Channel Coding Techniques with Emphasis on Convolutional and Turbo Codes

Alexandros Katsiotis [⋆]

National and Kapodistrian University of Athens
Department of Informatics and Telecommunications
akats@di.uoa.gr

Abstract. In this thesis, a family of low complexity convolutional codes is constructed, by modifying appropriately the trellis diagram of punctured convolutional codes. The goal is to improve performance at the expense of a reasonable low increase of the trellis complexity. Many new convolutional codes of various code rates and values of complexity are provided. In many cases, a small increase in complexity can lead to a great improvement of performance, compared to punctured convolutional codes. Furthermore, a method is presented for designing new flexible convolutional codes, by combining the techniques of path pruning and puncturing. The new codes can vary their rate, as well as the complexity of their trellis diagram, and hence the computational complexity of the decoding algorithm, leading to coding schemes that manage more efficiently the system resources, compared to classical variable rate convolutional codes. The complexity of the trellis diagram affects the computational complexity of every trellis based decoding algorithm, like BCJR and its variations. Thus, the possibility of applying the aforementioned results using recursive convolutional encoders, which are used as constituent encoders in turbo codes, is investigated. The goal is to construct efficient flexible turbo coding schemes. Simulation results indicate that in specific ranges of the signal to noise ratio, a great decrease in the computational complexity of the decoding procedure can even result to a decrease in the bit error rate.

1 Introduction

Traditionally, an (n, k, m) convolutional code $\mathcal{C}$ can be represented by a "conventional" semi-infinite trellis diagram. Although the trellis is semi-infinite, it consists, after a short initial segment, of concatenated copies of an 1-section structure called *trellis module*. The conventional *trellis module* consists of 2^m "initial states", and 2^m "final states". Each "initial state" is connected by a directed edge to 2^k "final states". Furthermore, each edge is labeled by an n-size binary tuple, which corresponds to the output of the encoder during the specific state transition. Thus, the conventional trellis module contains $n2^{m+k}$ edge bits,

[⋆] Dissertation Advisor: Nicholas Kalouptsidis, Professor and this is called the *trellis complexity* of the module [1]. When a trellis module is used for maximum likelihood decoding of $\mathcal{C}$, the decoding complexity is proportional to the trellis complexity of the module. Particularly, the trellis complexity of the conventional trellis module increases exponentially with k and m, which means that especially for high rate codes, maximum likelihood decoding becomes a very complex procedure.

In order to overcome this problem, punctured convolutional codes (PCC) were introduced by Cain et al. [2]. The trellis module of an (n, k, m) PCC is constructed by "blocking" k "conventional" trellis modules of an $(N, 1, m)$ "mother" code, and then deleting (puncturing) all but n edge bits from each output tuple of this "block". The semi-infinite trellis diagram of the PCC consists (after a short initial transient) of repeated copies of the k-section *punctured trellis module.* The trellis complexity of an (n, k, m) PCC is equal to $n2^{m+1}$ and it is significantly less than the complexity of the "conventional" trellis module of an (n, k, m) convolutional code. PCCs have been extensively studied for various rates [2–4]. Bocharova et al. [4] published comprehensive tables with the best (n, k) PCCs for $3 \leq n \leq 8$, $2 \leq k \leq n - 1$.

Apart from the low decoding complexity, another significant advantage of PCCs is *rate variability.* That is, a whole family of PCCs of various code rates can be constructed, using a single mother code and a variety of puncturing matrices [3, 5]. Each one of the respective codes can be decoded using the decoder (and hence the trellis) of the mother code. Hence, depending on the channel conditions the suitable code is used, a fact that results in a more efficient use of bandwidth. The flexibility offered by PCCs is valuable for modern communication systems, like for instance the IEEE 802.22 standard [6]. Rate variability can be also achieved by using the technique of *path pruning* [7]. That is, using determinate and indeterminate information bits in order to prune away some codeword paths from the trellis of a convolutional code.

