g_{2,i}^{(k+1)} \\ \vdots & \vdots & & \vdots \\ g_{k,i}^{(k+1)} & g_{k,i}^{(k+2)} & g_{1,i}^{(k+1)} & g_{k,i}^{(n)} \end{bmatrix} \tag{8}$$

Here, $g_{p,i}$ is equal to 1 or 0 coresponding to whether or not the i^{th} stage of the shift register for the input contributes to output $j(i = 0, 1, ..., m; j = (k + 1), (k + 2), ..., n; p = 1, 2, ..., k)$. Since the last m bits serve as the starting state and are also fed into the encoder, there is an end-round-shift phenomenon for the last m columns of **H**.

Example 2: Consider the previous encoder shown in Example 1, assume that a block of k = 6 information bits are encoded. Then the tail-biting construction gives a binary (12, 6) code with generator and parity check matrices,

$$G = \begin{bmatrix} 11 & 01 & 10 & 11 & 00 & 00 \\ 00 & 11 & 01 & 10 & 11 & 00 \\ 00 & 00 & 11 & 01 & 10 & 11 \\ 11 & 00 & 00 & 11 & 01 & 10 \\ 10 & 11 & 00 & 00 & 11 & 01 \\ 01 & 10 & 11 & 00 & 00 & 11 \end{bmatrix} \tag{9}$$

and

$$H = \begin{bmatrix} 11 & 00 & 00 & 11 & 01 & 10 \\ 10 & 11 & 00 & 00 & 11 & 01 \\ 01 & 10 & 11 & 00 & 00 & 11 \\ 11 & 01 & 10 & 11 & 00 & 00 \\ 00 & 11 & 01 & 10 & 11 & 00 \\ 00 & 00 & 11 & 01 & 10 & 11 \end{bmatrix} \tag{10}$$

B. Degree distribution of Tanner graph for tail-biting convolution codes

Looking at the **H** matrix of a tail-biting convolutional code, we can notice that it is similar to the **H** matrix of an irregular LDPC code where the number of non-zero elements is not a fixed number per row and column. Our goal is to represent the tail-biting convolutional codes through Tanner graphs in order to decode them using the BP algorithm. Therefore, it is important to obtain the degree distribution of the Tanner graph which describes the number of edges into the bit and check nodes in irregular LDPC codes. The fraction of edges which are connected to degree-i bit nodes is denoted λ_i, and the fraction of edges which are connected to degree-i check nodes is denoted ρ_i. The functions

$$\lambda(x) = \lambda_1 x + \lambda_2 x^2 + \cdots + \lambda_i x^{i-1} + \cdots \tag{11}$$

$$\rho(x) = \rho_1 x + \rho_2 x^2 + \cdots + \rho_i x^{i-1} + \cdots \tag{12}$$

are defined to describe the degree distributions.

III. TURBO CODES

To replace the traditional decoders of turbo codes by the BP decoder, we have to obtain the parity-check matrix for the turbo code as was done in the previous section for the tail-biting convolutional codes.

A. Parity check matrix for turbo codes

Let us consider a recursive systematic convolutional (RSC) code C_0 of rate $R = 1/2$. It has two generator polynomials $g_1(X)$ and $g_2(X)$ of degree $v + 1$, where v is the memory of the encoder. Let $u(X)$ be the input of the encoder and $x_1(X)$and $x_2(X)$ its outputs. We consider this code as a block code obtained from the zero-tail truncation of the RSC. Using the parity check matrix **H** of the RSC, we can do some column permutations and rewrite the **H** matrix as H_{new}, where $H_{new} = [H_1\ H_2]$. As mentioned before, we consider a conventional turbo code *C*, resulting from the parallel concatenation of two identical RSC codes $C_0$and whose common inputs are separated by an interleaver of length *N*, represented by matrix **M** of size $N \times N$ with exactly one nonzero element per row and column. It is well known that the superior performance of turbo codes is primarily due to the interleaver, i.e., due to the cycle structure of the Tanner graph [17]. Hence, the parity check matrix of the whole turbo code is (for a detailed proof, the reader is referred to [18]):

$$H_{turbo} = \begin{bmatrix} H_2 & H_1 & 0 \\ H_2M^T & 0 & H_1 \end{bmatrix}. \qquad (13)$$

