Turbo Decoding Using the Sectionalized Minimal Trellis of the Constituent Code: Performance-Complexity Trade-off

Guilherme Luiz Moritz, Richard Demo Souza, Cecilio Pimentel, Marcelo Eduardo Pellenz, Bartolomeu F. Uchôa-Filho, and Isaac Benchimol

Abstract—The performance and complexity of turbo decoding using rate $k/n$ constituent codes are investigated. The conventional, minimal and sectionalized trellis modules of the constituent convolutional codes are utilized. The performance metric is the bit error rate (BER), while complexity is analyzed based on the number of multiplications, summations and comparisons required by the max-log-MAP decoding algorithm. Our results show that the performance depends on how the systematic bits are grouped in a trellis module. The best performance is achieved when the $k$ systematic bits are grouped together in the same section of the module, so that the log-likelihood ratio (LLR) of the $k$-bit vector is calculated at once. This is a characteristic of the conventional trellis module and of some of the sectionalizations of the minimal trellis module. Moreover, we show that it is possible to considerably reduce the decoding complexity with respect to the conventional trellis if a particular sectionalization of the minimal trellis module is utilized. In some cases, this sectionalization is found within the best performing group, while in some other cases a small performance loss can be traded off for a large complexity reduction.

I. INTRODUCTION

It is known that error correcting codes can be responsible for a large part of the baseband power consumption of a communication device [1]. For instance, in [2] the authors show that the Viterbi decoding [3] accounts for 35% of the overall power consumption of a typical 802.11 [4] receiver. Recent studies reinforce the importance of the energy efficiency by showing that enormous gains in terms of computational capacity in the last decades have been followed by an unimpressive improvement in the nominal capacity of the batteries [5], [6]. For these reasons, besides throughput and coverage, energy efficiency has become a major goal in wireless communications.

Turbo codes, a class of powerful error-correcting codes introduced by Berrou and Glavieux [7], are currently used in many modern communication systems [8], [9]. Therefore, reductions in the decoding complexity of turbo codes can have a positive impact on the energy efficiency and consequently on the battery lifetime of many wireless devices.

As originally proposed in [7], turbo codes are based on the parallel concatenation of two systematic recursive constituent convolutional encoders of rate $r = \frac{k}{n}$ connected via an interleaver, resulting in a coding scheme of rate $r = \frac{k}{m}$, which can be iteratively decoded in an efficient way. The usual approach to increase the rate of the turbo coding scheme is to puncture selected bits of the turbo codeword [10], [11]. However, Douillard and Berrou [12] proposed the construction of higher rate turbo codes by means of constituent convolutional codes of rate $r = \frac{m}{n+1} > \frac{1}{2}$. For instance, if $m = 2$ then two $r = \frac{2}{3}$ constituent codes are used to generate an $r = \frac{2}{4}$ turbo code without puncturing. Higher rate turbo codes generated by the technique in [12] present some advantages over classical punctured turbo codes [7], as better iterative process convergence and less performance loss when using suboptimal decoding algorithms such as the max-log-MAP algorithm [13]. If puncturing is still needed to achieve a required rate, fewer bits have to be discarded when compared to the conventional scheme employing $r = \frac{k}{2}$ constituent codes, resulting in less degradation of the correcting capability of the constituent code [12].

Optimized constituent codes of rate $r = \frac{k}{m} > \frac{1}{2}$ are presented by Graell i Amat et al in [14]. Turbo codes constructed with the family of constituent codes given in [14] are also shown to outperform some higher rate turbo codes obtained by puncturing the output of a rate $r = \frac{k}{2}$ constituent code. Turbo codes as those discussed in [12], [14] are used, for instance, in the WiMAX standard [8].

One major drawback of such turbo codes constructed with constituent codes of rate $r = \frac{k}{m} > \frac{1}{2}$ is the decoding complexity, as it increases exponentially with $k$ and with the constraint length for various decoding algorithms. One approach to reduce the decoding complexity is to consider alternative trellis representations for the constituent codes, other than the conventional trellis representation. In [15], the authors present a method for constructing the minimal trellis representation for a systematic recursive convolutional encoding matrix, typical of turbo code constituent encoders. This construction is based on the minimal trellis obtained for nonrecursive nonsystematic convolutional encoding matrices presented in [16], which represents the coded sequences minimally under various complexity measures. The subject of minimal trellis was given a great deal of attention in the
90’s, however recently a renewed interest has emerged due to its good error performance versus decoding complexity trade-off [17]–[19], as well as due to its potential reductions in power consumption and hardware utilization in real implementations [20].

A trellis operation called sectionalization [21] can be applied to the minimal trellis in order to obtain new trellis topologies more suited to a given application. This operation removes the states at a given depth and redirects the incoming and outgoing edges of these states [21]. Starting from the minimal trellis module, sectionalizations at more and more depths generate shorter and shorter trellis modules up to the shortest one, namely, the conventional trellis module.