An (n, k, m) convolutional code C can be represented by various trellis diagrams (periodic in general), and hence various trellis modules [1, 8]. Apart from the trellis complexity, an important quantity associated with a trellis module is the total number of merges contained in it. The number of merges at a specific state is equal to the total number of branches reaching it minus one [9]. The trellis complexity and the total number of merges are equal to the number of real additions and real comparisons respectively, the Viterbi algorithm has to perform per trellis module [1, 9]. Hence, they constitute the computational complexity of the decoding procedure of C, for a specific trellis module. McEliece et al. demonstrate in [1] that any (n, k, m) convolutional code has a minimal trellis representation, which in most cases is significantly less complex (in terms of trellis complexity), than the conventional trellis diagram. A minimal trellis has the minimum possible complexity and can be constructed directly by a trellis canonical generator matrix. The minimal trellis module minimizes many other quantities, like the total number of states, the total number of merges, etc. They also show that PCCs are simply convolutional codes whose generator matrices have a special structure, and this fact explains the reason why they can be represented by trellis diagrams of low complexity. Finally, they set under consideration the existence of other classes of low complexity convolutional codes, that contain good codes.

2 New Constructions of Low Complexity Convolutional Codes

In this section, we introduce a class of low complexity convolutional codes, produced by modifying the time varying punctured trellis module [10]. Our goal is to achieve better performance than the initial PCC code, at the expense of a reasonable low increase of the trellis complexity. The increase of the complexity is achieved by allowing variations in the state dimension of the trellis module.

We consider a periodic time-varying linear encoder of period k [11], described by the $1 \times n_t$ binary matrices $\mathbf{G}_j(t)$, for $0 \leq j \leq m_t$, such that

$$\boldsymbol{v}_t = \sum_{j=0}^{m_t} u_{t-j}\mathbf{G}_j(t), \qquad t \geq 0 \tag{1}$$

where u_t and $\boldsymbol{v}_t$ denote the input value of size 1 and output value of size n_t respectively at time instant t. By way of convention, $\mathbf{G}_j(t) = \mathbf{0}$ for $j < 0$ and $j > m_t$, where m_t is the memory of the encoder at the t-th time instant. It holds, $\mathbf{G}_j(t+k) = \mathbf{G}_j(t)$, $m_{t+k} = m_t$, and $n_{t+k} = n_t$ for all j and t.

The trellis diagram representing the above construction is periodic (after a short initial transient) and consists of repeated copies of a k-section structure, called *trellis module*. During a trellis module, k information bits are associated to n encoded bits, where $n = \sum_{t=0}^{k-1} n_t$. The state space dimension of the trellis module at the t-th time instant is m_t, $0 \leq t \leq k-1$. We call the time interval of the trellis module between the t-th and the $(t+1)$-th time instant, as the t-th *trellis section*. $v_{t,i}$ denotes the i-th output bit which is transmitted during the t-th trellis section, for $1 \leq i \leq n_t$, and for $0 \leq t \leq k-1$, and is produced by the generator sequence $\mathbf{g}^{(t,i)} = [g_0^{(t,i)}\ g_1^{(t,i)} \cdots g_{m_t}^{(t,i)}]$.

Given a positive integer m, we impose the following restrictions: $m_t \in \{m, m+1\}$, and $m_0 = m$. We denote by T_{ζ_1,ζ_2} the set of *trellis sections* t, $0 \leq t \leq k-1$, during which the state dimension changes from ζ_1 to ζ_2, where $\zeta_1, \zeta_2 \in \{m, m+1\}$. That is, $T_{\zeta_1,\zeta_2} = \{t | m_t = \zeta_1,\ m_{t+1} = \zeta_2,\ 0 \leq t \leq k-1\}$. We further confine the code structure so that the encoding matrices $\mathbf{G}_j(t) = [g_j^{(t,1)}\ g_j^{(t,2)} \cdots g_j^{(t,n_t)}]$ satisfy the following properties:
P_1. $\mathbf{G}_0(t) = \mathbf{1}$, $0 \leq t \leq k-1$,
P_2. $\mathbf{G}_m(t) = \mathbf{1}$, when $t \in T_{m,m}$,
P_3. $\mathbf{G}_{m+1}(t) = \mathbf{1}$, when $t \in T_{m+1,m+1}$,
P_4. $g_m^{(t,i)} \neq 0$ or $g_{m+1}^{(t,i)} \neq 0$, for all $1 \leq i \leq n_t$, and $\mathbf{G}_m(t)$ and $\mathbf{G}_{m+1}(t)$ are linearly independent, when $t \in T_{m+1,m}$.
Note that from Property P_4, it holds that $n_t \geq 2$, when $t \in T_{m+1,m}$. We denote by p_t the smallest integer, $1 \leq p_t < n_t$, such that

$$g_m^{(t,p_t+1)} = \cdots = g_m^{(t,n_t)} \quad \text{and} \quad g_{m+1}^{(t,p_t+1)} = \cdots = g_{m+1}^{(t,n_t)}$$

when $t \in T_{m+1,m}$.