Example 3: Let us now consider the special case of a RSC code C_0 of rate $R = 1/2$ whose input *u(X)* has a finite degree *N−1* (i.e. the input vector has size *N*). Its parity-check matrix **H** can now be written as an *N ×2N* matrix over GF(2) whose coefficients are fixed by its generator. The first (respectively second) N ×N part of **H** consists of shifted rows representing the coefficients of $g_2(X)$ (respectively $g_1(X)$). For example, choosing $g_1 = 101,\ g_2 = 111,$ and *N = 8,* we have:

Note that the number of non-zero elements per row and
per column in the "diagonal" sub-matrices $\mathbf{H_1}$and $\mathbf{H_2}$ is upper bounded by *L = v + 1*, the constraint length of the constituent codes, which is always very small in comparison to the length of the interleaver. Also, the interleaver does not change the weights of the sub-matrix H_2M^T. As with the tailbiting convolutional code, the **H**

$$\mathbf{H} = \begin{bmatrix} 11100000 & | & 10100000 \\ 01110000 & | & 01010000 \\ 00111000 & | & 00101000 \\ 00011100 & | & 00010100 \\ 00001110 & | & 00001010 \\ 00000111 & | & 00000101 \\ 00000011 & | & 00000010 \\ 00000001 & | & 00000001 \end{bmatrix}$$

matrix for a turbo code can also be seen as the **H** matrix of an irregular LDPC code, since the weight of non-zero elements per row and column is not strictly constant, but always very small compared to the size of the parity-check matrix. Then, the parity-check matrix for the mentioned turbo code in our example will be as follows:

$$\mathbf{H} = \begin{bmatrix} 10100000 & 11100000 & 0000000 \\ 01010000 & 01110000 & 0000000 \\ 00101000 & 00111000 & 0000000 \\ 00010100 & 00011100 & 0000000 \\ 00001010 & 00001110 & 0000000 \\ 00000101 & 00000111 & 0000000 \\ 00000010 & 00000011 & 0000000 \\ 00000001 & 00000001 & 0000000 \\ 00001001 & 0000000 & 11100000 \\ 01010000 & 0000000 & 01110000 \\ 00000101 & 0000000 & 00111000 \\ 10010000 & 0000000 & 00011100 \\ 00000110 & 0000000 & 00001110 \\ 10100000 & 0000000 & 00000111 \\ 00000010 & 0000000 & 00000011 \\ 00100000 & 0000000 & 00000001 \end{bmatrix}$$

Following (9) and (10) provided in the previous section, the degree distribution of this turbo code is given by

$$\lambda(x) = 0.291667x + 0.083333x^2 + 0.25x^3 + 0.16667x^4 + 0.208333x^5 \qquad (14)$$

$$\rho(x) = 0.083333x^2 + 0.125x^3 + 0.033333x^4 + 0.208333x^5 + 0.25x^6 \qquad (15)$$

IV. WIMAX AND LTE CODING STRUCTURE

To address the low and high rate requirements of LTE, the 3rd Generation Partnership Project (3GPP) working group undertook a rigorous study of advanced channel coding candidates such as tail-biting convolutional and turbo codes for low and high data rates, respectively. We investigate here the application of the BP decoder for the proposed turbo code in LTE systems. Meanwhile, a rate ½, memory-6 tail-biting convolutional code has been adopted in the WiMAX (802.16*e*) system, because of its best minimum distance and the smallest number of minimum weight codewords for larger than 32-bit payloads which is used for both frame control header (FCH) and data channels. In fact, we will focus here on the FCH which has much shorter payload sizes (12 and 24 bits) as shown in the next subsection.