In this paper, we analyze both the computational complexity, in terms of arithmetic operations, and the bit error rate (BER) performance when the max-log-MAP algorithm is applied over the conventional, minimal and sectionalized trellis representations of the systematic recursive convolutional constituent encoders of rate \( r = \frac{k}{n} \). We derive expressions for the decoding arithmetic complexity and investigate the BER performance through computer simulations when turbo codewords are binary phase shift keying (BPSK) modulated and sent over the additive white Gaussian noise (AWGN) channel.

It is known that the minimal trellis representation is less complex, in terms of the theoretical trellis complexity measure defined in [16], than the conventional trellis representation. However, we show that the use of the minimal trellis in turbo decoding using the max-log-MAP algorithm [13] brings a loss in BER performance, while not necessarily reducing the number of required arithmetic operations. This behavior contrasts with the one observed when the minimal trellis is used with the Viterbi algorithm for the decoding of convolutional codes, in which case there is no loss in performance and the number of required operations can be potentially reduced.

Aiming at turbo coding applications, we further exploit the use of the minimal trellises by considering their sectionalized versions in turbo decoding. We are then able to achieve the same BER performance as when using the conventional trellis, while reducing the number of required arithmetic operations. Therefore, by using the sectionalized version of the minimal trellis we are able to reduce the complexity of high-rate turbo decoding, without any loss in performance. Moreover, if a small loss in performance is tolerable, it is possible to achieve larger savings in decoding complexity, so that a performance-complexity trade-off is achieved. We provide examples of turbo codes for which the presented technique is able to reduce the turbo decoding complexity by 8% without any loss in performance, while reaching 60% of complexity reduction with a performance degradation of 0.25 dB. Additionally, we perform a density evolution analysis [22] of the turbo decoders based on the conventional, sectionalized and minimal trellises in order to better understand their difference in performance.

The rest of this paper is organized as follows. Section II discusses the conventional, minimal and sectionalized trellis modules. Section III analyzes the arithmetic complexity of the max-log-MAP algorithm for turbo decoding over the trellis modules defined in Section II. In Section IV the performance-complexity trade-off of the different trellis representations is investigated, while Section V presents a density evolution analysis of the conventional, sectionalized and minimal trellises based turbo decoders. Finally, Section VI concludes the paper.

II. TRELLIS REPRESENTATIONS OF A CODE

Convolutional codes can be represented by a semi-infinite trellis which (apart from a short transient in its beginning) is periodic, the shortest period being called a trellis module [18]. In general terms, a trellis module \( M \) for a rate \( r = \frac{k}{n} \) convolutional code \( C(n, k, \nu) \) consists of \( n' \) trellis sections (from depth 0 to depth \( n' \)), \( 2^\nu_i \) states at depth \( i \), \( 2^{b_i} \) branches emanating from each state at depth \( i \), and \( a_i \) bits labeling each edge from depth \( i \) to depth \( i + 1 \) (for \( 0 \leq i \leq n' - 1 \)). The overall constraint length is \( \nu \). The quantities \( \nu_i \) and \( b_i \) are usually called the state and branch complexities, respectively. The trellis complexity of a trellis module, in symbols per information bit, is [16]:

\[
TC(M) = \frac{1}{k} \sum_{i=0}^{n'-1} a_i \cdot 2^{\nu_i+b_i}.
\]

The most common trellis module representation for a convolutional code is the conventional trellis module \( M_{conv} \), which has a very regular structure consisting of a single \( n' = 1 \) section with \( 2^\nu \) initial states and \( 2^\nu \) final states; each initial state is connected to \( 2^k \) directed edges to final states, and each edge is labeled by \( n \) bits. For instance, the conventional trellis module of code \( C(4,2,3) \), which is the constituent code of the WiMAX standard [8] turbo code, has a single \( n' = 1 \) section with \( 2^\nu = 8 \) states. There are \( 2^k = 4 \) edges emanating from each state, and they are labeled by \( n = 4 \) bits.

A. Minimal Trellis Module

Another important trellis module representation for a convolutional code is the minimal trellis module, \( M_{min} \), introduced by McEliece and Lin [16]. The minimal trellis module was proposed for a nonrecursive nonsystematic encoding matrix, and allows an irregular trellis structure consisting of \( n' = n \) sections and \( a_i = 1 \) bit per edge \( i \). The edge complexity \( b_i \) is 1 for exactly \( k \) sections, those sections associated with information bits, and is 0 for the remaining sections. This representation is minimal in terms of the trellis complexity in (1), and, among all trellis representations with \( n \) sections, it also minimizes many other commonly accepted theoretical complexity measures such as the merge complexity, the maximum and the total number of states [21, Theorem 4.26].

