The Augmented State Diagram and its Application to Convolutional and Turbo Codes

Ioannis Chatzigeorgiou, *Member, IEEE,* Miguel R. D. Rodrigues, *Member, IEEE,* Ian J. Wassell, and Rolando A. Carrasco

Abstract

Convolutional block codes, which are commonly used as constituent codes in turbo code configurations, accept a block of information bits as input rather than a continuous stream of bits. In this paper, we propose a technique for the calculation of the transfer function of convolutional block codes, both punctured and nonpunctured. The novelty of our approach lies in the augmentation of the conventional state diagram, which allows the enumeration of all codeword sequences of a convolutional block code. In the case of a turbo code, we can readily calculate an upper bound to its bit error rate performance if the transfer function of each constituent convolutional block code has been obtained. The bound gives an accurate estimate of the error floor of the turbo code and, consequently, our method provides a useful analytical tool for determining constituent codes or identifying puncturing patterns that improve the bit error rate performance of a turbo code, at high signal-to-noise ratios.

Index Terms

Turbo codes, convolutional codes, puncturing, transfer function, augmented state diagram

This work was supported by EPSRC under Grant GR/S46437/01. Parts of this paper were presented at the 2006 International Symposium on Turbo Codes and Related Topics and at the 2006 International Conference on Communications.

I. Chatzigeorgiou and I. J. Wassell are with the Computer Laboratory, University of Cambridge, Cambridge CB3 0FD, United Kingdom (e-mail: ic231@cam.ac.uk; ijw24@cam.ac.uk).

Miguel R. D. Rodrigues was with the Computer Laboratory, University of Cambridge, CB3 0FD, United Kingdom. He is now with the Instituto de Telecomunicações and the Department of Computer Science, University of Porto, Porto 4169-007, Portugal (e-mail: mrodrigues@dcc.fc.up.pt).

Rolando A. Carrasco is with the School of Electrical, Electronic and Computer Engineering, University of Newcastle-upon-Tyne, Newcastle NE1 7RU, United Kingdom (e-mail: r.carrasco@newcastle.ac.uk).

I. Introduction

Turbo codes, originally conceived by Berrou *et al.* [1], [2] are widely known for their astonishing performance on the additive white Gaussian noise (AWGN) channel. A tight upper bound on the bit error probability of a turbo code can be easily computed, if the distance properties of the code, conveyed by its transfer function, are known. Calculation of the transfer function of a turbo code is computational infeasible for long deterministic interleavers but the assumption of a uniform interleaver, a concept introduced by Benedetto and Montorsi [3], drastically simplifies the calculations and reduces the computational burden. Methods to evaluate the transfer function of a parallel concatenated convolutional coding scheme have been proposed by Divsalar *et al.* [4] as well as Benedetto and Montorsi [3]. In addition, guidelines for the optimal design of the constituent convolutional codes were presented in [5].

When bandwidth efficiency is of critical importance, the use of high-rate codes is imperative. High-rate turbo codes can be easily obtained by puncturing selected bits from the output of a turbo encoder. Based on the research carried out by Hagenauer [6] on rate-compatible punctured convolutional codes and the work of Haccoun and Bégin [7] on punctured convolutional codes, design criteria for punctured turbo codes were proposed by Barbulescu and Pietrobon [8], Fan Mo *et al.* [9], Açikel and Ryan [10], and Babich *et al.* [11]. Simulation-based analyses to identify a relationship between the structure of a puncturing pattern and the performance of the corresponding punctured turbo code were also carried out by Land and Hoeher [12], Blazek *et al.* [13] and Crozier *et al.* [14]. In turn, an analytic approach to evaluate the performance of punctured turbo codes was developed by Kousa and Mugaibel [15]. The approach is based on a modification of Divsalar's technique that takes the puncturing pattern into account. Although elegant, the proposed approach is only applicable to turbo codes using short interleavers. Furthermore, the authors draw conclusions on rate-1/2 turbo codes assuming that only the parity check outputs of the turbo encoders are punctured.

Inspired by Benedetto and Montorsi's technique, this paper proposes an alternative method based on the concept of the "augmented" state diagram, for the evaluation of the transfer function of turbo codes, both punctured and nonpunctured. The augmented state diagram for nonpunctured constituent convolutional codes and its counterpart for punctured constituent convolutional codes are considered in Sections II and III, respectively. The transfer function of a constituent code can be readily obtained using our approach, however as the length of the input sequence or, equivalently, the size of the interleaver increases, computation becomes intensive. Thus, Section IV proposes a modification of the augmented state diagram that results in a less computationally intensive process. Section V considers the application of the proposed technique to identify puncturing patterns that lead to high-rate turbo codes yielding low error floors. Finally, the main conclusions of this work are summarized in Section VI.

II. NONPUNCTURED CONVOLUTIONAL BLOCK CODES

A. Preliminaries

A codeword sequence generated by a convolutional encoder is often described by a monomial of the form $W^wU^uZ^zL^\ell$, where W, U and Z are indeterminate variables that correspond to the input, systematic output and parity check output sequences, respectively, whilst L corresponds to the associated path of the codeword sequence in the trellis diagram. The exponent of either W, U or Z, namely w, u or z respectively, denotes the Hamming weight of the corresponding sequence, i.e., its Hamming distance from the all-zero sequence. Finally, the exponent of L corresponds to the length of the generated trellis path.