The *trellis complexity* (TC) of a T-section *trellis module* M is defined as the total number of edge symbols contained in the module per information bit [1]. That is, $TC(M) = \sum_{t=0}^{T-1} n_t 2^{m_t+k_t}$, where k_t is the number of information bits associated with the t-th trellis section. It is straightforward to verify that the trellis complexity of the new module M associated with the new code is given by

$$\begin{aligned} TC(M) = &\sum_{t \in T_{m,m}} n_t 2^{m+1} + \sum_{t \in T_{m,m+1}} n_t 2^{m+1} \\ &+ \sum_{t \in T_{m+1,m+1}} n_t 2^{m+2} + \sum_{t \in T_{m+1,m}} n_t 2^{m+2} \end{aligned} \tag{2}$$

since $k_t = 1$, for $0 \leq t \leq k-1$.

Theorem 1. *The complexity of the minimal trellis module M_{min} of $\mathcal{C}$ is given by*

$$\begin{aligned} TC(M_{min}) = &\sum_{t \in T_{m,m}} n_t 2^{m+1} + \sum_{t \in T_{m,m+1}} n_t 2^{m+1} \\ &+ \sum_{t \in T_{m+1,m+1}} n_t 2^{m+2} + \sum_{t \in T_{m+1,m}} (n_t + p_t) 2^{m+1}. \end{aligned} \tag{3}$$

Comparison of (3) and (2), indicates that the initial trellis module is not minimal. Next, we show that if we replace every trellis section $t \in T_{m+1,m}$ by an equivalent pair of trellis sections, the resulting trellis module is minimal.

Theorem 2. *The trellis section $t \in T_{m+1,m}$ is equivalent to a pair of trellis sections, where the first trellis section t_1 carries the first p_t of the n_t output bits and has one bit input. Furthermore, it has 2^{m+1} initial states and 2^{m+1} final states. The second section t_2 produces the last $n_t - p_t$ output bits of the original trellis section t and it is informationless. Moreover, it has 2^{m+1} initial states and 2^m final states.*

The first and second sections of the pair contain $p_t 2^{m+2}$ and $(n_t - p_t)2^{m+1}$ edge symbols respectively (the second section is informationless, thus only one edge diverges from each initial state). Thus, in total the pair of sections contains $(n_t + p_t)2^{m+1}$ edge symbols. If we replace in (2) the term $\sum_{t \in T_{m+1,m}} n_t 2^{m+2}$ with $\sum_{t \in T_{m+1,m}} (n_t + p_t) 2^{m+1}$, relation (3) results. That is, by simply replacing in the initial trellis module of a code $\mathcal{C}$ each one of the trellis sections $t \in T_{m+1,m}$ with its equivalent pair of sections, the result is a minimal trellis module for $\mathcal{C}$.

2.1 Search Results

The proposed search procedure initializes with the punctured trellis module of an (n, k, m) convolutional code, and increases the state space dimension of specific trellis sections from m to $m+1$, performing search over the generator sequences $\mathbf{g}^{(t,i)}$. The objective is to determine codes of better spectra at the expense of higher trellis complexity. However complexity is maintained at affordable levels, as it remains less than the trellis complexity of the respective $(n, k, m+1)$ PCC. The search was conducted over a range of rates and memory sizes, and many good codes were found. The following comments are worth to point out:

- Some of the new codes achieve greater d_f than the respective PCCs of memory m, at the expense of a small increase of the TC.
- Some of the new codes achieve d_f equal to the d_f of the PCCs of memory $m+1$. That is, they have similar asymptotic performance (for high signal to noise ratio) with the corresponding $(n, k, m+1)$ PCCs but less trellis complexity.
- Many of the new codes achieve a specific d_f, with the least known trellis complexity (for specific rate).
- In many other cases, a small increase of TC was enough to significantly improve the spectra (for equal d_f) of the best PCC of memory m.

3 Flexible Convolutional Codes: Variable Rate and Complexity

In this section, we present a method that combines the techniques of path-pruning and puncturing, in order to construct convolutional codes that can vary both their rate and the computational complexity of the decoding procedure [12]. Starting from an $(n, 1, m)$ mother convolutional code, we construct a large family of codes of various code rates. For each code rate the family contains trellis modules of various values of computational complexity. Path pruning is used in order to remove from the mother trellis an amount of state transitions (branches), a fact that results into a less complex trellis. Furthermore, puncturing is utilized for adjusting the code rate. Particularly, the two aforementioned techniques are employed as follows.