Until recently, the concept of minimal trellis module had always been attached to a code, without any reference to a particular encoder. However, in [15], [23], a method has been introduced to construct the minimal trellis module of a convolutional code based on its recursive systematic encoding matrix, as this is the encoder of interest in turbo codes. In this construction, an equivalent nonrecursive nonsystematic generator matrix is first obtained from the recursive systematic one, and then the minimal trellis is built following the construction presented in [16]. The final minimal trellis module is produced...
by altering the association of the edges to the information bits, in the trellis sections with \( b_i = 1 \), in such way that the information bit value equals the (coded) bit labeling the edge. It is important to highlight that this change in convention has no impact on the topology of the trellis module nor on its trellis complexity.

Using the method introduced in [15], [23], we construct the minimal trellis module of the WiMAX constituent code \( C(4, 2, 3) \) which has a recursive systematic encoding matrix. Notice that this module, shown in Figure 1, is composed of four irregular sections (\( n' = 4 \)), where only the first and the second sections carry information bits while the other two sections are informationless. Note that there is a significant reduction in terms of trellis complexity with respect to the conventional trellis module, as \( TC(M_{\text{conv}}) = 64 \) and \( TC(M_{\text{min}}) = 48 \) symbols per information bit.

B. Sectionalized Trellis Module

Consider a minimal trellis module \( M_{\text{min}} \) composed of \( n \) sections. The sectionalization of \( M_{\text{min}} \) at depth \( i \) for \( i = 1, \cdots, n - 1 \) is defined as the removal of states at depth \( i \) and the connection of states from \( i - 1 \) directly to states at \( i + 1 \), but only if there exists in \( M_{\text{min}} \) a path between the initial state in \( i - 1 \) and the final state in \( i + 1 \). The edge labels of the new section are obtained by the concatenation of the edge labels of the paths connecting states from \( i - 1 \) to \( i + 1 \) on \( M_{\text{min}} \). The resulting trellis module is composed of \( n' = n - 1 \) sections. The process can be extended, with \( 2^{n-1} \) possible ways of sectionalizing a minimal trellis module. In order to represent the different ways of sectionalizing a minimal trellis module, a binary sectionalization vector (vetsec) is defined. Its \( i \)th entry, \( \text{vetsec}_i \), \( 1 \leq i \leq n - 1 \), is equal to 1 when \( M_{\text{min}} \) is sectionalized at depth \( i \), otherwise \( \text{vetsec}_i = 0 \). We denote by \( a_i \) and \( b_i \) the number of bits that label the edges, the state, and the branch complexities, respectively, associated with depth \( i \) of the sectionalized trellis.

As an example, in Figure 2 we show a sectionalized version of the minimal trellis module shown in Figure 1, with \( \text{vetsec} = \{0, 1, 0\} \), that is, \( M_{\text{min}} \) is sectionalized at depth 2 (the states at depth 2 are deleted, with the states at depth 1 joined directly to the ones at depth 3). The sectionalized trellis module has now \( n' = 3 \) and \( a_0 = 1, a_1 = 2, a_2 = 1 \). The trellis complexity of the sectionalized trellis module shown in Figure 2 is 48 symbols per information bit, the same as that of the minimal trellis in Figure 1. Note that the minimal trellis and the conventional trellis are particular cases of the sectionalized trellis with sectionalization vectors \( \text{vetsec} = \{0, 0, 0\} \) and \( \text{vetsec} = \{1, 1, 1\} \), respectively.

III. COMPLEXITY OF MAX-LOG-MAP TURBO DECODING

In this paper we consider the widely used iterative max-log-MAP decoding algorithm [13]. Moreover, a detailed description of the algorithm for turbo codes constructed with constituent codes of rate \( r = \frac{k}{n} > \frac{1}{2} \) can be found for instance in [24], [25], and the analysis that follows is based on these references.

In the original turbo codes, using rate \( r = \frac{1}{2} \) constituent codes, the log-likelihood ratio of the information bits are calculated one at a time, as each trellis section is labeled by a single information bit. However, this is not necessarily the case for turbo codes using constituent codes of rate \( r = \frac{k}{n} \), \( k > 1 \). If decoding is performed using the conventional trellis module, then the bits are decoded in groups of \( k \) bits, as each edge in a conventional trellis module is associated with an information word of \( k \) bits. If decoding is performed using the minimal trellis, then the information bits are decoded one at a time, as the edges of the sections containing information bits represent a single information bit each. Finally, if the sectionalized minimal trellis is utilized, then the information bits can be decoded in groups of one to \( k \) bits at a time, depending of the sectionalization process. As we discuss later in this paper, the way the bits are grouped for decoding has a considerable impact on the BER performance.
The trellis complexity defined in (1) is a theoretical measure that captures the cost of Viterbi decoding. In order to define a complexity metric for turbo decoding over the sectionalized trellis module, and therefore over the particular cases of the minimal and the conventional trellis, we investigate the mathematical complexity of the max-log-MAP algorithm. We focus on the operations performed over a single section of a trellis module. In particular, consider a trellis section with \(2^{k_{\text{sec}}}\) initial states and \(2^{k_{\text{sec}}+1}\) final states, representing \(b_k\) information bits. These section edges are labeled by words of length \(a_k\) bits. We next summarize the main decoding steps in each section of the sectionalized trellis module.