Due to the linear properties of convolutional codes, it is common practice to assume that the all-zero sequence is input to the convolutional encoder and, consequently, the all-zero codeword sequence is generated and transmitted. Conceptually, both the input and the output sequences are of infinite length. In addition, it is assumed that the encoder is initialized to the zero memory state. The transfer function $T(W, U, Z, L)$ of a convolutional code enumerates all codeword sequences of the form $W^wU^uZ^zL^\ell$, represented by paths in the trellis diagram that start from the zero state and remerge with it only once, after ℓ trellis steps.

The transfer function of a convolutional code can be obtained from its state diagram, after labeling each branch according to the input, systematic and parity check weights it conveys and splitting the zero state into two separate states, namely the start state X_S and the end state X_E. As an example, the weight-labeled state diagram of a rate-1/2 recursive systematic convolutional (RSC) encoder, with memory size $\nu = 2$, feedback generator polynomial 7_8 and feedforward generator polynomial 5_8 is shown in Fig. 1. For brevity, we use the notation RSC(1,5/7) to describe the afore-mentioned code. Note that both polynomials are expressed in octal form. Using the weight-labeled state diagram, we can express each memory state as a function of the other states and, hence, obtain the so-called state equations. Upon solving these equations for the ratio X_E/X_S, we obtain the transfer function $T(W,U,Z,L)$.

When the input information sequence has finite length, the corresponding convolutional code can be seen as a block code. Throughout this paper we assume that the trellis of a convolutional block code (CBC) is terminated, i.e., the information sequence contains bits that force the trellis path of fixed length N to return to the zero state [16]. In contrast to $T(W,U,Z,L)$, the transfer function $B(W,U,Z)$ of a CBC enumerates all codeword sequences that correspond to paths of length N; hence, paths that remerge with the zero state more than once and stay at it for a consecutive number of trellis steps are also considered. In the subsequent section, we introduce the concept of the augmented state diagram, which can be used to derive the transfer function of a CBC.

B. The Augmented State Diagram: a Novel Approach

For consistency, we consider the case of the binary RSC(1,5/7) code to demonstrate the method for obtaining the augmented state diagram of a CBC from the weight-labeled state diagram of the original convolutional code. In order to allow a path to revisit the zero state, we have inserted a node X_0, which is different from the states X_S and X_E, as illustrated in Fig. 2. States which are connected to X_S or X_E, have also been connected to this "intermediate" zero state X_0 in a similar manner. The self-loop has been appended, since a path can remain at X_0 for an indefinite period of time. Furthermore, two branches, both with zero input and output weight, are added to connect X_S with X_0 and X_0 with X_E, so as to permit paths to diverge from the all-zero sequence at a time step other than the very first, or to remerge with the all-zero sequence at a time step other than the very last. The resultant augmented state diagram of the RSC(1,5/7) block code, depicted in Fig. 2, is used to derive the system of state equations.

Upon solving the state equations for the ratio X_E/X_S, we obtain

$$\frac{X_E}{X_S} = f_{\text{zero}}(L) + f(W,U,Z,L). \tag{1}$$

The first term, $f_{\text{zero}}(L)$, is the sum of all paths that correspond to all-zero sequences of various lengths. These paths start from state X_S, stay at state X_0 for an indefinite number of steps by circulating around the self-loop and finally terminate at state X_E. Since the transfer function, $B(W,U,Z)$, enumerates all codeword sequences other than the transmitted sequence, and since we have assumed that the all-zero sequence has been transmitted, term $f_{\text{zero}}(L)$ is not of interest, hence it is ignored.

The second term on the right hand side of (1) enumerates all paths of various lengths that start from the zero state, end at the zero state and are different from the all-zero path. Essentially, these paths represent all single-error and multiple-error events. Term $f(W,U,Z,L)$, which we call the extended transfer function of a CBC, can be expressed as

$$f(W,U,Z,L) = \sum_{\ell=1}^{\infty} f_\ell(W,U,Z)L^\ell, \tag{2}$$

where the conditional extended transfer function, $f_\ell(W,U,Z)$, enumerates all codeword sequences having specific path length ℓ. More specifically, $f_\ell(W,U,Z)$ is defined as

$$f_\ell(W,U,Z) \triangleq \sum_{w,u,z} B_{w,u,z,\ell} W^w U^u Z^z, \tag{3}$$

where $B_{w,u,z,\ell}$ denotes the number of codeword sequences of particular path length ℓ, having input, systematic and parity check weights w, u and z, respectively. Consequently, for an input information sequence of length N, the transfer function $B(W,U,Z)$ of a binary CBC is, by definition, equivalent to $f_\ell(W,U,Z)$ for $\ell = N$, that is

$$B(W,U,Z) \triangleq f_N(W,U,Z). \tag{4}$$

For example, let us assume that we want to compute the transfer function of the RSC(1,5/7) block code for an input block length of $N = 4$. Using the augmented state diagram depicted in Fig. 2, we obtain the extended transfer function

$$\begin{aligned} f(W,U,Z,L) &= f_3(W,U,Z)L^3 + f_4(W,U,Z)L^4 + f_5(W,U,Z)L^5 + \ldots \\ &= W^3U^3Z^2L^3 + (W^2U^2Z^4 + 2W^3U^3Z^2)L^4 + \\ &\quad + (2W^2U^2Z^4 + 3W^3U^3Z^2 + W^3U^3Z^4 + W^4U^4Z^2)L^5 + \ldots \end{aligned} \tag{5}$$

For an input information sequence of length $N = 4$, the desired transfer function $B(W,U,Z)$ coincides with the conditional extended transfer function $f_4(W,U,Z)$, hence

$$\begin{aligned} B(W,U,Z) &\triangleq f_4(W,U,Z) \\ &= W^2U^2Z^4 + 2W^3U^3Z^2. \end{aligned} \tag{6}$$

In practice, $f_N(W,U,Z)$ can be directly derived from the ratio X_E/X_S while avoiding the calculation of unnecessary terms [17]. In particular, we can use the properties of binomial series to express the ratio X_E/X_S as a sum of products of polynomials; consequently, only those products that generate monomials containing L^N need to be calculated. These products can be identified and computed using the multinomial theorem. A detailed description of the derivation process is given in the Appendix [1].