Let u_t be the information bits and $\hat{u}_t$ the input bits of the mother encoder. The path-pruning on the trellis of a convolutional code can be implemented by mapping the information u_t and register state bits into the final input $\hat{u}_t$ of the encoder.

More precisely, every T_{pr} time units the single input bit of the encoder is not an information bit, rather it is computed as a linear combination of bits of the current state $S_t = \{\hat{u}_{t-1}, \cdots, \hat{u}_{t-m}\}$, i.e.

$$\hat{u}_{t_1 T_{pr}+t_2} = \begin{cases} u_{t_1(T_{pr}-1)+t_2} & \text{, if } t_2 \neq 0 \\ \sum_{i=1}^{\hat{d}} c_i \hat{u}_{t_1 T_{pr}-i}, & \text{if } t_2 = 0 \end{cases} \tag{4}$$

where $t_1 = \left\lfloor \frac{t}{T_{pr}} \right\rfloor$, $t_2 = t \bmod T_{pr}$, $t = 1, 2, \ldots,$ and $\hat{d}$ is the degree of the polynomial $c(X) = \sum_{i=1}^{m} c_i X^i$. The binary coefficients are chosen, such that

$c_i = 0$ for $0 = i \bmod T_{pr}$, i.e. the non information bits are produced by the linear combination of information bits only.

Path-pruning results to an $(nT_{pr}, T_{pr} - 1)$ time-varying convolutional code. The resulting semi-infinite trellis diagram starts at the all zero state, and after a short initial transient, it becomes periodic, consisting of concatenations of a T_{pr}-section trellis module M_{pr}.

Theorem 3. *The state space dimension s_l^{pr} at depth l and the branch space dimension b_l^{pr} of the l-th trellis section of trellis module $\mathcal{M}_{pr}$ are given by*

$$s_l^{pr} = \begin{cases} m - 1 - \hat{\beta}, & \text{for } 1 \le l \le \hat{\alpha} \\ m - \hat{\beta}, & \text{for } \hat{\alpha} + 1 \le l \le T_{pr} \end{cases}$$

$$b_l^{pr} = \begin{cases} m - \hat{\beta}, & \text{for } 1 \le l \le \hat{\alpha} \text{ or } l = T_{pr} \\ m + 1 - \hat{\beta}, & \hat{\alpha} + 1 \le l \le T_{pr} - 1 \end{cases}$$

where $\hat{\alpha} = m - \hat{d} \bmod T_{pr}$, $\hat{\beta} = \left\lfloor \frac{m - \hat{d}}{T_{pr}} \right\rfloor$.

The final trellis module M_{pu} is constructed by concatenating p copies of the M_{pr} module, and puncturing a portion of the encoded bits based on a specific puncturing matrix P, in order to adjust the rate. The final trellis module consists of $T_{pu} = pT_{pr}$ trellis sections. The total number of merges E_{pu} contained in M_{pu} are

$$E_{pu} = p(T_{pr} - \frac{\hat{a}}{2} - 1) \cdot 2^{m - \hat{b}}. \quad (5)$$

Depending on the puncturing matrix, the trellis complexity of the final trellis module M_{pu} is equal to

$$TC(M_{pu}) = \sum_{j=1}^{pT_{pr}} n_j 2^{b^{pr}_{j \bmod T_{pr}}}, \quad (6)$$

where $0 \le n_j \le n$ is the number of output bits of the j-th section of M_{pu}, for $1 \le j \le pT_{pr}$. The final trellis module M_{pu} corresponds to a $(pT_{pr}n - \tilde{n}, p(T_{pr} - 1))$ convolutional code, where $\tilde{n}$ is the number of the punctured encoded bits, i.e. $pT_{pr}n - \tilde{n} = \sum_{j=1}^{pT_{pr}} n_j$.

Given an $(n, 1, m)$ mother code, for various choices of T_{pr}, p, $c(X)$ and puncturing matrix P, arbitrarily many codes of different code rates and values of computational complexity can be constructed. In this study, we restrict ourselves to a subset according to the following criterion.

Given a rate k'/n', we construct codes with computational complexity (i.e. trellis complexity and number of merges) equal to the computational complexity of the trellis module of a PCC with memory size m', for various values of m'.