### A. Decoding Steps

A different branch metric \(\psi_i(q)\), with \(q = 0, \cdots, 2^{k_{\text{sec}}}-1\), is determined for each of the possible information words of \(b_k\) bits. One of these metrics, for instance \(\psi_i(0)\), is taken as a reference in order to calculate the other \((2^{k_{\text{sec}}} - 1)\) log-likelihood metrics \(\Lambda_i(q)\) as follows:

\[
\Lambda_i(q) = \psi_i(0) - \psi_i(q), \quad (2)
\]

\[
\psi_i(q) = \max_{q+1} \left[ \alpha_{i-1}(l') + \gamma_i(l', l) + \beta_i(l) \right], \quad (3)
\]

where \(0 \leq l' \leq 2^{k_{\text{sec}}}-1\) is one of the \(2^{k_{\text{sec}}}\) initial states while \(0 \leq l \leq 2^{k_{\text{sec}}+1}-1\) is one of the \(2^{k_{\text{sec}}+1}\) final states in that section of the trellis module, while the notation \(q \rightarrow l', l\) means that the transition between initial state \(l'\) to final state \(l\) is caused by this particular \(b_k\) bits information word, denoted as an integer \(q\). Moreover, \(\alpha_i\) and \(\beta_i\) are defined as

\[
\alpha_i(l) = \max_{l, l'} \left[ \alpha_{i-1}(l') + \gamma_i(l', l) \right], \quad (4)
\]

and

\[
\beta_i(l) = \max_{l, l'} \left[ \beta_{i+1}(l') + \gamma_{i+1}(l, l') \right], \quad (5)
\]

where \(l, l'\) means that state \(l\) is connected to state \(l'\), no matter the information word that yields that connection, and

\[
\gamma_i(l', l) = \left\{ \begin{array}{ll}
\sum_{u=1}^{s} a_k^{i+1} \cdot x_{i,u} \cdot \Gamma_i(l', l) + \Lambda_i(l', l) \\
\end{array} \right. \quad (6)
\]

where \(r_{i,u}\) is the \(u\)-th symbol of the \(a_k\)-tuple received at depth \(i\), \(x_{i,u}\) is the BPSK symbol which should be transmitted at that instant if the transition between states \(l'\) and \(l\) happened, while \(\Lambda_i(l', l)\) is the extrinsic information \(\text{(a priori)}\) of that transition. Notice that (4) and (5) are recursive, and we assume that their initial conditions are

\[
\alpha_0(l) = \left\{ \begin{array}{ll}
0, & \text{if } l = 0 \\
-\infty, & \text{otherwise}
\end{array} \right. \quad (7)
\]

and

\[
\beta_{\tau}(l) = \left\{ \begin{array}{ll}
0, & \text{if } l = 0 \\
-\infty, & \text{otherwise}
\end{array} \right. \quad (8)
\]

where \(\tau\) is the last depth of the last trellis module in the complete trellis used for decoding.

It is important to note that the LLRs are calculated in a different manner depending on the type of trellis being used:

- In the case of the conventional trellis module, \(2^k - 1\) different LLRs are calculated per section;
- In the case of the minimal trellis module, single bit LLRs are calculated at the sections containing information bits;
- When considering a sectionalized trellis module, the LLRs are calculated for groups of bits (from 1 to \(k\) bits) in the sections containing information bits, depending on how many information bits are grouped at each trellis section.

For instance, consider the case of the WiMax turbo code (with constituent code \(C(4,2,3)\)). Following the definition of the algorithm in (2) and (3), in the case of the conventional trellis module we calculate three non-binary LLRs per section, as \(k = 2\) information bits are related with each section of the module. In the case of the sectionalized trellis let us consider a binary sectionalization vector \(\text{vsec} = \{1,0,0\}\), so that there are three sections within the module with the two information bits grouped in the first section while the other two sections are informationless. In the first section of this sectionalized trellis module we also calculate three non-binary LLRs as the \(k = 2\) information bits are grouped in this section, while in the next two sections we do not calculate any LLR (but we do calculate the corresponding \(\alpha\)'s, \(\beta\)'s and \(\gamma\)'s). In the case of the minimal trellis module, which can be seen in Figure 1, we calculate one binary LLR in the first section and one binary LLR in the second section (as these sections are related to one information bit each), and we do not calculate any LLR in the following two sections as they are informationless (note that, as in the case of the sectionalized trellis, we do calculate the corresponding \(\alpha\)'s, \(\beta\)'s and \(\gamma\)'s).

### B. Complexity Metric

A complexity metric is developed next where only mathematical operations – summations (\(S\)), multiplications (\(M\)) and comparisons (\(C\)) – are taken into account. In a real world implementation, several additional factors besides the mathematical operations should be included in the overall complexity of max-log-MAP, like memory reads and writes. Since these additional factors are architecture dependent, it is very hard (if possible) to determine a general complexity measure taking them into account, and therefore we omit their contribution in this paper.

