

Reduced complexity decoding of convolutional codes based on the M-algorithm and the minimal trellis

Richard Demo Souza · Cecilio Pimentel ·
Daiana Nascimento Muniz

Received: 13 October 2011 / Accepted: 29 February 2012 / Published online: 13 March 2012
© Institut Télécom and Springer-Verlag 2012

Abstract In this paper, we propose three new sub-optimum, reduced complexity decoding algorithms for convolutional codes. The algorithms are based on the minimal trellis representation for the convolutional code and on the M-algorithm. Since the minimal trellis has a periodically time-varying state profile, each algorithm has a different strategy to select the number of surviving states in each trellis depth. We analyse both the computational complexity, in terms of arithmetic operations, and the bit error rate performance of the proposed algorithms over the additive white Gaussian noise channel. Results demonstrate that considerable complexity reductions can be obtained at the cost of a small loss in the performance, as compared to the Viterbi decoder.

Keywords Convolutional codes · Minimum trellis · M-algorithm · Viterbi decoding

1 Introduction

Digital communication systems make use of error correcting codes to reliably transmit information over a noisy channel [1]. Convolutional codes are among the

most used ones, with extensive applications in modern wireless communication standards [2–5]. However, the decoding of convolutional codes demands a large amount of processing and energy consumption of a regular wireless digital receiver, which is usually battery powered. For instance, in [6], it is shown that the decoding of a convolutional code, executed by the Viterbi algorithm (VA) [1], accounts for up to 76% of the processing required by a HYPERLAN/2 receiver [4]. In [7], the authors analyse different receiver implementations compatible to the IEEE 802.11 standard [5], showing that the VA accounts for 35% of the overall power consumption.

Moreover, in the last two decades, we had an unimpressive improvement of only 3.5% per year in the nominal battery capacity, while the throughput has increased a million times [8, 9]. Thus, complexity (power) reductions in the decoding of convolutional codes would increase the energy efficiency and lifetime of many wireless devices, being of considerable impact in a number of applications. A survey in the literature shows a series of papers related to this issue; among them, we can cite [10–25]. These works can be divided into three groups: (a) hardware-specific implementations [10–13], (b) sub-optimum decoding methods [14–19] and (c) simpler trellis representations [20–25]. Supported by two recent new paradigms for the implementation of digital communications systems, the software-defined radio model [26] and the cognitive radio concept [27], we concentrate on software-oriented implementations. Therefore, we consider only two of the above options: sub-optimum decoding methods and simpler trellis representations.

Most sub-optimum decoding algorithms were proposed based on the conventional trellis representation

R. D. Souza (✉) · D. N. Muniz
Federal University of Technology—Paraná (UTFPR),
Av. Sete de Setembro 3165, Curitiba, Paraná, Brazil
e-mail: richard@utfpr.edu.br

D. N. Muniz
e-mail: dai_muniz@yahoo.com.br

C. Pimentel
Federal University of Pernambuco (UFPE),
C.P. 7800, Recife, Pernambuco, Brazil
e-mail: cecilio@ufpe.br

of a convolutional code and are similar to the VA. The sub-optimality comes from pruning some of the trellis edges, based on specific criteria. One of the most famous sub-optimum decoding algorithms is the M-algorithm (MA) [14, 15]. The MA achieves a bit error ratio (BER) close to that of the VA with reduced complexity (for an appropriate choice of M) [19]. The reduction in complexity is obtained by expanding only M surviving states per trellis section, instead of expanding all trellis states at each section. Due to this characteristic, the MA is also used when the convolutional code has a large overall constraint length [19] and the application of the optimum VA is prohibitive.

Minimal trellis representation of a convolutional code, introduced in [20] by McEliece and Lin, is an alternative trellis representation for the code, allowing to minimise several theoretical quantities based on the trellis complexity measures defined in [28]. Such a trellis, as opposed to the conventional trellises, has an irregular structure (the number of states and the number of edges emanating from each state are periodically time varying). New minimal trellis codes, with good distance spectrum and fixed complexity of the minimal trellis, have been proposed recently [21, 22]; furthermore, in [29], the authors have shown that the decoding of convolutional codes using the minimal trellis can result in practical implementations with both reduced power consumption and hardware complexity.

In this paper, we combine both approaches, sub-optimum decoding algorithms and simpler trellis representations. More specifically, we design three decoding algorithms based on the MA operating over the minimal trellis, namely modular M-algorithm, proportional M-algorithm and fixed M-algorithm (see Section 3). Each algorithm relies on a different strategy to select the number of states with best metrics in each depth of the trellis. This number can be either fixed or variable when the MA operates over the minimal trellis which establishes distinct trade-off between complexity and performance. We analyse both the computational complexity, in terms of arithmetic operations, and the BER performance for each proposed algorithm over the additive white Gaussian noise (AWGN) channel. Our results show that considerable reductions in complexity can be obtained while achieving a performance close to that of the VA within a given tolerance. Moreover, we show that the proposed algorithms offer a wide range of performance-complexity trade-off which can increase even more the savings in computational complexity. Finally, based on the complexity results, we discuss the applicability of each algorithm in terms of receiver architecture.

The rest of this paper is organised as follows: Some fundamental concepts are introduced in Section 2. The new proposed algorithms are described in Section 3. The BER performance of the new algorithms is numerically investigated in Section 4, while a complexity analysis is carried out in Section 5. Section 6 concludes the paper.