The technique we propose can be seen as a refinement of Benedetto's and Montorsi's approach [3]; they both have similar complexity, whilst they are less complex than Divsalar's method [4]. However, our technique can be implemented in a straightforward manner since it is based on the conventional state diagram of convolutional codes and, more importantly, it can be easily extended to punctured convolutional codes, as it will become evident in the following section.

III. Punctured Convolutional Block Codes

A. *Preliminaries*

In certain applications, such as satellite communications, link reliability is of prime importance and, consequently, low rate codes are used to achieve it. However, bandwidth occupancy is of much greater importance in wireless communications and so high-rate codes are preferred. A high-rate convolutional code can be obtained by periodic elimination, known as puncturing, of particular codeword bits from the output of a parent low rate convolutional encoder. If the parent convolutional encoder generates n_0 output sequences, we define which output bits are eliminated at each time-step by means of a puncturing pattern $\mathbf{P}$. The puncturing pattern, which is repeated periodically every M time steps, is represented by a $n_0 \times M$ matrix

$$\mathbf{P} = \begin{bmatrix} p_{1,1} & \cdots & p_{1,M} \\ \vdots & p_{i,j} & \vdots \\ p_{n_0,1} & \cdots & p_{n_0,M} \end{bmatrix}, \tag{7}$$

where $p_{i,j} \in \{0, 1\}$, with $i = 1, \ldots, n_0$ and $j = 1, \ldots, M$. For $p_{i,j} = 0$ the corresponding output bit is punctured otherwise it is transmitted. Note that puncturing does not compromise

[1]One could argue that $B(W, U, Z)$ can also be generated by enumerating all linear combinations of the terms contained in $T(W, U, Z, L)$ that produce paths of length N. Although this combinatorial approach is a possible alternative, it is more computational expensive than our method. As it will become evident in the Appendix, for a given CBC, our method uses a fixed number of equations that contain a constant number of terms. As N increases, only the number of solutions returned by the equations increases. On the contrary, the combinatorial approach tries to solve a problem, *both* the terms and solutions of which grow with N.

the computational complexity of the decoder, whilst a variety of code rates can be achieved by using different puncturing patterns.

The puncturing pattern $\mathbf{P}$ can either be seen as a matrix of row vectors $[\mathbf{P}^{\mathrm{r}}_1 \ \ldots \ \mathbf{P}^{\mathrm{r}}_{n_0}]^{\mathrm{T}}$ or as a matrix of column vectors $[\mathbf{P}^{\mathrm{c}}_1 \ \ldots \ \mathbf{P}^{\mathrm{c}}_M]$; in the former representation $\mathbf{P}^{\mathrm{r}}_i$ is the row puncturing vector (RPV) of the i-th output of the parent convolutional encoder, whilst in the latter representation $\mathbf{P}^{\mathrm{c}}_j$ is the column puncturing vector (CPV) of the j-th puncturing step. We use the latter representation throughout the following subsection, where we introduce the modified augmented state diagram for punctured CBCs.

B. Revisiting the Augmented State Diagram

In this section, we extend the technique based on the augmented state diagram to the case of punctured CBCs. For consistency, we consider the rate-1/2 RSC(1,5/7) code, having a memory size of $\nu = 2$ and, consequently, 4 available states. In order to construct the augmented state diagram of the punctured RSC(1,5/7) block code, we need to introduce the CPV into the labeling procedure of each branch. Recall that in the case of the augmented state diagram of nonpunctured convolutional codes, a branch connecting two states, e.g., X_{k_1} and X_{k_2} with k_1,$k_2 \in [0, 2^\nu - 1]$, is labeled using the notation $W^w U^u Z^z L$. Let us concentrate on the same transition from X_{k_1} to X_{k_2} when puncturing occurs. If $\mathbf{P}^{\mathrm{c}}_j = [p_{1,j} \ \ p_{2,j}]^{\mathrm{T}}$ is the active CPV at a specific step, the label of the branch will change to $W^w U^{u'} Z^{z'} L$, where u' and z' are the weights of the punctured output codewords given by

$$u' = u \cdot p_{1,j}, \quad z' = z \cdot p_{2,j}. \tag{8}$$

We observe that the values of u' and z' depend on the value of the active CPV, hence the label of the branch that connects X_{k_1} with X_{k_2} cannot be constant.

To overcome this problem we introduce M sets of states, namely $\mathbf{Y}_1, \ldots, \mathbf{Y}_M$. Each set $\mathbf{Y}_j$ contains all possible states of the convolutional encoder, i.e., $\mathbf{Y}_j = \{X^{(j)}_0, \ldots, X^{(j)}_{2^\nu - 1}\}$, where the index j next to a state X_k denotes the set which state X_k belongs to. When $\mathbf{P}^{\mathrm{c}}_j$ is the active CPV, a transition from state X_{k_1} to state X_{k_2} in the conventional state diagram corresponds to a transition from state $X^{(j-1)}_{k_1}$ to state $X^{(j)}_{k_2}$ in the modified augmented state diagram. As time progresses CPVs are repeated periodically, i.e., $\mathbf{P}^{\mathrm{c}}_1$,$\mathbf{P}^{\mathrm{c}}_2$,...,$\mathbf{P}^{\mathrm{c}}_M$,$\mathbf{P}^{\mathrm{c}}_1$,..., resulting in transitions to states that belong to sets $\mathbf{Y}_1$,$\mathbf{Y}_2$,...,$\mathbf{Y}_M$,$\mathbf{Y}_1$,..., respectively. Therefore, the problem of having

M labels assigned to a single branch that connects states X_{k_1} and X_{k_2} is overcome by having M branches each one of which pairs state X_{k_1} of a set with state X_{k_2} of the subsequent set.