Before we proceed, we define a notation for the number of operations of a given complexity measure. We use

\[m \cdot M + s \cdot S + c \cdot C\]

to denote \(m\) multiplications, \(s\) summations and \(c\) comparisons. Also, associativity applies. For instance,

\[n \cdot (M + S) = n \cdot M + n \cdot S,\]

which denotes \(n\) multiplications and \(n\) summations. Likewise, two or more complexity measures add up in a component-wise fashion, i.e., if

\[T_1 = m_1 \cdot M + s_1 \cdot S + c_1 \cdot C,\]

\[T_2 = m_2 \cdot M + s_2 \cdot S + c_2 \cdot C,\]
1. Calculation of $\alpha$’s, $\beta$’s and $\gamma$’s: We start our analysis with the number of operations required to calculate $\gamma$ in (6). Since $x_{i,u}$ is a BPSK symbol it can only assume two values. This way, $2 \cdot M$ are required for each of $a_i^{sec}$ code bits of a trellis section. Further, $(a_i^{sec} - 1) \cdot S$ are required to sum each $r_{i,u} \cdot x_{i,u}(l,l)$ term. The sum of extrinsic information (if there is any) requires another $S$. Each of the $2^{\nu_i^{sec}}$ states in section $i$ has $2^{\nu_i^{sec}}$ outgoing edges, so that the number of required operations per section is

$$T_{\gamma_i}^{sec} = \begin{cases} 
2a_i^{sec} \cdot M + 2^{\nu_i^{sec}+b_i^{sec}} \cdot a_i^{sec} \cdot S, & \text{if } b_i^{sec} > 0 \\
2a_i^{sec} \cdot M + 2^{\nu_i^{sec}+b_i^{sec}} \cdot (a_i^{sec} - 1) \cdot S, & \text{otherwise.}
\end{cases}$$

(9)

Notice that (9) is conditional because sections without information bits have no extrinsic information.

Next we calculate the required number of operations performed in the determination of $\alpha$ in (4) and $\beta$ in (5). The number of edges that reach a given state is $\Psi = 2^{\nu_i^{sec}+b_i^{sec}} \cdot C$. As for each of the $2^{\nu_i^{sec}+b_i^{sec}}$ $\alpha$’s to be calculated we need $\Psi \cdot S$ and $(\Psi - 1) \cdot C$, then the total number of required operations per section for calculating the $\alpha$ values is

$$T_{\alpha_i}^{sec} = 2^{\nu_i^{sec}+b_i^{sec}} \cdot C + \left(2^{\nu_i^{sec}+b_i^{sec}} - 2^{\nu_i^{sec}+1}\right) \cdot C.$$  

(10)

In the case of $\beta$, each of the $2^{\nu_i^{sec}}$ states have $2^{\nu_i^{sec}}$ emerging edges, demanding $2^{\nu_i^{sec}} \cdot S$ and $(2^{\nu_i^{sec}} - 1) \cdot C$, then

$$T_{\beta_i}^{sec} = 2^{\nu_i^{sec}+b_i^{sec}} \cdot S + \left(2^{\nu_i^{sec}+b_i^{sec}} - 2^{\nu_i^{sec}+1}\right) \cdot C.$$  

(11)

2. Calculation of the LLR’s: The last calculation step is related to (2), which we split in two steps. The first one is defined in (3). Note that we have $2^{\nu_i^{sec}}$ different metrics per section. Moreover, each of the $2^{\nu_i^{sec}}$ states contributes with one operand for the max operation in (3). Each operand requires two summations resulting in a complexity of

$$T_{\psi_i} = \left(2^{\nu_i^{sec}+1}\right) \cdot S + \left(2^{\nu_i^{sec}} - 1\right) \cdot C.$$  

(12)

Since $2^{\nu_i^{sec}}$ metrics are to be calculated, we have a total complexity in this step of

$$T_{\psi_i}^{total} = 2^{\nu_i^{sec}} \left[\left(2^{\nu_i^{sec}+1}\right) \cdot S + \left(2^{\nu_i^{sec}} - 1\right) \cdot C\right].$$  

(13)

The second step is the LLR calculation. In this step one label metric is taken as reference and the other label metrics are subtracted from it. Each subtraction generates the $\Lambda_i(q)$ for each $q$. Thus $(2^{\nu_i^{sec}} - 1) \cdot S$ are added to $T_{\psi_i}^{total}$, resulting in

$$T_{\Lambda_i}^{sec} = \begin{cases} 
2^{\nu_i^{sec}} \left[\left(2^{\nu_i^{sec}+1}\right) \cdot S + \left(2^{\nu_i^{sec}} - 1\right) \cdot C\right], & \text{if } b_i^{sec} > 0 \\
(2^{\nu_i^{sec}} - 1) \cdot S, & \text{otherwise.}
\end{cases}$$