A. Tail-biting convolutional code in 802.16e

Here, we briefly describe the WiMAX frame control header structure. In the WiMAX Orthogonal Frequency Division Multiplexing (OFDM) physical layer, the payload size of the frame control header is either 24 bits or 12 bits and the smallest unit for generic data packet transmission is one subchannel. A subchannel consists of 48 QPSK symbols (96 coded bits). At a code rate of ½, one subchannel translates to 48 bits as the smallest information block size. Currently, the FCH payload bits are repeated to meet the minimum number (48) of encoder information bits. The generator polynomials for the rate ½ WiMAX tail-biting convolutional code are given by $g_1 = (1011011)$ and $g_2 = (1111001)$in binary notation. According to [19], these generator polynomials have the best $d_{min} = (minimum\, distance)$ and $n_{min} = (number\ of\ codewords\ with\ weight\ d_{min})$
for payload sizes ≥ 33 bits and for some payload sizes between 25 and 33 bits, under the constraint of memory size $m = 6$ and code rate ½.

B. Turbo code in LTE system

The 3GPP turbo code is a systematic parallel concatenated convolutional code (PCCC) with two 8-state constituent encoders and one turbo code internal interleaver. Each constituent encoder is independently terminated by tail bits. For an input block size of K bits, the output of a turbo encoder consists of three length K streams, corresponding to the systematic and two parity bit streams (referred to as the “Systematic”, “Parity 1”, and “Parity 2” streams in the following), respectively, as well as 12 tail bits due to trellis termination. Thus, the actual mother code rate is slightly lower than 1/3. In LTE, the tail bits are multiplexed to the end of the three streams, whose lengths are hence increased to (K + 4) bits each [5]. The transfer function of the 8-state constituent code for the PCCC is:

$$G(D) = \left[1, \frac{g_1(D)}{g_0(D}\right], \tag{16}$$

$$g_0(D) = 1 + D^2 + D^3, \tag{17}$$

$$g_1(D) = 1 + D + D^3. \tag{18}$$

The initial value of the shift registers of the 8-state con- stituent encoders will be all zeros when starting to encode the input bits. The output from the turbo encoder is $d_k^{(0)} = x_k,\ d_k^{(1)} = z_k$, and $d_k^{(2)} = z_k'$ for k = 0, 1, 2, . . . , K-1. If the code block to be encoded is the 0-th code block and the number of filler bits is greater than zero, i.e., $F > 0$, then the encoder will set $c_k = 0$, k = 0,. . . , (F-1) at its input and will set $d_k^{(0)} =< NULL >$, k = 0,. . . , (F-1) and $d_k^{(0)} =< NULL >$,, k = 0,. . . , (F-1) at its output [5]. The bits input to the turbo encoder are denoted by $c_0, c_1, c_2, c_3, \dots, c_{k-1}$,, and the bits output from the first and second 8-state constituent encoders are denoted by $z_0, z_1, z_2, z_3, \dots, z_{k-1}$ and $z_0', z_1', z_2', z_3', \dots, z_{k-1}'$,respectively. The bits output from the turbo code internal interleaver are denoted by $C_0', C_1', \dots, C_{K-1}'$, and these bits are to be the input to the second 8-state constituent encoder.

V. SIMULATION RESULTS

Considering the previous example of the tail-biting convolu-tional code in WiMAX systems and binary transmission over an AWGN channel, the BP algorithm as in [4] is compared with the maximum-likelihood (ML) Viterbi type algorithm to decode the same tail-biting convolutional code [9, 12] To determine by simulation the maximum decoding performance capability of each algorithm, at least 300 codeword errors are detected at each SNR value. Figure 1 shows a performance comparison between the two mentioned decoding algorithms for a payload size of 24 bits. Note that the maximum number of iterations for the BP algorithm is 30 iterations. The imulation results show that the proposed BP algorithm exhibits a slight performance penalty with respect to the ML Viterbtype algorithm. However, ince the BP decoder is less complex than this traditional decoder and enables a unified decoding approach, this loss in BER performance is deemed acceptable

In addition, a comparison between the same short length code using the BP and ML Viterbi algorithms has been performed in Figure 2. In this case, a loss of 1.85 dB or less in FER compared with the traditional decoder is observed.