So as to better understand the concept of the augmented state diagram of a punctured CBC, we give an example for a puncturing period of $M = 2$. In order to increase the code rate of RSC(1,5/7) from 1/2 to 2/3, we use the puncturing pattern

$$\mathbf{P} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \tag{9}$$

which can be decomposed into two CPVs, namely $\mathbf{P}_1^{\mathrm{c}} = [1 \;\; 1]^{\mathrm{T}}$ and $\mathbf{P}_2^{\mathrm{c}} = [1 \;\; 0]^{\mathrm{T}}$. The augmented state diagram of the rate-2/3 RSC(1,5/7) block code is presented in Fig. 3. Solid branches originating from states in set $\mathbf{Y}_2$ and terminating at states in set $\mathbf{Y}_1$, represent transitions during which $\mathbf{P}_1^{\mathrm{c}}$ is the active CPV. Since both elements of $\mathbf{P}_1^{\mathrm{c}}$ are equal to 1, the outputs of the encoder are not punctured therefore the labels of those branches are identical to the labels of the corresponding branches of the augmented diagram in Fig. 2. Dashed branches, originating from states in set $\mathbf{Y}_1$ and terminating at states in set $\mathbf{Y}_2$, represent transitions during which $\mathbf{P}_2^{\mathrm{c}}$ is the active CPV. In this case, the parity check output of the encoder is punctured, therefore term Z does not appear in any of the branch labels. To complete the augmented diagram, states X_S and X_E have to be included. Since the encoder starts from state X_S, $\mathbf{P}_1^{\mathrm{c}}$ is the active CPV during the transition from X_S to a state in set $\mathbf{Y}_1$ at the first time-step. At the last time-step, the encoder returns to the zero state, i.e., a transition to state X_E occurs. In order to terminate the code, those states of each set which are connected to state X_0 of a different set, must also be connected to state X_E.

In the general case of a binary CBC which is punctured using a pattern of period M, we obtain $(M \times 2^{\nu} + 1)$ state equations which we solve for the ratio X_E/X_S. Similarly to the case of nonpunctured convolutional codes presented in Section II, the transfer function $B(W, U, Z)$ of a punctured CBC for a particular input block size N can be derived from the extended transfer function $f(W, U, Z, L)$ by isolating the conditional extended transfer function $f_N(W, U, Z)$.

Although the augmented state diagram in this form can be used to derive the transfer function of a punctured CBC for any input block size, computational complexity becomes more intensive as the puncturing period M increases, since the number of branches terminating at X_E or, equivalently, the number of terms in the state equation for X_E is proportional to M. However, if we are interested in computing the transfer function for a specific input block length N, we can work out the active CPV when the paths of length N terminate at state X_E, retain only the associated branches ending at that state and remove all others. For example, let us assume that we only consider odd-length input sequences to the punctured rate-2/3 RSC(1,5/7) block code, whose augmented state diagram is depicted in Fig. 3. We observe that odd-length paths terminate at X_E when $\mathbf{P}_1^{\mathrm{c}}$ is active, either through the branch connecting $X_0^{(2)}$ to X_E or the branch connecting $X_1^{(2)}$ to X_E. Hence, we can remove the branches connecting $X_0^{(1)}$ and $X_1^{(1)}$ to X_E, before deriving the state equations. As a result, the terms in the state equation for X_E have been reduced from 4 to 2 in our example, or from $2M$ to 2 in the case of a binary CBC using a puncturing pattern of period M.

In the special case where the length of the input sequence N is an integer multiple of the puncturing period M, i.e., $N = \kappa M$, the augmented state diagram of $(M \times 2^{\nu} + 2)$ states can collapse into a state diagram of $(2^{\nu} + 2)$ states. In particular, only the states in set $\mathbf{Y}_M$ of the original augmented state diagram as well as the start state X_S and the end state X_E need to be considered; each sequence of M consecutive branches originating from a state $X_k^{(M)}$ or terminating at a state $X_k^{(M)}$ can be replaced by a single *composite* branch. Effectively, the original augmented state diagram of a punctured binary CBC collapses to an augmented state diagram of only $(2^{\nu} + 2)$ states, where 2^M composite branches originate from each state. Each composite branch is labeled using a monomial, which is the product of the monomials associated with the constituent branches. If two or more composite branches of the same direction connect two states, a single branch can be used to replace them; in that case, the monomials of all removed composite branches are added together to give a polynomial, which is used to label the new branch. The collapsed augmented state diagram of the rate-2/3 punctured RSC(1,5/7) block code for $M = 2$ and even N is presented in Fig. 4. As an example, the transition sequence $X_3^{(2)} \rightarrow X_1^{(1)} \rightarrow X_2^{(2)}$ in the original augmented state diagram (Fig. 3) has been replaced by a single transition $X_3 \rightarrow X_2$ in the collapsed augmented state diagram (Fig. 4), conveying a codeword given by the product $ZL \cdot L = ZL^2$.