(14)

Note that, once more, the complexity is conditioned on the presence/absence of information bits on that specific trellis section. Finally, the overall complexity over a sectionalized trellis module with $n'$ sections, defined as a function of the number of required mathematical operations, can be expressed as

$$T_{MAP}^{sec} = \sum_{i=0}^{n'-1} T_{\alpha_i}^{sec} + T_{\beta_i}^{sec} + T_{\Lambda_i}^{sec}. \quad (15)$$

IV. PERFORMANCE VERSUS COMPLEXITY EVALUATION

Lower complexity Viterbi decoding without any performance loss is possible using the minimal trellis representation [16], [18], however it is not clear if the same holds true for the case of turbo decoding using minimal trellis. Therefore, in this section we investigate the performance and complexity of turbo decoding when using the conventional, minimal and sectionalized trellises in terms of BER versus $E_b/N_0$, where $E_b$ is the energy per information bit and $N_0$ is the noise power spectral density. We consider two cases. A rate $r = 2/6$ turbo code, included in the WiMAX standard, with constituent codes of rate $r = 2/4$; and a rate $r = 3/7$ turbo code with constituent codes of rate $r = 3/5$. The coded bits were BPSK modulated prior to transmission over the AWGN channel. Each point shown in the figures was simulated until 400 bit errors were achieved. A total of 30 turbo decoding iterations were carried out. Moreover, we used the permutation symbol interleaver proposed by Douillard and Berrou in [12].

A. C(4, 2, 3) Constituent Code

First we analyze the case of the C(4, 2, 3) constituent code used in the WiMAX turbo code. The minimal trellis of this constituent code is shown in Figure 1. All the possible sectionalized of the minimal trellis in Figure 1 were considered and their decoding complexities, according to the definitions in Section III, are shown in Table I. Note that the sectionalization vectors in the first and the last rows of this table define the minimal and the conventional trellis, respectively.

The BER performance of turbo decoding using each of the possible trellis sectionalizations are shown in Figure 3, for the case of an interleaver of length $N = 480$ bits. Each vector $vet_{sec}$ is shown as the label of each curve. From the figure we can observe that the eight sectionalization schemes have been gathered in two performance groups, which differ on how the LLR of the information bits are calculated. Note that the BER performances of turbo decoding based on the minimal trellis and turbo decoding based on some sectionalizations (vetsec = $\{0, 0, 1\}$, $\{0, 1, 0\}$, and $\{0, 1, 1\}$) of the minimal trellis coincide. Moreover, the BER performance of turbo decoding using the conventional trellis coincides with those using some other sectionalizations (vetsec = $\{1, 0, 0\}$, $\{1, 0, 1\}$, and $\{1, 1, 0\}$) of the minimal trellis. When the sectionalization scheme joins the two information bits (vetsec$_1 = 1$), so that their LLR is jointly calculated, the performance is better. That is the case of the conventional trellis and of three different sectionalizations of the minimal trellis. Note that when the constituent code is decoded using the minimal trellis module, so that each information bit is decoded individually, the performance loss is of around 0.3

1Each symbol corresponds to $k$ information bits.
possible groups. In this section we consider the case of a rate $r = 2/3$ constituent code, in order to investigate whether the previous conclusions still hold when there are more grouping possibilities.

Figures 7 and 8 show the conventional and the minimal trellis module, respectively, of a $C(5, 3, 5)$ constituent code found by the authors. Figure 9 shows one possible sectionalization of the minimal trellis in Figure 8, with $\text{vetsect} = \{0, 0, 1, 0\}$, while the decoding complexity of all possible sectionalizations are shown in Table II. In the case of $k = 3$ there are four ways to group the systematic bits in the trellis module. Figure 10 shows that, as expected, there are four corresponding BER performance groups. Differently from the case of the $C(4, 2, 3)$ constituent code, here the less complex sectionalizations, shown in boldface in Table II – are not in the best performing group, so that some design trade-off must be accepted: the less complex sectionalizations (64% fewer summations, 60% fewer comparisons than the conventional trellis) imposes a
TABLE I  
DECODING COMPLEXITY OF THE C(4, 2, 3) CONSTITUENT CODE.

<table>
<thead>
<tr>
<th>vetsec</th>
<th>γ α β</th>
<th>Λ</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,0,0</td>
<td>48 8</td>
<td>96 24</td>
<td>98 44</td>
</tr>
<tr>
<td>0,0,1</td>
<td>80 8</td>
<td>80 24</td>
<td>98 44</td>
</tr>
<tr>
<td>0,1,0</td>
<td>80 8</td>
<td>64 24</td>
<td>64 24</td>
</tr>
<tr>
<td>0,1,1</td>
<td>112 8</td>
<td>48 24</td>
<td>48 24</td>
</tr>
<tr>
<td>1,0,0</td>
<td>64 8</td>
<td>80 24</td>
<td>80 24</td>
</tr>
<tr>
<td>1,0,1</td>
<td>96 8</td>
<td>64 24</td>
<td>64 24</td>
</tr>
<tr>
<td>1,1,0</td>
<td>96 8</td>
<td>48 24</td>
<td>48 24</td>
</tr>
<tr>
<td>1,1,1</td>
<td>128 8</td>
<td>32 24</td>
<td>32 24</td>
</tr>
</tbody>
</table>

Fig. 6. Bit error rate (BER) performance of turbo decoding as a function of $E_b/N_0$ for different trellis sectionalizations, using the C(4, 2, 4) constituent code and a 480 bits interleaver.