2 Fundamental concepts

Consider a convolutional code $C(n, k, v)$, where v , k and n are the overall constraint length, the number of binary inputs and binary outputs, respectively. The code rate is $R = k/n$. Every such convolutional code 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*. In general, a trellis module Φ for a convolutional code C consists of n' trellis sections, 2^{v_t} states at depth t , 2^{b_t} edges emanating from each state at depth t and l_t bits labelling each edge from depth t to depth $t + 1$, for $0 \leq t \leq n' - 1$.

Two important complexity measures of the computational effort per decoded bit of the VA operating over a specific trellis module are the trellis complexity and the number of merges, which are the number of real additions and real comparisons (normalised by the number of information bits), respectively; the VA has to perform to update the state metrics in each module. The trellis complexity of the module Φ for the code C , denoted by $TC(\Phi)$, is [20]

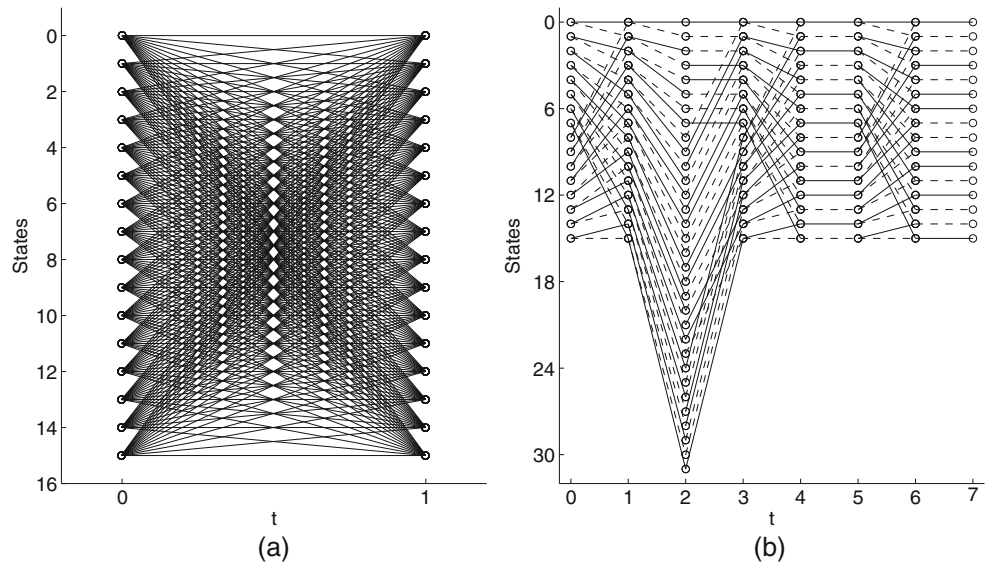
$$TC(\Phi) = \frac{1}{k} \sum_{t=0}^{n'-1} l_t 2^{v_t+b_t} \quad (1)$$

symbols per bit. The number of comparisons required at a specific state, in its turn, is the number of edges reaching it minus 1 [30]. Then, the number of comparisons in the module Φ at depth $t + 1$ is $2^{v_t+b_t} - 2^{v_{t+1}}$. So, the merge complexity of the module Φ , denoted by $MC(\Phi)$, is

$$MC(\Phi) = \frac{1}{k} \sum_{t=0}^{n'-1} 2^{v_t+b_t} - 2^{v_{t+1}}. \quad (2)$$

In particular, the conventional trellis module Φ_{conv} for a rate $R = k/n$ convolutional code C consists of one trellis section with 2^v initial states and 2^v final states; each initial state is connected by 2^k directed edges to final states, and each edge is labelled with n bits. The trellis complexity and the merge complexity of the conventional trellis are $TC(\Phi_{\text{conv}}) = (n/k) 2^{v+k}$ and

Fig. 1 Conventional **a** and minimal **b** trellis modules for the $C(7, 4, 4)$ convolutional code with generator matrix in Eq. 3. In the minimal trellis, *solid edges* represent “0” codeword bits while *dashed edges* represent “1” codeword bits



$MC(\Phi_{conv}) = 2^v(2^k - 1)/k$. For instance, consider the $C(7, 4, 4)$ code with the following generator matrix:

$$G(D) = \begin{pmatrix} 1 + D & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 + D & 1 + D & D & 0 & 0 & 1 \\ D & D & D & 1 & 1 & 0 & 1 \\ 0 & D & 0 & D & D & 1 + D & 1 \end{pmatrix} \quad (3)$$

whose conventional trellis module Φ_{conv} is shown in Fig. 1a, with $TC(\Phi_{conv}) = 448$ and $MC(\Phi_{conv}) = 60$. The limiting factor for using the VA, when the trellis module is dense (with many states and edges per state), is the implementation complexity growth.

2.1 Minimal trellis

The “minimal” trellis module, Φ_{min} , for convolutional codes was developed in [20]. This “minimal” structure has $n' = n$ sections and $l_t = 1$ bit per edge $\forall t$. The state complexity and the edge complexity at depth t will be denoted by \tilde{v}_t and \tilde{b}_t , respectively. The state and the edge complexity profiles of the “minimal” trellis module are denoted by $\tilde{\mathbf{v}} = (\tilde{v}_0, \dots, \tilde{v}_{n-1})$ and $\tilde{\mathbf{b}} = (\tilde{b}_0, \dots, \tilde{b}_{n-1})$, respectively. This module presents an irregular pattern in each section. For instance, Fig. 1b shows the minimal trellis module for the $C(7, 4, 4)$ code with generator matrix in Eq. 3. While the single-section conventional trellis module in Fig. 1a has a very regular structure, 16 states with 16 edges leaving each state, each edge labelled by 7 bits, the minimal trellis module in Fig. 1b has $n = 7$ sections, with 16 or 32 states each. Note that only the first, second, fourth and sixth sections have information bits, i.e. two edges leave each state ($\tilde{b}_t = 1$).