3.1 Results and Simulations

In this study we present three families of flexible codes, generated by a $(2, 1, 8)$, a $(3, 1, 8)$ and a $(4, 1, 8)$ mother convolutional code respectively. All codes are of rates $(n'-1)/n'$, for $2 \leq n' \leq 8$. Many of the codes have the same free distance with the best codes of the same rate and complexity presented in [4, 13].

We have simulated the performance of various codes produced by the $(4, 1, 8)$ mother convolutional code, for the AWGN channel, using BPSK modulation. Some of the results are depicted in Figure 1, which indicates BER plots for all values of m' and rates 1/2, 2/3 and 3/4.

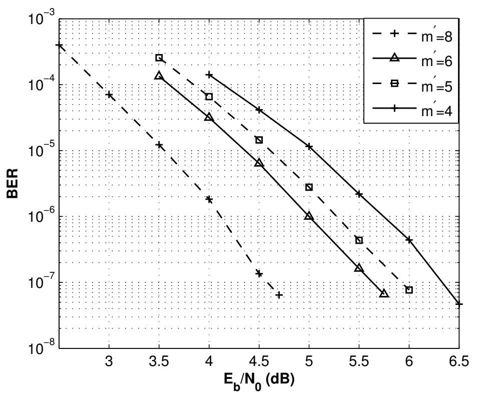
In contrast to the classical variable rate convolutional codes [3, 5] (i.e. a mother convolutional encoder and a set of puncturing matrices), which "exchange" coding gain for bandwidth and vice versa, the proposed constructions add one more "dimension", i.e. the computational complexity of decoding, hence leading to coding schemes that manage more efficiently the system resources. Indeed, consider the family of codes produced by the $(4, 1, 8)$ mother convolutional code. Assume that the SNR is equal to 7.1dB, and that the code of rate 3/4 and $m' = 4$ ($TC = 42.7$, $E_{pu} = 16$, all normalized by k') is used. Then, an error probability of 10^{-7} is achieved (Fig. 1(c)). Furthermore, assume that the SNR decreases by 0.8dB. There are many strategies that can be followed during the next transmission, for the anticipation of the SNR's decrease. For instance, one choice is the use of the code of rate 1/2 and $m' = 4$ ($TC = 64$, $E_{pu} = 16$) (Fig. 1(a)), i.e. decreasing the rate and keeping the complexity in similar levels. Another option is the use of the code of rate 3/4 and $m' = 6$ (Fig. 1(c)) ($TC = 170.7$, $E_{pu} = 64$), i.e. keeping the rate constant and increasing the complexity. Finally, the code of rate 2/3 and $m' = 5$ ($TC = 96$, $E_{pu} = 32$) (Fig. 1(b)) could be used, resulting in a small decrease in the rate and a small increase in complexity.

4 Recursive Flexible Convolutional Encoders for Parallel Concatenation

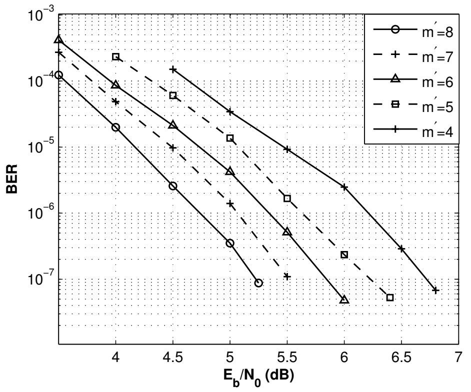
In this section, we extend the previous analysis and we combine path-pruning and puncturing for the construction of flexible convolutional codes for parallel concatenation that can vary both the rate and the computational complexity of the decoding procedure, leading to flexible turbo codes.

Consider a trellis module as it is constructed in section 3, that corresponds to a code of rate k'/n' and has the complexity profile of the respective PCC with memory m'. It can be shown that if the Max-Log-MAP algorithm is used for decoding, then $(n'+q+2k')2^{m'+1}+k'$ additions and $4k'2^{m'}-k'$ comparisons have to be performed in order to decode k' information bits. q denotes the number of trellis section that carry encoded bits