By analogy, a *collapsed trellis diagram* can be derived from the collapsed augmented state diagram. A path of length ℓ in the conventional trellis diagram is represented by a path of length ℓ/M in the collapsed trellis diagram, since M successive branches in the conventional trellis diagram compose a composite branch in the collapsed trellis diagram. We will refer to the collapsed trellis diagram again in the following section, which discusses complexity reduction techniques when long input sequences are considered.

IV. Complexity Considerations for Long Input Blocks

Enumeration of all codeword sequences generated by a convolutional block encoder, and thus derivation of its transfer function $B(W, U, Z)$ based on the augmented state diagram, becomes computational intensive as the length of the input sequence, or equivalently the length of all possible trellis paths, increases. Nevertheless, we can adopt the approach introduced by Benedetto and Montorsi [3], [18] to compute an intermediate transfer function, defined as

$$T(W, U, Z, L, \Omega) = \sum_{w,u,z,\ell,n} T_{w,u,z,\ell,n} W^w U^u Z^z L^\ell \Omega^n. \tag{10}$$

Contrary to the transfer function $B(W, U, Z)$, the intermediate transfer function $T(W, U, Z, L, \Omega)$ of a CBC for input blocks of length N, only enumerates those paths having length $\ell \leq N$, that leave the zero state at step one, re-visit the zero state n times but never stay at it, and terminate at the zero state; note that $T_{w,u,z,\ell,n}$ in (10) denotes the multiplicity of a path having particular weights w, u and z, trellis length ℓ and n remergings with the all-zero sequence. Due to these restrictions, the intermediate transfer function $T(W, U, Z, L, \Omega)$ enumerates a smaller number of codeword sequences compared to the transfer function $B(W, U, Z)$ and hence it is less computational demanding. However, the additional information stored in the indeterminate variables L and Ω of $T(W, U, Z, L, \Omega)$ can be used to fully acquire $B(W, U, Z)$.

Both the augmented state diagram of a nonpunctured CBC and the collapsed augmented state diagram of a punctured CBC can be easily modified to give the less computational intensive intermediate transfer function $T(W, U, Z, L, \Omega)$. First, we need to remove the branch that connects X_S to X_0 since all paths must leave the zero state at step one. The self-loop at X_0, as well as the branch that connects X_0 to X_E, need also to be removed in order to force paths that re-visit the zero state to leave it at the following time step. Finally, the labels of those branches that lead to a remerging into the zero state, either X_0 or X_E, should be updated to include variable Ω, as well. The simplified version of the collapsed augmented state diagram presented in Fig. 4 is depicted in Fig. 5. Upon solving the state equations for the ratio X_E/X_S we obtain

$$\frac{X_E}{X_S} = T(W, U, Z, L, \Omega). \tag{11}$$

A codeword sequence $W^w U^u Z^z L^\ell \Omega^n$ conveyed by $T(W,U,Z,L,\Omega)$ corresponds to a trellis path that remerges n *successive* times with the path of the transmitted all-zero codeword sequence; hence, the total number $K[\ell,n]$ of codeword sequences having path length N with zeroes before, after or between the n remergings can be obtained from a single codeword sequence of path length $\ell \leq N$ associated with n successive remergings, as follows [3]

$$K[\ell,n] = \binom{N-\ell+n}{n} = \frac{(N-\ell+n)!}{(N-\ell)!n!}. \tag{12}$$

The above expression can be directly used when the intermediate transfer function of a nonpunctured CBC has been obtained from the simplified augmented state diagram. However, it can be modified to also encompass punctured CBCs, provided that the path length of the generated codeword sequences is an integer multiple of the puncturing period, that is $N = \kappa M$; in that case the simplified collapsed augmented state diagram can be used to derive the intermediate transfer function $T(W,U,Z,L,\Omega)$ of the punctured CBC. Let us assume that $W^w U^u Z^z L^\ell \Omega^n$ is now a codeword sequence generated by the punctured CBC in question. Here, n refers to the number of successive remergings of the corresponding path with the all-zero sequence in the collapsed trellis diagram. Moreover, ℓ represents the length of the path in the conventional trellis diagram, where $\ell \leq N$; the equivalent length in the collapsed trellis diagram is ℓ/M. Having depicted the sequence $W^w U^u Z^z L^\ell \Omega^n$ as a path in the collapsed trellis diagram, we can now use (12) to obtain the number of codeword sequences having path length N/M also in the collapsed trellis diagram. In particular, we obtain

$$K[\ell,n] = \binom{\frac{N-\ell}{M}+n}{n} = \frac{\left(\frac{N-\ell}{M}+n\right)!}{\left(\frac{N-\ell}{M}\right)!n!}, \tag{13}$$

which is a variant of (12) for punctured CBCs.

If $T(W,U,Z,L,\Omega)$ has revealed that there are $T_{w,u,z,\ell,n}$ codeword sequences of length $\ell \leq N$, then the total number of codeword sequences of length *exactly* equal to N with zeroes before, after or between the n remergings is given by the product $K[\ell,n] \cdot T_{w,u,z,\ell,n}$. Therefore, the transfer function $B(W,U,Z)$ of a CBC can be computed by substituting the total number of codeword sequences $B_{w,u,z}$, obtained by [3]

$$B_{w,u,z} = \sum_{\ell,n} K[\ell,n] T_{w,u,z,\ell,n}, \tag{14}$$

into

$$B(W,U,Z) = \sum_{w,u,z} B_{w,u,z} W^w U^u Z^z, \tag{15}$$

where each codeword sequence $W^w U^u Z^z$ has overall path length N.