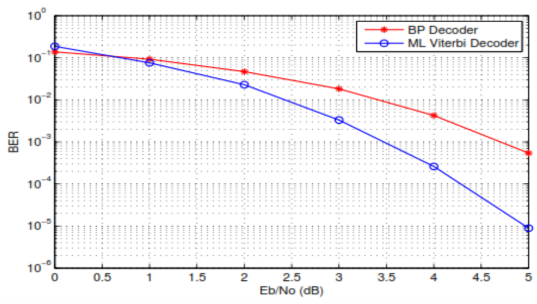

Figure 1. Comparisons of BER for length-24 rate ½ tail-biting convolutional code.

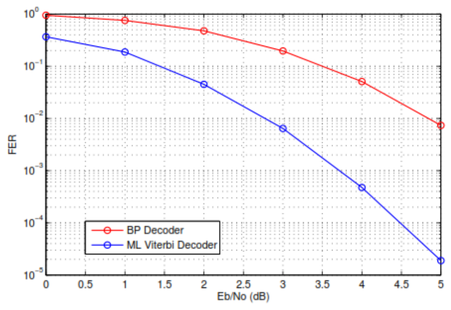

Figure 2. Comparisons of FER for length-24 rate ½ tail-biting convolutional code.

In Figure 3, we report simulation results for the AWGN channel for the LTE turbo code that was studied in the previous section. When compared to the traditional MAP and SOVA decoders [11, 12], the BP algorithm is about 1.7 dB worse at a BER value of 10^{-2}. Also, as we obtained a general form for the parity-check matrices of tail-biting convolutional and turbo codes, then we can enhance the performance by investigating other decoding algorithms which are also applicable for LDPC codes.

For further research, we propose exploring alternatives to the flooding schedule usually adopted for LDPC codes to enhance the BER performance.

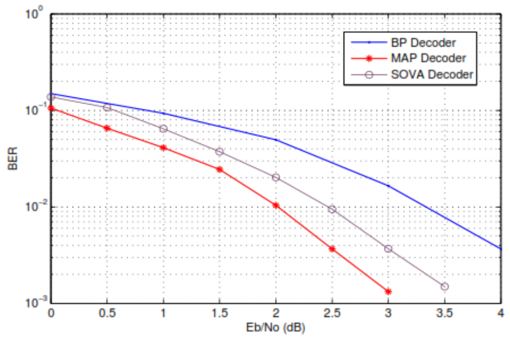

Figure 3. Comparisons of BER for length-40 rate $1/3$ turbo code.

VI. COMPLEXITY COMPARISON

A direct comparison between the complexity of different decoding algorithms is implementation dependent. Starting with the traditional decoders for turbo codes, the MAP process computes the log-likelihood for all paths in the trellis. The MAP algorithm estimates the metric for both received binary zero and a received binary one, then compares them to determine the best overall estimate. The SOVA process only considers two paths of the trellis per step: the best path with a data bit of zero and the best path with a data bit of one. In addition, it utilizes the difference of the log-likelihood function for each of these paths. However, the SOVA is the least complex of the two algorithms in terms of number of calculations [14]. Finally, for the BP algorithm, the decoding complexity per iteration grows linearly with the number of edges (the number of messages passed per iteration is twice the number of edges in the graph E). Moreover, one can argue that the complexity of the operations at the variable and check nodes frequently scales linearly with

$$E = \sum_{i=1}^{d_{vmax}} nv_i i = \sum_{i=1}^{d_{cmax}} nc_j j \qquad (19)$$

Following the notations of Luby et al. [13], consider a Tanner graph with n left nodes, where $v_i = \frac{n_i}{n}$ represents the fraction of left nodes of degree $i > 0$ and d_v(resp. d_c) isthe variable node degree (res. check node degree). Also, $C_j = \frac{r_j}{r}$ is defined to be the fraction of right nodes of degree $j > 1$.