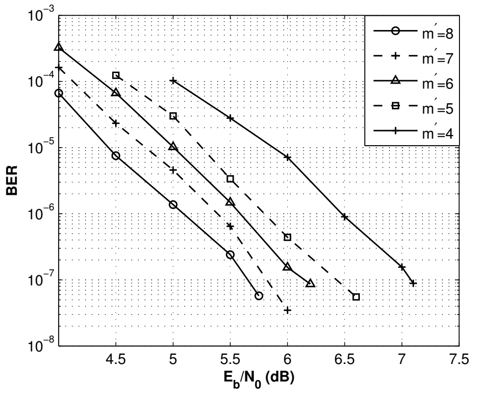
Consider a feedforward $(n, 1, m)$ mother convolutional encoder, described by the polynomial generator matrix $G(D) = \begin{bmatrix} g^{(0)}(D) & g^{(1)}(D) \dots g^{(n-1)}(D) \end{bmatrix}$. Assume that for particular path-pruning and puncturing parameters (i.e. $c(X)$, T_{pr}, p, P), a specific code C_{pu} and its respective trellis module M_{pu} are constructed. Consider also the equivalent $(n, 1, m)$ recursive systematic mother encoder given by $G_{sys}(D) = \begin{bmatrix} 1 & \frac{g^{(1)}(D)}{g^{(0)}(D)} \dots \frac{g^{(n-1)}(D)}{g^{(0)}(D)} \end{bmatrix}$. It can be shown that by using the same



(a)



(b)



(c)

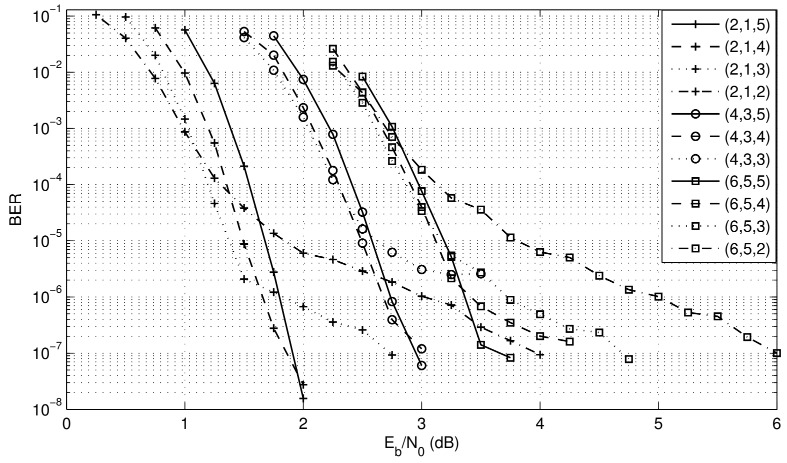
Fig. 1. BER simulations of codes produced by the $(4, 1, 8)$ mother convolutional code, for various values of m' and rates (a) 1/2, (b) 2/3 and (c) 3/4.

parameters T_{pr}, p, P and the pruning polynomial $c'(X) = \sum_{i=1}^{m}(c_i + g_i^{(0)})X^i$ instead of the polynomial $c(X)$, the same code C_{pu} is produced and a corresponding trellis module M'_{pu}. c_i and $g_i^{(0)}$ are the coefficients of polynomials $c(X)$ and $g^{(0)}(D)$ respectively. Trellis modules M_{pu} and M'_{pu} have identical complexity profiles and they produce the same code. However, they correspond to different encoders. Hence, we can apply the results and the analysis presented in section 3, in case where recursive mother convolutional encoders are used.

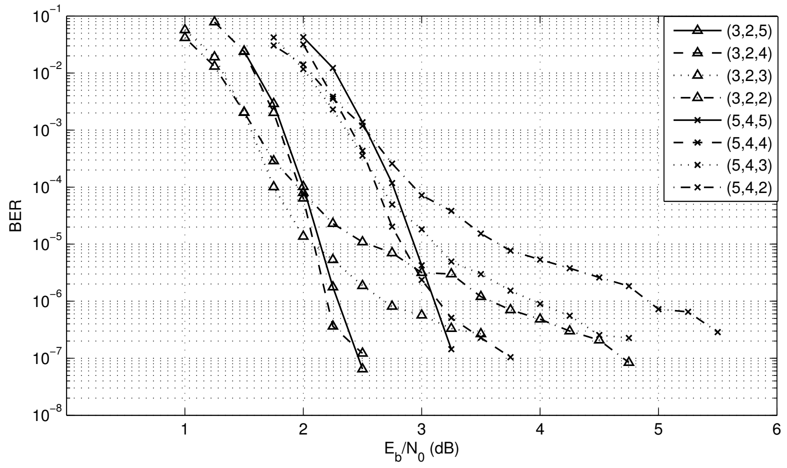
In this thesis we provide a family of recursive systematic convolutional encoders constructed from a $(2, 1, 5)$ recursive systematic mother encoder. All encoders are of rates $(n' - 1)/n'$, for $2 \leq n' \leq 6$. In [14], the authors provide the best recursive systematic punctured convolutional encoders for parallel concatenation, constructed from $(2, 1)$ mother encoders. Variable rate encoders are also provided. Almost all the encoders constructed in this thesis achieve the same value of distance d_2, as the encoders in [14].