In this section we used the approach presented in [3], [18] to accelerate computation of the transfer function $B(W,U,Z)$ by simplifying the augmented state diagram of a CBC, both punctured and nonpunctured. In the case of punctured CBCs, simplification can only take place when the length of the input information sequence N is an integer multiple of the puncturing period M; in that case, we can reduce the original augmented state diagram to its collapsed version and then simplify it so as to obtain the desired intermediate transfer function. Finally, we must emphasize that (14) is applicable only when no loops of zero output weight remain in the simplified augmented state diagram. If such loops do exist, (14) will only compute a fraction of the total number of codeword sequences having path length N. Nevertheless, note that state diagrams having loops of zero output weight correspond to catastrophic codes; such codes should be avoided since a finite number of transmission errors can cause an infinite number of errors in the decoded information sequence [19].

V. Application to Turbo Codes on AWGN Channels

A. *Turbo Codes*

A turbo code $\mathcal{P}$ is the parallel concatenation of convolutional codes (PCCC) separated by random interleavers. However, the most common configuration uses two RSC codes of memory size ν each, separated by a random interleaver of size N [1]. An information sequence of length N is input to both the first constituent systematic encoder $\mathcal{C}_1$ of rate 1/2 and the random interleaver. The interleaved information sequence is then input to the second convolutional encoder $\mathcal{C}_2$ of rate 1. The output of the rate 1/3 turbo encoder consists of the systematic and parity check sequences of $\mathcal{C}_1$ and the parity check sequence of $\mathcal{C}_2$. Rates higher than 1/3 can be obtained by puncturing the three outputs of a parent rate-1/3 turbo encoder using a $3 \times M$ pattern of the form

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_1^r \\ \mathbf{P}_2^r \\ \mathbf{P}_3^r \end{bmatrix} = \begin{bmatrix} p_{1,1} & \cdots & p_{1,M} \\ p_{2,1} & \cdots & p_{2,M} \\ p_{3,1} & \cdots & p_{3,M} \end{bmatrix}. \tag{16}$$

Note that punctured turbo codes are classified as systematic, partially systematic or nonsystematic depending on whether all, some or none of their systematic bits are transmitted [12]. It was shown in [3] and [4] that the transfer function $B^{\mathcal{P}}(W,U,Z)$ of a turbo code, punctured or nonpunctured, can be obtained using the transfer functions of its constituent codes, namely $B^{\mathcal{C}_1}(W,U,Z)$ and $B^{\mathcal{C}_2}(W,U,Z)$. Below, we briefly describe the steps required to obtain $B^{\mathcal{P}}(W,U,Z)$.

Let us use $\mathcal{C}$ to refer to one of the constituent codes, either $\mathcal{C}_1$ or $\mathcal{C}_2$; the transfer function $B^{\mathcal{C}}(W,U,Z)$ of the constituent code, which can be obtained using our method based on the concept of the augmented state diagram, assumes the form given in (15). However, (15) can be rewritten as

$$B^{\mathcal{C}}(W,U,Z) = \sum_{w} B_w^{\mathcal{C}}(U,Z)W^w, \tag{17}$$

where

$$B_w^{\mathcal{C}}(U,Z) = \sum_{u,z} B_{w,u,z}^{\mathcal{C}} U^u Z^z \tag{18}$$

is the so-called conditional weight enumerating function (CWEF) and provides all codeword sequences of specific input weight w. A relationship between the CWEF of a turbo code and the CWEFs of the constituent codes can be easily derived only if we assume the use of a uniform interleaver of size N, an abstract probabilistic concept introduced in [3]. More specifically, if $B_w^{\mathcal{C}_1}(U,Z)$ and $B_w^{\mathcal{C}_2}(U,Z)$ are the CWEFs of the constituent terminated RSC block codes, the CWEF of the terminated turbo code, $B_w^{\mathcal{P}}(U,Z)$, is given by [3], [5]

$$B_w^{\mathcal{P}}(U,Z) = \frac{B_w^{\mathcal{C}_1}(U,Z)\cdot B_w^{\mathcal{C}_2}(U=1,Z)}{\binom{N}{w}}, \quad \text{for every value of } w. \tag{19}$$

Recall that the systematic output sequence of the second constituent encoder is not transmitted, hence it is eliminated by setting $U=1$ in $B_w^{\mathcal{C}_2}(U,Z)$. The transfer function of the turbo code $B^{\mathcal{P}}(W,U,Z)$ can be computed from the CWEF, $B_w^{\mathcal{P}}(U,Z)$, in a manner identical to (17). Therefore, first we can use our proposed technique to derive the transfer function of each constituent terminated CBC for input sequences of length N and then compute the transfer function of the corresponding terminated turbo code for a uniform interleaver of size N.