The complexity comparisons of the various decoding algorithms are shown in Table I where k is the number of systematic bits and v is the memory order of the encoder. The table gives the operations per iteration for MAP, SOVA, and BP decoding for the horizontal (H) and vertical (V) step. Note that, for the BP algorithm, the complexity per information bit is [20]

Table I
DECODER COMPLEXITY COMPARISONS

	MAP	SOVA		BP
$\oplus$	$2 \times 2^k \times 2^v + 6$	$2 \times 2^k \times 2^v + 9$	(H)	$(3d_{v\,max} - 4)M2^2$
			(V)	$2d_{v\,max}M$
$\otimes$	$5 \times 2^k \times 2^v + 8$	$2^k \times 2^v$	(H)	$(3d_{v\,max} - 4)M2^2$
			(V)	$2d_{c\,max}d_{v\,max}M$
max*	0	$2 \times 2^v - 1$	(H)	0
			(V)	0

$$K = \frac{1}{\sum_i {}^{\lambda_i}/_{i} - \sum_i {}^{\rho_i}/_{i}} \int_{pt}^{p0} \frac{dp}{p \log\left(\frac{p}{\sum_i \lambda_i f_i(p)}\right)}, \qquad (20)$$

Considering example 3, Figure 4 shows a comparison between these mentioned algorithms in terms of the number of operations required in the implementations. In comparison with MAP and SOVA decoders, BP exhibits the lowest implementation complexity over all the required operations.

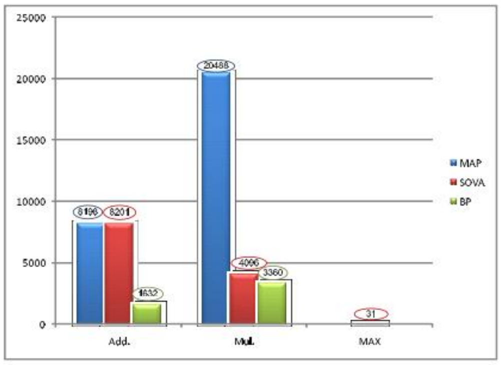

Figure 4. Comparison of MAP, SOVA, and BP decoders in terms of number of operations.

VII. CONCLUSION

In this paper, the feasibility of decoding arbitrary tailbiting convolutional and turbo codes using the BP algorithm was demonstrated. Using this algorithm to decode the tailbiting convolutional code in WiMAX systems speeds up the error correction convergence and reduces the decoding computational complexity with respect to the ML-Viterbi-based algorithm. In addition, the BP algorithm performs a non-trellis based forward-only algorithm and has only an initial decoding delay, thus avoiding intermediate decoding delays that usually accompany the traditional MAP and SOVA components in LTE turbo decoders. However, with respect to the traditional decoders for turbo codes, the BP algorithm is about 1.7 dB worse at a BER value of 10−2. This is because the nonzero element distribution in the parity-check matrix is not random enough. Also, there are a number of short cycles in the corresponding Tanner graphs. Finally, as an extended work, we propose the BP decoder for these codes in a combined architecture which is advantageous over a solution based on two separate decoders due to efficient reuse of computational hardware and memory resources for both decoders. In fact, since the traditional turbo decoders (based on MAP and SOVA components) have a higher complexity, the observed loss in performance with BP is more than compensated by a drastically lower implementation complexity. Moreover, the low decoding complexity of the BP decoder brings about endto- end efficiency since both encoding and decoding can be performed with relatively low hardware complexity.

REFERENCES

[1]. C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," IEEE Intl. Conf. on Commun., vol. 2, Geneva, Switzerland, pp. 1064-1070, 1993.