Fig. 7. Conventional trellis module of the C(5, 3, 5) code.

0.25 dB performance loss at a BER of $10^{-5}$ with respect to the best performing group. Note that in this case the complexity reduction is quite significant. If such performance degradation cannot be accepted, the least complex sectionalization within the best performing group – $vetsec = \{0, 1, 1\}$, highlighted in gray in Table II – is able to reduce summations by 8%.

Moreover, from Figure 10 we observe that performance improves as more systematic bits are grouped together in the trellis module. Note that the minimal trellis module, which has no systematic bits grouping, is in the worst performing group, with a loss of around 0.25 dB at a BER of $10^{-5}$ with respect to the conventional trellis module. This effect is observed because when decoding a convolutional code with $k > 1$ using its minimal trellis module, the LLR is calculated one bit at a time, in the same way as it is done when a convolutional code with $k = 1$ is employed. Therefore, when using the minimal trellis for turbo decoding we are not able to achieve the benefits of constituent codes with $k > 1$ as described in [12].

Alternatively, it is possible by using the minimal trellis for decoding to have the same performance as the one achieved with the conventional trellis. In this case, the LLRs must be calculated symbol-wise ($k$ bits at a time), therefore jointly processing the sections with information bits. For instance, in the case of the C(4, 2, 3) constituent code, a single $\gamma$ would be calculated for sections 1 and 2 of the minimal trellis module (thus, accounting for the transitions between depths 0 and 2 of Figure 1), so that the values of $\alpha$ and $\beta$ would be calculated only for depths 0, 2, and 3. Between depths 0 and 2 the values of $\gamma$ would be calculated considering the transitions generated by $k$-tuples; as a consequence the LLRs would also be related to $k$-tuples, not to single bits. Implementing the turbo decoding over the minimal trellis following the above steps achieves the same performance as over the conventional trellis. However, note that this implementation is actually the description of the sectionalized trellis with $vetsec = \{1, 0, 0\}$, so that sections 1 and 2 are grouped together, yielding a single section with $k = 2$ information bits.

V. DENSITY EVOLUTION ANALYSIS

As can be seen from the results in the previous section, when using the BCJR algorithm (or the max-log-MAP algorithm) for decoding turbo codes with a sectionalized trellis, the
TABLE II
DECODING COMPLEXITY OF THE C(5, 3, 5) CONSTITUENT CODE.

<table>
<thead>
<tr>
<th>vetsec</th>
<th>$\gamma$</th>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\Lambda$</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,0,0,0</td>
<td>160</td>
<td>10</td>
<td>224</td>
<td>80</td>
<td>224</td>
</tr>
<tr>
<td>0,0,1,0</td>
<td>256</td>
<td>10</td>
<td>224</td>
<td>96</td>
<td>224</td>
</tr>
<tr>
<td>0,0,1,1</td>
<td>448</td>
<td>10</td>
<td>224</td>
<td>128</td>
<td>224</td>
</tr>
<tr>
<td>0,1,0,0</td>
<td>256</td>
<td>10</td>
<td>224</td>
<td>128</td>
<td>224</td>
</tr>
<tr>
<td>0,1,0,1</td>
<td>256</td>
<td>10</td>
<td>224</td>
<td>128</td>
<td>224</td>
</tr>
<tr>
<td>0,1,1,0</td>
<td>288</td>
<td>10</td>
<td>288</td>
<td>224</td>
<td>224</td>
</tr>
<tr>
<td>1,0,0,0</td>
<td>224</td>
<td>10</td>
<td>192</td>
<td>80</td>
<td>192</td>
</tr>
<tr>
<td>1,0,0,1</td>
<td>256</td>
<td>10</td>
<td>192</td>
<td>96</td>
<td>192</td>
</tr>
<tr>
<td>1,0,1,0</td>
<td>320</td>
<td>10</td>
<td>192</td>
<td>128</td>
<td>192</td>
</tr>
<tr>
<td>1,0,1,1</td>
<td>512</td>
<td>10</td>
<td>192</td>
<td>128</td>
<td>192</td>
</tr>
<tr>
<td>1,1,0,0</td>
<td>288</td>
<td>10</td>
<td>160</td>
<td>80</td>
<td>160</td>
</tr>
<tr>
<td>1,1,0,1</td>
<td>320</td>
<td>10</td>
<td>160</td>
<td>96</td>
<td>160</td>
</tr>
<tr>
<td>1,1,1,0</td>
<td>448</td>
<td>10</td>
<td>224</td>
<td>128</td>
<td>224</td>
</tr>
<tr>
<td>1,1,1,1</td>
<td>1024</td>
<td>10</td>
<td>224</td>
<td>128</td>
<td>224</td>
</tr>
</tbody>
</table>

Fig. 8. Minimal trellis module of the C(5, 3, 5) code. The solid edges represent “0” codeword bits and the dashed edges represent “1” codeword bits.