For the $C(7, 4, 4)$ code, $\tilde{\mathbf{v}} = (4, 4, 5, 4, 4, 4, 4)$ and $\tilde{\mathbf{b}} = (1, 1, 0, 1, 0, 1, 0)$; thus, it readily follows from Eqs. 1 and 2 that $TC(\Phi_{min}) = 48$ and $MC(\Phi_{min}) = 16$. It is important to say that the minimal trellis module achieves [28, Theorem 4.26] not only the minimum trellis complexity and the minimum merge complexity but other complexity measures as well, such as maximum number of states and total number of states. Such reduced complexity is our motivation for considering the use of the minimal trellis in this work. However, it is not completely clear whether such a trellis complexity measure directly translates into decoding complexity of reduced complexity decoding algorithms, such as the MA. For that sake, we will compute in Section 5 the number of arithmetic operations required by a specific decoding algorithm, be it over the conventional or the minimal trellis.

2.2 MA over the conventional trellis module

Further reduction of decoding complexity can be attempted by using, besides the simpler trellis representation, sub-optimum algorithms such as the MA [14]. The MA is very similar to the VA, the main difference resting in the fact that the MA calculates the accumulative metric of at most $M \leq 2^v$ paths along the conventional trellis module.¹ The MA operating over the

¹When running the MA, it is possible that the number of states in section t that are reached by edges coming from section $t - 1$ be smaller than M . In this case, the MA stores all surviving states. In Section 5, we illustrate one of such cases.

conventional trellis can be described by the following steps:

1. Start from the leftmost module of the trellis.
2. Expand all the states metrics stored at depth $t - 1$ to depth t . Select the surviving edges for each state reached at time t .
3. Select, at most, the M states at time t with the best metrics and discard the others.
4. Store the selected states, their metrics and surviving edges.
5. Repeat steps 2–4 for all trellis modules.
6. Estimate the transmitted sequence by tracebacking from the state with the best final metric.

The main difference between the VA and MA is that the MA selects the M best states before expanding the metrics of these states.

3 MA over the minimal trellis module

When the MA operates over the conventional trellis, the parameter M is fixed; however, since the minimal trellis has a time-varying state profile, the maximum number of states selected in each section of this trellis can be either fixed or variable. We propose three different variants of the MA to operate over the minimal trellis, namely modular M-algorithm (MMA), proportional M-algorithm (PMA) and fixed M-algorithm (FMA). These algorithms differ only in step 3, that is, in the way they select the best states.

The MMA selects the surviving states only in the end of each minimal trellis module. Thus, in the intermediate sections, all expansions are retained. The metrics are expanded up to the end of the module, storing the metrics of all states reached by any surviving edge. Then step 3 of the MMA is thus described as:

3. If this is the last section of the module, store at most the M best states and discard the others; otherwise, store the metrics of all states reached by a surviving edge.

In the PMA, the maximum number of stored states varies according to the number of states in that section. The maximum number of states stored per section is $2^{\tilde{v}_t} \times \frac{M}{2^v}$. Recall that $2^{\tilde{v}_t}$ is the number of states at that section of the minimal trellis, while 2^v is the number of states in the conventional trellis. For instance, if in the MA operating over the conventional trellis the parameter M means a reduction of 50% in the maximum number of stored states, then in the PMA this value of M means a reduction of 50% in the maximum number of states stored in each section of the minimal

trellis. Thus, the maximum number of stored states is proportional to the number of states in that section of the minimal trellis. Step 3 of the PMA can be written as:

3. Store at most the $2^{\tilde{v}_t} \times \frac{M}{2^v}$ states with the best metric for each section. Discard the others.

In the FMA, up to M states are stored at each section of the minimal trellis, independent of the trellis pattern or section. Step 3 of the FMA can be described as:

3. Store at most the M best states at each section. Discard the others.

4 BER performance

In this section, we numerically investigate the BER performance of the proposed algorithms. In the simulations, the coded blocks were binary phase shift keying (BPSK) modulated, sent over the AWGN channel and decoded by soft decision decoding. We considered the $C(7, 4, 4)$ code with generator matrix in Eq. 3 and the $C(5, 3, 4)$ code with generator matrix:

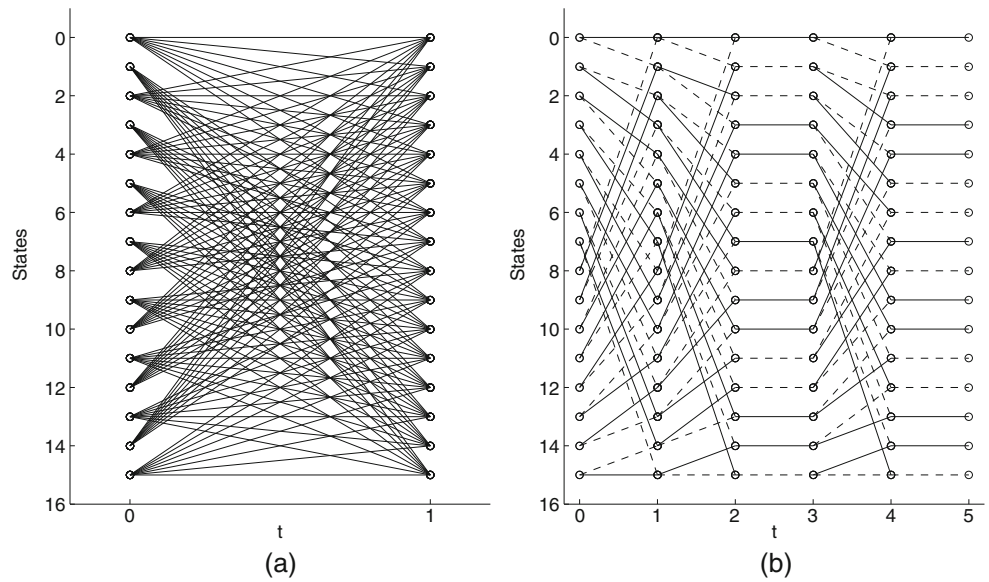
$$G(D) = \begin{pmatrix} 1 + D & D & 1 & 1 & 0 \\ 0 & 1 & 1 + D & 1 + D & 1 \\ D + D^2 & D & 0 & 1 + D & 1 + D \end{pmatrix}, \quad (4)$$

whose conventional and minimal trellises are shown in Fig. 2.

Figure 3 shows the BER performance of the three new algorithms based on the minimal trellis, as a function of parameter M for a fixed value of the signal-to-noise ratio E_b/N_0 (so that the BER is close to 10^{-6}), for the $C(7, 4, 4)$ code. In the figure, it is also shown the performance of the VA over the conventional trellis (or the minimal trellis), which is not a function of M . Note that the performance of the proposed algorithms increase with M , up to the limit where it reaches the performance of the VA.² Figure 4 shows similar results but for the $C(5, 3, 4)$ code. The performance curves show sometimes a rough behaviour due to the particular dynamics of the proposed algorithms. The increase of only 1 in M may have a huge impact sometimes and almost not any impact in some other cases, also

²As the minimum trellis and the conventional trellis represent the same code and the MMA algorithm only selects the surviving states at the end of the trellis module, the performances of the MMA algorithm operating over the minimum trellis and of the typical M-algorithm operating over the conventional trellis are identical.

Fig. 2 Conventional **a** and minimal **b** trellis modules for the $C(5, 3, 4)$ convolutional code with generator matrix in Eq. 4. In the minimal trellis, *solid edges* represent “0” codeword bits while *dashed edges* represent “1” codeword bits



depending on the algorithm. We noted this same behaviour when examining the performance, obtained by simulating the proposed algorithms over other codes.

In order to determine the minimal value of M required by each algorithm to achieve a performance close to the VA, we define the tolerance γ , which is the maximum acceptable BER degradation, so that the BER achieved by the sub-optimum decoding algorithm is at most γ above the BER obtained by the VA. For instance, considering the $C(7, 4, 4)$ code, $E_b/N_0 = 6$ dB, and supposing a tolerance of $\gamma = 10^{-6}$, the MMA requires $M = 6$, while the PMA $M = 9$ and the FMA $M = 10$. Thus, the MMA requires a smaller number of surviving states than the PMA and the FMA. But recall

that MMA only selects the surviving states at the end of the module, while the PMA and the FMA select them at each section. If the tolerance is increased to $\gamma = 5 \times 10^{-6}$, then there is a reduction in the required values of M for the MMA from $M = 6$ to $M = 5$ and for the FMA from $M = 10$ to $M = 9$. Increasing the tolerance even more, to $\gamma = 10^{-5}$, decreases the required values of M by 1 for all algorithms. Different results are obtained if we consider the $C(5, 3, 4)$ code at an $E_b/N_0 = 6$ dB. In this case and supposing $\gamma = 10^{-6}$, the MMA needs $M = 6$ while the PMA and FMA require $M = 10$ surviving states. The PMA and the FMA have exactly the same performance because the minimal trellis

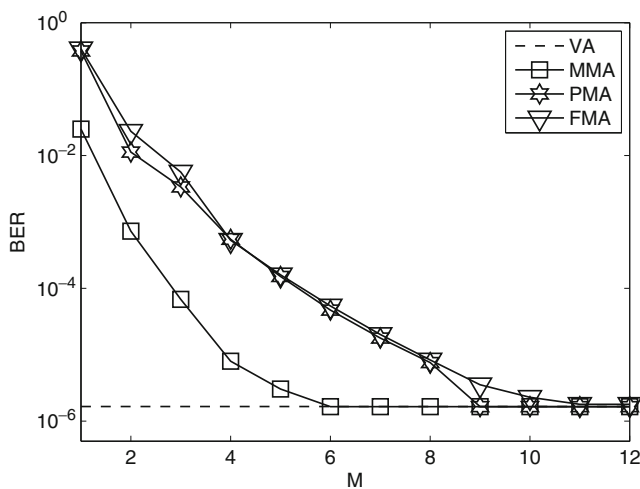


Fig. 3 BER versus M for the proposed algorithms, considering the $C(7, 4, 4)$ code, soft decision decoding and $E_b/N_0 = 6$ dB