We have simulated the performance of turbo codes that use the flexible mother encoder presented in this thesis and its associative family, as constituent encoders, for the AWGN channel using BPSK modulation and the Max-Log-MAP decoding algorithm. In Fig. 2 we have used a random interleaver of length 1000 for all the members of the family. Furthermore, for a particular encoder and a particular SNR value we have assumed a fixed number of iterations L^f_{SNR}, which is less than or equal to 10. Particularly, L^f_{SNR} is equal to the minimum number of iterations needed to achieve the minimum possible error probability, if the former is less than 10, and equal to 10 otherwise. Note that, apart from the computational complexity of the decoding algorithm that uses a particular trellis module, the overall computation complexity of turbo decoding depends on the number of iterations as well. Thus, in Table 1 we provide the values of L^f_{SNR} associated with the BER plots in Fig. 2. Particularly, the first row of Table 1 indicates the curve points. A specific curve point for a specific encoder corresponds to a particular value of SNR (Fig. 2). As depicted in Fig. 2, the *waterfall region* for the (n', k', m') constituent codes (of all rates) for $m' < m$ appears at smaller values of SNR than the waterfall region in case where the respective (n', k', m) codes of the family (including the mother code) are used. This comes with a significant reduction in computational complexity. In other words, in particular SNR regions, a reduction in complexity can result in an increase of performance. As an example, the $(2, 1, 5)$ mother code achieves error probability $2 \cdot 10^{-4}$ for SNR equal to 1.5dB (Fig. 2a). The $(2, 1, 3)$ code, for the same SNR value, achieves error probability $2 \cdot 10^{-6}$ and reduces the computational complexity by 75%. Similar behavior can be observed for all code rates. As a matter of fact, there is no need to use the (most complex) (n', k', m) code of each rate for error probabilities greater than 10^{-6}.

The proposed constructions can lead to turbo coding schemes that manage more efficiently the system resources. Assume that the SNR is equal to 1.75dB, and that the turbo code uses the $(2, 1, 4)$ (9 iterations) convolutional encoder from the family, achieving error probability $3 \cdot 10^{-7}$ (Fig. 2a). Furthermore, assume that the SNR increases by 1.75dB. In this case, there is no need for using a



(a)



(b)

Fig. 2. Simulations of turbo codes that use as constituent encoders the constructed family of flexible convolutional encoders.

powerful constituent encoder. During the next transmission we can increase the rate, or decrease the complexity. For instance, we can use the $(5,4,4)$ constituent encoder (Fig. 2b) (7 iterations), increasing significantly the rate and keeping the same value of m'. Furthermore, the $(2,1,2)$ encoder (Fig. 2a) (3 iterations) can be used, i.e. keeping the rate constant and decreasing substantially the complexity. At this point, one may wonder whether the reduction in computational complexity can be achieved by simply reducing the number of iterations. The computational complexity that corresponds to 3 iterations of the $(2,1,2)$ decoder is slightly less than the complexity of one iteration of the $(2,1,4)$ decoder which was initially used. The $(2,1,4)$ code achieves error probability $6 \cdot 10^{-5}$ at SNR equal to 3.5dB, if only one iteration is performed.

Table 1. number of iterations of the codes in fig. 2

code	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
(2,1,5)	3	8	10	10	10											
(2,1,4)	5	7	10	10	9	9										
(2,1,3)	4	7	10	10	10	9	7	6	5	3						
(2,1,2)	3	6	10	10	10	8	8	7	5	5	5	3	3	3	3	3
(3,2,5)	7	10	10	10	9											
(3,2,4)	2	5	10	10	10	10										
(3,2,3)	5	6	10	10	10	8	8	7	4	3	3					
(3,2,2)	5	7	10	10	8	8	7	7	6	4	4	4	3	3	3	3
(4,3,5)	5	10	10	10	10	10										
(4,3,4)	5	6	10	10	10	10	8									
(4,3,3)	4	8	10	10	10	10	7	6	5							
(5,4,5)	3	6	10	10	10	10										
(5,4,4)	4	9	10	10	10	10	7	6								
(5,4,3)	3	7	10	10	9	8	8	7	6	5	5	4	4			
(5,4,2)	3	5	7	8	8	8	7	6	6	6	5	3	3	3	2	2
(6,5,5)	7	10	10	10	10	7										
(6,5,4)	5	10	10	10	10	7	6	6	6							
(6,5,3)	5	9	10	10	9	8	7	6	6	5	5					
(6,5,2)	4	6	10	10	8	7	7	6	5	5	3	3	3	2	2	2