Using the union bound argument [20], the bit error probability (BEP), denoted as P_b, of a turbo code for maximum likelihood (ML) soft decoding on an AWGN channel can be upper bounded as follows

$$P_b \leq P_b^u = \sum_{w,d} \frac{w}{N} B_{w,d}^{\mathcal{P}} Q\left(\sqrt{\frac{2E_b R_{\mathcal{P}}}{N_0} d}\right), \tag{20}$$

where P_b^u is the union bound on the BEP, E_b is the energy per information bit, N_0 is the noise power spectral density and $R_{\mathcal{P}}$ is the code rate. The sum runs over all valid values of input weight w and overall output weight $u+z=d$. The coefficients $B_{w,d}^{\mathcal{P}}$ can be obtained from the coefficients $B_{w,u,z}^{\mathcal{P}}$ of the transfer function $B^{\mathcal{P}}(W,U,Z)$ using

$$B_{w,d}^{\mathcal{P}} = \sum_{\substack{u,z \\ u+z=d}} B_{w,u,z}^{\mathcal{P}}. \tag{21}$$

The $Q(\cdot)$ function is defined as

$$Q(\xi) = \frac{1}{\sqrt{2\pi}} \int_{\xi}^{\infty} e^{-r^2/2} dr. \tag{22}$$

A complete overview of bounding techniques for block and turbo codes has been presented in [21]. In this monograph, the authors explain that union bounds are accurate only at high signal-to-noise ratio (SNR) values but their weakness is pronounced at the low SNR regime. Improved upper bounds, which are derived by properly defining the region around the transmitted codewords, are considerably tighter than the union bound at low SNR values. However, computation of both the standard union bound and the improved upper bounds relies on the transfer function of the examined code. For our work, we elected to use the union bound argument because of its simplicity; nevertheless, once the transfer function of the examined code has been obtained using our proposed approach, expressions that provide more tight upper bounds could be used.

In the following two subsections, we concentrate on terminated punctured turbo codes and we demonstrate that we can use our method to obtain union bounds on their BEP as well as to identify puncturing patterns that lead to high-rate turbo codes yielding low error floors.

B. Evaluation of Performance Upper Bounds

Having obtained the transfer function $B^{\mathcal{P}}(W,U,Z)$ of a terminated turbo code $\mathcal{P}$, we can compute the coefficients $B_{w,d}^{\mathcal{P}}$ and hence an upper bound on the BEP for ML soft decoding, using (21) and (20), respectively, as we have previously explained. However, it is not always convenient to provide the two-dimensional coefficients $B_{w,d}^{\mathcal{P}}$ of turbo codes for a range of w and d values; instead, we use an alternative but equivalent single dimensional representation, which only depends on the output weight d. More specifically, we define a single-dimensional coefficient D_d as [3]

$$D_d = \sum_w \frac{w}{N} B^{\mathcal{P}}_{w,d}, \tag{23}$$

consequently, (20) assumes the form

$$P_b \leq P^u_b = \sum_d D_d Q\left(\sqrt{\frac{2E_b R_{\mathcal{P}}}{N_0} d}\right). \tag{24}$$

The first 15 coefficients D_d, which were derived based on our proposed method for various rate-1/2 systematic turbo codes, are shown in Table I. Two parent rate-1/3 turbo codes are considered; the first PCCC consists of two RSC(1,5/7) encoders and, using the notation of RSC codes, it is denoted as PCCC(1,5/7,5/7) for brevity; the second turbo code is fully described by the generator polynomials PCCC(1,17/15,17/15). Coefficients for three interleaver sizes have been computed, namely 100, 1,000 and 10,000 bits. In all cases, the RPVs of the puncturing pattern are $\mathbf{P}^r_1 = [1 \quad 1]$, $\mathbf{P}^r_2 = [1 \quad 0]$ and $\mathbf{P}^r_3 = [0 \quad 1]$; this particular pattern is often used [1], [22] to generate rate-1/2 systematic turbo codes from parent rate-1/3 codes.

Theoretical upper bounds on the average ML decoding performance and simulation results for the rate-1/2 systematic PCCCs of Table I, are presented in Fig. 6 and Fig. 7. The performance of suboptimal iterative decoders employing the exact-log maximum a-posteriori (MAP) decoding algorithm [23] after 8 iterations is compared with the corresponding union bounds. As expected, the performance of systematic punctured turbo codes quickly converges to the error floor region [1], [22], identified by the union bound on the average ML decoding performance of the corresponding turbo code for high E_b/N_0 values [3], provided that puncturing is distributed equally between parity check bits and is well scattered [15], as in our case.

We have thus demonstrated that our technique can be used to derive the transfer function of a terminated turbo code and hence obtain a tight upper bound on its BEP for ML decoding, which coincides with the error floor of the code. Note that the error floor of a turbo code can be lowered if the operation mode of the constituent encoders switches from terminated coding to *continuous coding* [24]; in continuous coding, both constituent CBCs maintain the states they are in when encoding of an input sequence is completed and start from those states when encoding of the next input sequence begins. The transfer function of a turbo code operating in continuous mode can only be obtained by using the notion of the *hyper-trellis*, as described in [3], [24]. Our method can be used to derive the labels of the hyper-trellis branches, provided that the start states and end states of the augmented state diagrams of both constituent codes correspond to the hyper-states of each branch.

In this paper we have not considered continuous coding because its performance is almost identical to that of terminated coding when constituent CBCs of small memory size ($\nu = 2$ or 3 in our simulations) and interleavers of size larger than ten times that of the memory size are studied [3]. The performance advantage of continuous over terminated coding only becomes significant when the memory size of the constituent codes is $\nu = 5$ or higher [24]. Nevertheless, future work could investigate the effect of continuous coding on the performance of punctured turbo codes that use constituent codes of large memory size.

C. Identification of Good Puncturing Patterns

A different application of our technique is the identification of "good" puncturing patterns, i.e., patterns that generate punctured turbo codes exhibiting low error floors. In particular, we study the evolution of the coefficients $B_{w,d}$, obtained using our method, as the interleaver size of a turbo code gradually increases. We then extrapolate our conclusions to turbo codes using long interleavers.