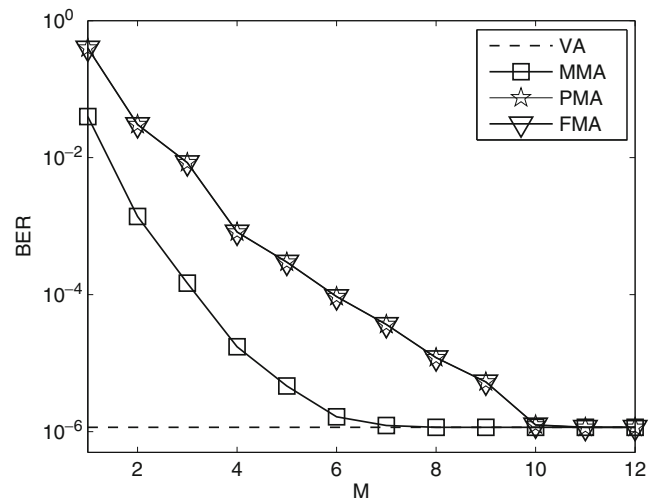


Fig. 4 BER versus M for the proposed algorithms, considering the $C(5, 3, 4)$ code, soft decision decoding and $E_b/N_0 = 6$ dB

Table 1 Required value of M for each code for different BER tolerances γ at $E_b/N_0 = 6$ dB

	Algorithm	$\gamma = 10^{-6}$	$\gamma = 5 \times 10^{-6}$	$\gamma = 10^{-5}$
C(7, 4, 4)	MMA	6	5	4
	PMA	9	9	8
	FMA	10	9	8
C(5, 3, 4)	MMA	6	5	5
	PMA	10	9	9
	FMA	10	9	9

module for this code has a constant number of states per section, namely 16 states per section. If the tolerance is increased to $\gamma = 5 \times 10^{-6}$, then the required M is decreased by 1 for all algorithms. By further increasing the tolerance to $\gamma = 10^{-5}$, there is no change in the required values of M . These results are summarised in Table 1. We observe that different M are required by each algorithm to achieve a performance close to the VA, within a given tolerance. A comparison among the complexity of each algorithm, for each particular required value of M , is performed in the next section.

5 Complexity analysis

The complexity of a decoding algorithm can be determined in many ways. For instance, in [31], the authors characterise the hardware cost of each operation related to the Viterbi algorithm. Here, we analyse the complexity only as a function of the number of arithmetic operations required by the algorithm, as to be independent of the specific implementation technology or architecture. In this paper, we consider only summations (S), multiplications (\mathcal{X}) and comparisons (C). First we consider only the operations required to calculate the accumulated state metrics. Later we take into account the effort required to select the best M states in each section, when needed by the MA and its variants. For the sake of simplicity, memory reads and writes are not taken into account.

Either in the VA, in the MA, or in the proposed algorithms, the first step in decoding is to calculate the edge metrics. These metrics may be calculated by means of the Hamming distance (hard decision decoding) or the Euclidean distance (soft decision decoding). We focus on the case of the Euclidean distance, where when using a constant modulus modulation such as BPSK it is known that the edge metric can be greatly simplified [32]. Supposing the use of a trellis module Φ with parameters v_t , b_t and l_t in section t , as defined in Section 2, then the edge metric can be calculated as $\sum_{j=1}^{l_t} y_t^j \cdot x_t^j$, where y_t^j is the j -th symbol of the re-

ceived l_t -tuple and x_t^j is the corresponding edge symbol. Therefore, a total of $l_t \mathcal{X}$ and $(l_t - 1) S$ are required per edge. Then, the state metrics are expanded, by summing the previous state metrics with the calculated edge metrics. Therefore, one additional summation per edge is required. Furthermore, in order to select the best new state metrics, we have to compare the accumulated metrics of all edges reaching a given state. If N_e^{it} is the number of edges starting in section t and reaching state i in section $t + 1$, then the total number of operations Λ^t in the section t of the trellis module Φ , required to calculate the accumulated state metrics, is:

$$\Lambda^t = \sum_{i=1}^{N_s^{t+1}} l_t N_e^{it} (\mathcal{X} + S) + (N_e^{it} - 1) C \tag{5}$$

where N_s^{t+1} is the number of states in section $t + 1$ that are reached by edges coming from section t . This analysis is carried out over the n' sections. Thus, the number of operations Λ in the module is

$$\Lambda = \sum_{t=0}^{n'-1} \sum_{i=1}^{N_s^{t+1}} l_t N_e^{it} (\mathcal{X} + S) + (N_e^{it} - 1) C. \tag{6}$$

The parameters N_e^{it} and N_s^{t+1} are a function of the state and edge complexity profiles of the trellis module and of the type of algorithm in use. For the VA, $N_e^{it} = \frac{2^{(b_t+v_t)}}{2^{v_{t+1}}}$ and $N_s^{t+1} = 2^{v_{t+1}}$. Note that the number of required summations and multiplications matches the non-normalised (multiplied by k) $TC(\Phi)$ defined in Eq. 1, while the number of required comparisons matches $MC(\Phi)$ in Eq. 2 after the same de-normalisation by k . Equation 6 is specialised for the minimal trellis ($n' = n, l_t = 1 \forall t$) as

$$\Lambda_{\min} = \sum_{t=0}^{n-1} \sum_{i=1}^{N_s^{t+1}} N_e^{it} (\mathcal{X} + S) + (N_e^{it} - 1) C. \tag{7}$$

In the case of the sub-optimum algorithms, as the MA and its variants operating over the minimal trellis proposed in this paper, it is hard to calculate exactly the number of required operations. That is because the number of edges in each section is a random variable, as well as the number of states reached at each trellis section. These random variables can be tracked during the computer simulations, so that the actual average values can be used for calculating the computational complexities. For instance, Table 2 lists \bar{N}_s^{t+1} , the average number of states in section $t + 1$, that are reached by edges coming from section t , for the three proposed algorithms, the C(7, 4, 4) code, $E_b/N_0 = 6$ dB, and using the minimal M for $\gamma = 10^{-6}$ ($M = 6$ for MMA, $M = 9$ for PMA and $M = 10$ for FMA).