[2]. R. M. Tanner, "A recursive approach to low complexity codes," IEEE Trans. Inform. Theory, vol. 27, no. 5, pp. 533-547, 1981.

[3]. G. D. Forney, Jr., "Codes on graphs: normal realizations," IEEE Trans. Inform. Theory, vol. 47, no. 2, pp. 520–548, 2001.

[4]. H. H. Ma and J. K. Wolf, "On Tail Biting Convolutional Codes," IEEE Trans. On Commun., vol. 34, no. 2, pp. 104-111, 1986.

[5]. 3GPP TS 45.003, "3rd Generation Partnership Project; Technical Specification Group GSM/EDGE Radio Access Network; Channel Coding (Release 7)," February, 2007. [6] IEEE Std 802.16-2004, "IEEE Standard for Local and

Metropolitan Area Networks – Part 16: Air Interface for Fixed Broadband Wireless Access Systems,” October, 2004.

[6]. IEEE Std P802.16e/D10, “IEEE Standard for Local and Metropolitan Area Networks – Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems,” August, 2005.

[7]. D. J. C. MacKay, “Good error-correcting codes based on very sparse matrices,” IEEE Trans. Inform. Theory, vol. 45, no. 2, pp. 399-431, 1999.

[8]. N. Wiberg, “Codes and Decoding on General Graphs,” Linkoping Studies in Science and Technology, Dissertation No. 440, Linkoping University, -Linkoping, Sweden, 1996.

[9]. R. J. McEliece, D. MacKay, and J.-Fu Cheng, “Turbo Decoding as an Instance of Pearl’s "Belief Propagation" Algorithm,” IEEE Trans. On Commun., vol. 16, no. 2, pp. 140-152, 1998.

[10]. G. Colavolpe, “Design and performance of turbo Gallager codes,” IEEE Trans. On Commun., vol. 52, no. 11, pp. 1901-1908, 2004.

[11]. T. T. Chen, and S-He Tsai, “Reduced-Complexity Wrap-Around Viterbi Algorithms for Decoding Tail-Biting Convolutional Codes,” 14th European Wireless Conference, Jun. 2008.

[12]. M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, “Improved LDPC Codes Using Irregular Graphs,” IEEE Trans. Inform. Theory, vol. 47, no. 2, pp. 585-598, 2001.

[13]. Giulietti A., “Turbo Codes: Desirable and Designable,” IKluwer Academic Publishers, ISBN: 1-4020-7660-6, 2004.

[14]. P. Elias, “Coding for Noisy Channels,” IRE Conv. Rec., vol. 3, pt. 4, pp. 37–46, 1955.

[15]. A. J. Viterbi, “Convolutional Codes and their Performance in Communication Systems,” IEEE Trans. On Commun., vol. 19, no. 15, pp. 751-772, 1971.

[16]. C. Poulliat, D. Declercq, and T. Lestable, “Efficient Decoding of Turbo Codes with Nonbinary Belief Propagation,” EURASIP Journal on Wireless Communications and Networking, vol. 2008, no. 473613,

[17]. 2008.

[18]. A. Refaey, S. Roy, and P. Fortier, “On the Application of BP Decoding to Convolutiona and Turbo Codes,” Asilomar Conference on Signals, Systems, and Computers, Nov. 2009.

[19]. P. Stahl, J. B. Anderson, and R. Johannesson, “Optimal and near-optimal encoders for short and moderate-length tail-biting trellises,” IEEE Trans. Inform. Theory, vol. 45, no. 7, pp. 2562-2571, 1999.

[20]. W. Yu, M. Ardakani, B. Smith, and F. FKschischang, “Complexityoptimized low-density parity-check codes for Gallager decoding algorithm B,” IEEE International Symposium on Information Theory (ISIT)s, Sep. 2005.

[21]. Ahmed Refaey, Sébastien Roy, and Paul Fortier, “A New Approach for FEC Decoding Based on the

[22]. BP Algorithm in LTE and WiMAX Systems”, IEEE Transaction,2011.