Fig. 9. Sectionalized trellis module of the C(5, 3, 5) code, vetsec = {0, 1, 0, 1}.

Fig. 10. Bit error rate (BER) performance of turbo decoding as a function of $E_b/N_0$ for different trellis sectionalizations, using the C(5, 3, 5) constituent code and a 720 bits interleaver.

likelihoods of the information bits are computed jointly, and this causes a performance improvement due to the fact that the code structure is better exploited. Also, sectionalization enforces the decisions to be taken symbol-wise, which speeds up the convergence of the iterative algorithm. As the turbo decoding algorithm performance improves with the number of iterations but saturates for a large number of iterations, speeding up the convergence should also help in the waterfall effect of the turbo decoding.

In order to have a more formal view on this topic, we have made a density evolution analysis of turbo decoding using the conventional, sectionalized and minimal trellises for the case of the WiMax turbo code (with constituent code C(4, 2, 3)). The density evolution analysis is based on [22], and it tracks the evolution of the extrinsic information transfer between the two decoders of the turbo decoder. In particular, we consider the so-called half iteration model, so that the plots show the output SNR (the quality) of the extrinsic information of decoder 1 versus its input SNR, and the input SNR of decoder 2 versus its output SNR. Note that the input SNR of decoder 1 is the output SNR of decoder 2, while the input SNR of decoder 2 is the output SNR of decoder 1.

The results are illustrated in Figure 11, which shows the density evolution of the turbo decoders based on the conven-
tional, sectionalized and minimal trellises for different $E_b/N_0$ values. We always consider the transmission of an information message containing only 1’s, so that in the case of the minimal trellis we present the evolution of the extrinsic information considering the calculation of the LLR between bits 1 and 0, and in the case of the conventional and the sectionalized trellis we show the evolution of the extrinsic information considering the calculation of the LLR between tuples 11 and 00. Several points can be inferred from the analysis of the four subfigures:

- The density evolution analysis for the conventional and the sectionalized trellis (which groups the information bits in a single section) yields exactly the same results;
- The decoding tunnel associated with the conventional/sectionalized trellis is slightly wider than that associated with the minimal trellis in the case of the lowest $E_b/N_0$ value (Figure 11-a), showing that the waterfall behavior of the turbo decoder tends to start earlier in the case of the conventional/sectionalized trellis than in the case of the minimal trellis;
- In the case of the first three figures (Figures 11-a to c) the extrinsic values are larger for the conventional/sectionalized trellis than in the minimal trellis. This indicates that the convergence speed is superior in the case of the conventional/sectionalized trellis than in the case of the minimal trellis;
- For a larger $E_b/N_0$ value (Figure 11-d) the evolution of extrinsic information transfer for the two decoders (based on the conventional/sectionalized or the minimal trellis) gets more similar, showing that the decoders tend to achieve the same performance at a sufficiently large $E_b/N_0$ value.

VI. Conclusions

In this paper, we have investigated the turbo decoding performance-complexity trade-off when using the conventional, minimal and sectionalized trellises. The complexity has been defined as the number of multiplications, summations and comparisons required by the max-log-MAP algorithm in each trellis module of a constituent code, while the BER has been considered as the performance metric. Our results show that the performance is dependent on the trellis module. The best performance is achieved when the systematic bits are grouped in one section of the trellis module, a characteristic inherent of the conventional trellis module and of some of the possible sectionalizations of the minimal trellis.

The decoding complexity is also a function of the trellis module sectionalization, however there is no clear evidence of a design rule. In some of our examples, the least complex module is also the best performing one, while in some other cases the least complex module is in the worst performing group. Therefore, we conclude that each trellis has a particular complexity behavior with respect to its sectionalization. Each possible trellis sectionalization must be analyzed beforehand, and the best one should be chosen depending on the design goal.

REFERENCES

[9] 3GPP TS 36.212 V8.7.0, "3rd generation partnership project; technical specification group radio access network; evolved universal terrestrial radio access (E-UTRA); multiplexing and channel coding (Release 8)," 2009.
Fig. 11. Density evolution analysis for the turbo decoder of the WiMax turbo code (constituent codes with $k/n = 2/4$): (a) $E_b/N_0 = 0.333$ dB; (b) $E_b/N_0 = 0.666$ dB; (c) $E_b/N_0 = 1.0$ dB; and, (d) $E_b/N_0 = 2.0$ dB.