Table 2 \bar{N}_s^{t+1} for $C(7, 4, 4)$, $E_b/N_0 = 6$ dB and $\gamma = 10^{-6}$

Algorithm	Section t						
	0	1	2	3	4	5	6
MMA ($M = 6$)	10.58	21.16	14.29	15.89	15.89	16.00	16.00
PMA ($M = 9$)	13.76	17.81	12.88	15.17	8.99	14.88	9.00
FMA ($M = 10$)	14.24	19.79	9.09	14.99	9.97	15.52	10.00

Supposing the use of the minimal trellis, the average number of edges starting in section t and reaching state i in section $t + 1$, \bar{N}_e^{it} , can be written as a function of \bar{N}_s^t , M , \tilde{b}_t and \tilde{v}_t , such that for the MMA, we have:³

$$\bar{N}_e^{it} = \begin{cases} \frac{2^{\tilde{b}_t} \bar{N}_s^t}{\bar{N}_s^{t+1}}, & \text{if } 0 < t \leq (n - 1) \\ \frac{2^{\tilde{b}_t} \min \{M, \bar{N}_s^t\}}{\bar{N}_s^{t+1}}, & \text{if } t = 0 \end{cases} \quad (8)$$

for the PMA

$$\bar{N}_e^{it} = \frac{2^{\tilde{b}_t} \min \{2^{\tilde{v}_t - v} M, \bar{N}_s^t\}}{\bar{N}_s^{t+1}} \quad (9)$$

while for the FMA

$$\bar{N}_e^{it} = \frac{2^{\tilde{b}_t} \min \{M, \bar{N}_s^t\}}{\bar{N}_s^{t+1}}. \quad (10)$$

The minimal operator in expressions (9) and (10) appears because, during the operation of the FMA or the PMA, \bar{N}_s^t can be smaller than the maximum number of surviving states allowed by each algorithm (this is illustrated in Table 2 for the case of the FMA in section $t = 2$, for instance). Then, we equate from Eqs. 7–10 the average arithmetic complexity of each proposed algorithm over the minimal trellis as

$$\Lambda_{\min}^{\text{FMA}} = \sum_{t=0}^{n-1} 2^{\tilde{b}_t} \min \{M, \bar{N}_s^t\} (\mathcal{X} + \mathcal{S}) + \left(2^{\tilde{b}_t} \min \{M, \bar{N}_s^t\} - \bar{N}_s^{t+1}\right) \mathcal{C} \quad (11)$$

$$\Lambda_{\min}^{\text{PMA}} = \sum_{t=0}^{n-1} 2^{\tilde{b}_t} \min \{2^{\tilde{v}_t - v} M, \bar{N}_s^t\} (\mathcal{X} + \mathcal{S}) + \left(2^{\tilde{b}_t} \min \{2^{\tilde{v}_t - v} M, \bar{N}_s^t\} - \bar{N}_s^{t+1}\right) \mathcal{C} \quad (12)$$

$$\Lambda_{\min}^{\text{MMA}} = \sum_{t=0}^{n-1} 2^{\tilde{b}_t} \bar{N}_s^t (\mathcal{X} + \mathcal{S}) + \left(2^{\tilde{b}_t} \bar{N}_s^t - \bar{N}_s^{t+1}\right) \mathcal{C}. \quad (13)$$

Moreover, since the number of states reached at any trellis section varies, the effort to select the best states

(when required) also varies. From Table 2, we see that when running the FMA with $\gamma = 10^{-6}$, for instance, in the first section the best 10 states have to be selected, on average, out of around 14 ($\bar{N}_s^1 = 14.24$ in Table 2), in the second section 10 states have to be selected out of about 20 ($\bar{N}_s^2 = 19.79$) and so on. Such an effort, in terms of comparisons, of selecting the M largest elements within a vector of \bar{N}_s^{t+1} elements can be approximated by [33]:⁴

$$\Pi_t(M, \bar{N}_s^{t+1}) = \bar{N}_s^{t+1} - M + \sum_{i=\bar{N}_s^{t+1}-M+1}^{\bar{N}_s^{t+1}} \log_2 i. \quad (14)$$

Therefore, in order to fairly compare the proposed algorithms, we have to take into account the effort required to select the best states, an action carried out at each section in the PMA and FMA and only at the end of the module in the MMA. Then, the complexity of the proposed algorithms over the minimal trellis can be written as

$$\Lambda_{\min}^{\text{total}} = \Lambda_{\min} + \left(\sum_{t=0}^{n-1} \Pi_t\right) \mathcal{C} \quad (15)$$

where Λ_{\min} is calculated according to Eqs. 11–13 and Π_t is the average number of comparisons required at section t to select the best states, as defined in Eq. 14, and as function of M , \bar{N}_s^{t+1} and to the particular operation of the algorithm.