5 Conclusion

In this thesis, a family of low complexity convolutional codes was proposed, which was constructed by modifying appropriately the trellis diagram of punctured convolutional codes. Many new convolutional codes of various code rates and values of complexity were provided. Furthermore, a method was presented for designing new flexible convolutional codes that can vary both their rate and the computational complexity of the decoding procedure, leading to coding schemes that manage more efficiently the system resources, compared to classical variable rate convolutional codes. This method was extended in order to construct recursive flexible convolutional encoders. Their use as constituent convolutional encoders in turbo codes can lead to pretty flexible parallel concatenated coding schemes.

References

1. R. J. McEliece and W. Lin, "The trellis complexity of convolutional codes," *IEEE Trans. Inf. Theory*, vol. IT-42, pp. 1855–1864, Nov. 1996.
2. J. B. Cain, J. C. Clark, Jr, and J. M. Geist, "Punctured convolutional codes of rate $(n-1)/n$ and simplified maximum likelihood decoding," *IEEE Trans. Inf. Theory*, vol. IT-25, pp. 97–100, Jan. 1979.
3. P. J. Lee, "Constructions of rate $(n-1)/n$ punctured convolutional codes with minimal required SNR criterion," *IEEE Trans. Commun.*, vol. COM-36, pp. 1171–1173, Oct. 1988.
4. I. E. Bocharova and B. D. Kudryashov, "Rational rate punctured convolutional codes for soft-decision Viterbi decoding," *IEEE Trans. Inf. Theory*, vol. IT-43, pp. 1305–1313, July 1997.
5. G. Begin, D. Haccoun, and C. Paquin, "Further results on high-rate punctured convolutional codes for Viterbi and sequential decoding," *IEEE Trans. Commun.*, vol. COM-38, pp. 1922–1928, Nov. 1990.
6. C. R. Stevenson, G. Chouinard, Z. Lei, W. Hu, S. Shellhamer, and W. Caldwell, "IEEE 802.22: The first cognitive radio wireless regional area network standard," *IEEE Commun. Mag.*, vol. 47, pp. 130–138, Jan. 2009.
7. C. H. Wang and C. C. Chao, "Path-compatible pruned convolutional (PCPC) codes," *IEEE Trans. Commun.*, vol. COM-50, pp. 213–224, Feb. 2002.
8. B. F. Uchoa-Filho, R. D. Souza, C. Pimentel, and M. Jar, "Convolutional codes under a minimal trellis complexity measure," *IEEE Trans. Commun.*, vol. COM-57, pp. 1–5, Jan. 2009.
9. R. J. McEliece, "On the BCJR trellis for linear block codes," *IEEE Trans. Inf. Theory*, vol. IT-42, pp. 1072–1092, July 1996.
10. A. Katsiotis, P. Rizomiliotis, and N. Kalouptsidis, "New constructions of high performance low complexity convolutional codes," *IEEE Trans. Commun.*, vol. 58, pp. 1950–1961, July 2010.
11. M. Mooser, "Some periodic convolutional codes better than any fixed code," *IEEE Trans. Inf. Theory*, vol. IT-29, pp. 750–751, Sept. 1983.
12. A. Katsiotis, P. Rizomiliotis, and N. Kalouptsidis, "Flexible convolutional codes: variable rate and complexity," *IEEE Trans. Commun.*, March 2012.
13. E. Rosnes and Ø. Ytrehus, "On maximum length convolutional codes under a trellis complexity constraint," *Journal of Complexity*, vol. 20, pp. 372-408, March-June 2004.
14. F. Daneshgaran, M. Laddomada, and M. Mondin, "High-rate recursive convolutional codes for concatenated channel codes," *IEEE Trans. Commun.*, vol. COM-52, pp. 1846–1850, Nov. 2004.