Based on Eq. 15, we can present the average number of arithmetic operations required by each of the proposed algorithms, considering the minimal value of M given in Table 1 for a given tolerance with respect to the performance of the VA. The values of \bar{N}_s^t , $t = 0, \dots, n$, used in Eqs. 11–15 are obtained by simulations. The case of the $C(7, 4, 4)$ code is shown in Table 3, while Table 4 deals with the case of the $C(5, 3, 4)$ code. The number of operations required by the VA, over the conventional (VA_c) and minimal trellises (VA_m), are also shown.

From Table 3, we conclude that all algorithms based on the minimal trellis required much less operations

³Note that, based on the trellis module definition, $\bar{N}_s^0 = \bar{N}_s^n$.

⁴Note that, alternatively, one could determine the $\bar{N}_s^{t+1} - M$ smallest elements within a vector of \bar{N}_s^{t+1} elements. If that is simpler (less comparisons required), then one should select the survivors that way. In our numerical results, we always consider the case that requires less comparisons.

Table 3 Arithmetic operations required for decoding $C(7, 4, 4)$ at $E_b/N_0 = 6$ dB

Algorithm	$\gamma = 10^{-6}$	$\gamma = 5 \times 10^{-6}$	$\gamma = 10^{-5}$
VA _c	$1,792S + 1,792\mathcal{X} + 240C$		
VA _m	$192S + 192\mathcal{X} + 64C$		
MMA	$146.44S + 146.44\mathcal{X} + 65.63C$	$136.28S + 136.28\mathcal{X} + 58.44C$	$124.85S + 124.85\mathcal{X} + 50.65C$
PMA	$107.79S + 107.79\mathcal{X} + 152.90C$	$107.79S + 107.79\mathcal{X} + 152.90C$	$95.81S + 95.81\mathcal{X} + 136.41C$
FMA	$109.76S + 109.76\mathcal{X} + 135.56C$	$98.79S + 98.79\mathcal{X} + 124.67C$	$87.81S + 87.81\mathcal{X} + 110.54C$

Table 4 Arithmetic operations required for decoding $C(5, 3, 4)$ at $E_b/N_0 = 6$ dB

Algorithm	$\gamma = 10^{-6}$	$\gamma = 5 \times 10^{-6}$	$\gamma = 10^{-5}$
VA _c	$640S + 640\mathcal{X} + 112C$		
VA _m	$128S + 128\mathcal{X} + 48C$		
MMA	$96.18S + 96.18\mathcal{X} + 51.48C$	$88.55S + 88.55\mathcal{X} + 44.50C$	$88.55S + 88.55\mathcal{X} + 44.50C$
PMA/FMA	$79.82S + 79.82\mathcal{X} + 90.64C$	$71.84S + 71.84\mathcal{X} + 82.46C$	$71.84S + 71.84\mathcal{X} + 82.46C$

than VA_c. Moreover, the MMA algorithm requires less S , \mathcal{X} and C than VA_m for the tolerances $\gamma = 5 \times 10^{-6}$ and $\gamma = 10^{-5}$, while requiring less S and \mathcal{X} and basically the same number of C than VA_m for a tolerance of $\gamma = 10^{-6}$. The savings can be considerable, being of 35% in terms of S and \mathcal{X} and of 21% in terms of C for the case of $\gamma = 10^{-5}$. The PMA and FMA can reduce even more the required number of S and \mathcal{X} , even to less than half of those of the VA_m (e.g. FMA with $\gamma = 10^{-5}$), but they require an increase in the number of C . It is interesting to note that FMA requires less operations than PMA.

Analysing Table 4, we reach similar conclusions, with the PMA and FMA considerably reducing the number of required S and \mathcal{X} , while increasing the number of C with respect to the VA_m.⁵ Again, the MMA requires less S and \mathcal{X} than VA_m for the three tolerances, while requiring basically the same number of C than VA_m for $\gamma = 10^{-6}$ and reducing it for larger γ . The above results explicitly show the performance-complexity trade-off that can be operated by means of the proposed algorithms. Depending on the applications, it may be interesting to lose just a little bit of performance in exchange of a reduced complexity, which may reflect on the device battery lifetime. Moreover, we can also conclude that PMA and FMA are the best option only if summations and multiplications have a cost that is similar to that of comparisons in the specific receiver architecture. However, if comparisons cost more, what is more usual, then MMA would be the best choice among our proposed algorithms.

⁵Similar results were obtained considering other codes of different rates and memory orders and are omitted here for the sake of brevity.

6 Final comments

In this paper, we proposed sub-optimum decoding algorithms to be used with the minimal trellis associated to the convolutional code. The algorithms are all variations of the M-algorithm, and their operation is matched to characteristics inherent to the minimal trellis. Three algorithms were proposed. Numerical results showed that considerable complexity reductions can be obtained with respect to the Viterbi algorithm operating over the conventional and the minimal trellis, at the cost of a very small reduction in the BER performance. Moreover, the best choice among the proposed algorithms, in terms of complexity reduction, depends on the minimal trellis topology and on the particular receiver architecture. As a future work, we intend to investigate the performance of the proposed algorithms in the fast fading channel (or the Rayleigh channel) scenario.

Acknowledgment This work was partially supported by CNPq under grants 561443/2008-4 and 132078/2009-0.

References

1. Lin S, Costello DJ (2004) Error control coding, 2nd edn. Prentice-Hall, Englewood Cliffs
2. IEEE Standard 802.16e-2005 (2006) IEEE standard for local and metropolitan area networks part 16: air interface for fixed and mobile broadband wireless access systems
3. 3GPP TS 45.003 (2007) 3rd generation partnership project; technical specification group GSM/EDGE radio access network; channel coding (release 7)
4. ETSI TS 101 475 (2000) Broadband radio access networks (BRAN), HYPERLAN type 2, physical (PHY) layer
5. IEEE Standard 802.11 (1999) Wireless LAN medium access control (MAC) and physical (PHY) layer specifications: high speed physical layer in the 5 GHz band

6. Masselos K, Blionas S, Rautio T (2002) Reconfigurability requirements of wireless communication systems. In: Proc. IEEE workshop on heterogeneous reconfigurable systems on chip
7. Bougard B et al (2004) Energy-scalability enhancement of wireless local area network transceivers. In: Proc. IEEE workshop on signal processing advances in wireless communications
8. Dohler M, Heath R, Lozano A, Papadias C, Valenzuela R (2011) Is the PHY layer dead? *IEEE Commun Mag* 49(4):159–165
9. Pentikousis K (2010) In search of energy-efficient mobile networking. *IEEE Commun Mag* 48(1):95–103
10. Demosthenous A, Taylor J (1999) Low-power CMOS and BiCMOS circuits for analog convolutional decoders. *IEEE Trans Circuits Syst II* 46(8):1077–1080
11. Tomatsopoulos B, Demosthenous A (2008) A CMOS hard-decision analog convolutional decoder employing the MFDA for low-power applications. *IEEE Trans Circuits Syst I Regular Pap* 55(9):2912–2923
12. Lin C-C, Shih Y-H, Chang H-C, Lee C-Y (2006) A low power turbo/Viterbi decoder for 3GPP2 applications. *IEEE Trans Very Large Scale Integr (VLSI) Syst* 14(4):426–430
13. Abdallah RA, Shanbhag NR (2008) Error-resilient low-power Viterbi decoders via state clustering. In: Proc. IEEE workshop on signal processing systems
14. Zigangirov KS, Kolesnik VD (1980) List decoding of trellis codes. *Probl Control Inf Theory* 6:347–364
15. Anderson JB (1989) Limited search trellis decoding of convolutional codes. *IEEE Trans Inf Theory* 35(5):944–955
16. Adachi F (1995) Reduced-state Viterbi differential detection using a recursively estimated phase reference for M-ary DPSK. *IEE Proc Commun* 142(4):263–270
17. Chan F, Haccoun D (1997) Adaptive Viterbi decoding of convolutional codes over memoryless channels. *IEEE Trans Commun* 45(11):1389–1400
18. Henning R, Chakrabarti C (2004) An approach for adaptively approximating the Viterbi algorithm to reduce power consumption while decoding convolutional codes. *IEEE Trans Signal Process* 52(5):1443–1451
19. Pottie GJ, Taylor DP (1989) A comparison of reduced complexity decoding algorithms for trellis codes. *IEEE J Sel Areas Commun* 7(9):1369–1380
20. McEliece RJ, Lin W (1996) The trellis complexity of convolutional codes. *IEEE Trans Inf Theory* 42(6):1855–1864
21. Pimentel C, Souza RD, Uchôa-Filho BF, Pellenz ME (2008) Generalized punctured convolutional codes with unequal error protection. *EURASIP J Adv Sig Proc* 2008: 1–6. doi:10.1155/2008/28083
22. Katsiotis A, Rizomiliotis P, Kalouptsidis N (2010) New constructions of high-performance low-complexity convolutional codes. *IEEE Trans Commun* 58(7):1950–1961
23. Rosnes E, Ytrehus Ø (2004) Maximum length convolutional codes under a trellis complexity constraint. *J Complex* 20:372–408
24. Uchôa-Filho B, Souza RD, Pimentel C, Jar M (2009) Convolutional codes under minimal trellis complexity measure. *IEEE Trans Commun* 57:1–5
25. Hug F, Bocharova I, Johannesson R, Kudryashov BD (2009) Searching for high-rate convolutional codes via binary syndrome trellises. In: Proc IEEE int symp inform theory (ISIT 2009). Seoul, Korea, pp 1358–1362
26. Jondral FK (2005) Software-defined radio: basics and evolution to cognitive radio. *EURASIP J Wirel Comm Netw* 2005(3):275–283
27. Mitola J (2000) Cognitive radio: an integrated architecture for software defined radio. Ph.D. dissertation, KTH, Stockholm, Sweden
28. Vardy A (1998) Trellis structure of codes. In: Pless VS, Huffman WC (eds) *Handbook of coding theory*, vol II. North-Holland, Amsterdam
29. Pedroni BU, Pedroni VA, Souza RD (2010) Hardware implementation of a Viterbi decoder using the minimal trellis. In: Proc of the 4th inter symp on commun control and signal processing (ISCCSP'2010). Limassol, Cyprus, pp 1–4
30. McEliece RJ (1996) On the BCJR trellis for linear block codes. *IEEE Trans Inf Theory* 42(4):1070–1092
31. Chatzigeorgiou I, Demosthenous AA, Rodrigues MRD, Wassell IJ (2010) Performance-complexity tradeoff of convolutional codes for broadband fixed wireless access systems. *IET Comm* 4(4):419–427
32. Vucetic B, Yuan J (2000) *Turbo codes: principles and applications*. Kluwer Academic, Boston
33. Knuth D (1997) *The art of computer programming*, vol 3: sorting and searching, 3rd edn. Addison-Wesley, Reading