Let us assume that our objective is to identify good patterns of period M that increase the rate of a parent turbo code from 1/3 to $R_{\mathcal{P}}$. Initially, we set the size of the interleaver to a small value ($N < 100$) and we compute the transfer function of the punctured turbo code of rate $R_{\mathcal{P}}$ for all valid patterns of period M. Next, we use (20) to derive the required E_b/N_0 ratio for a targeted P_b^u, for each possible pattern configuration. We also evaluate the free effective distance, d_{eff}, which conveys the minimum output weight of a codeword sequence for an input sequence of weight two; thus, the smallest value of d for which a coefficient $B_{2,d}$ is non-zero corresponds to d_{eff}. As was demonstrated in [5], [25] and [26], the free effective distance has a major impact on the performance of a turbo code, when long interleavers are employed.

When all valid patterns are exhausted, we group them according to the free effective distance that the corresponding turbo codes have achieved; each group contains patterns that lead to punctured turbo codes yielding the same d_{eff}, arranged in ascending order of E_b/N_0 for a particular P_b^u. We then increase the size of the interleaver by a small value and we repeat the same computations, grouping and ordering. If a subsequent increase in the interleaver size does not change the ordering of the puncturing patterns, we conclude our search and we use that ordering to draw conclusions for punctured turbo codes using long interleavers.

In Table II, we have listed seven puncturing patterns, which we have selected among all possible patterns of period $M=4$. A punctured PCCC obtained by any of these patterns achieves the maximum free effective distance and requires the lowest E_b/N_0 value for $P_b^u=10^{-6}$, among all PCCCs using patterns of the same puncturing distribution between systematic and parity check bits, when an interleaver size of 36 bits is considered. Four of the patterns lead to rate-1/2 turbo codes, whilst the remaining three give rate-2/3 turbo codes. Furthermore, two of the resultant punctured PCCCs are systematic (Sys) while the other five are partially systematic (PS) with a decreasing number of transmitted systematic bits. We observe that for $N=36$, the union bound on the BEP of the PS turbo codes reaches a value of 10^{-6} for a lower E_b/N_0 ratio than that of the systematic turbo codes. We also observe that PS turbo codes with a reduced number of transmitted systematic bits achieve a high free effective distance. Hence, we would expect that PS turbo codes using long interleavers achieve a lower error floor than that of systematic turbo codes of the same rate, especially those codes with few transmitted systematic bits.

Indeed, we observe in Fig. 8 that the conclusions drawn from Table II concur with simulation results of punctured turbo codes using interleavers of size $N = 1,000$. As expected, PS turbo codes achieve a lower error floor than systematic turbo codes of the same rate; in particular, as the number of transmitted systematic bits is reduced, the error floor of a suboptimal iterative decoder is lowered at the expense of an expanded non-convergence region. Hence, in accordance with the findings of Land and Hoeher [12], it can be more advantageous to put more puncturing to the systematic sequence than to the parity check sequences; however, our results have also confirmed the observations by Blazek *et al.* [13] and Crozier *et al.* [14], who claimed that the performance benefits of PS turbo codes are mainly for higher E_b/N_0 values and number of iterations.

We thus demonstrated that our proposed method can be used to quickly identify puncturing patterns that lead to high-rate turbo codes exhibiting low error floors. Nevertheless, puncturing affects the convergence behavior of iterative decoding [27]; hence, convergence towards the error floor region should also be investigated using techniques such as the extrinsic information transfer (EXIT) chart analysis, proposed by ten Brink in [28].

VI. Conclusion

In this paper, we have introduced the augmented state diagram based on which the transfer function of a CBC, either punctured or nonpunctured, can be evaluated. We have also addressed the complexity of the approach by showing that a simplification process can take place allowing us to compute the transfer function of a CBC for long input sequences. A tight upper bound on the average ML performance of a turbo code, which uses CBCs as constituent codes, can be then computed to accurately predict the error floor of the suboptimal iterative decoder.

Our analysis validated the observations of existing literature, which studied the relationship between the suboptimal performance of iterative decoding and the error floor region of punctured turbo codes. More specifically, partially systematic turbo codes achieve a lower error floor than systematic turbo codes of the same rate, at the expense of a decelerated convergence of their performance towards the ML bound. Recent results [27], [29] have shown that partially systematic turbo codes that yield error floors even lower than those of their parent rate-1/3 turbo codes can be identified using our proposed method.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions that helped improve this manuscript.

Appendix

Enumeration of monomials containing L^N

The extended transfer function $f(W,U,Z,L)$ can be expressed as [19]

$$f(W,U,Z,L) = \frac{\mathcal{Y}(W,U,Z,L)}{1-\mathcal{X}(W,U,Z,L)}, \tag{25}$$

owing to the inherent loops of the augmented state diagram of a CBC, where $\mathcal{Y}(W,U,Z,L)$ and $\mathcal{X}(W,U,Z,L)$ are polynomials of W, U, Z and L. If we group together all monomials containing the indeterminate variable L raised to the same power, we can write $\mathcal{Y}(W,U,Z,L)$ and $\mathcal{X}(W,U,Z,L)$ as

$$\mathcal{Y}(W,U,Z,L) = \sum_{\ell=\ell_1}^{\ell_\psi} y_\ell(W,U,Z)L^\ell, \quad \mathcal{X}(W,U,Z,L) = \sum_{\ell=\ell'_1}^{\ell'_\chi} x_\ell(W,U,Z)L^\ell, \tag{26}$$

where $y_\ell(W,U,Z)$ and $x_\ell(W,U,Z)$ are polynomials of W, U, Z whilst ℓ is a nonnegative integer that takes values from sets $\{\ell_1,\ldots,\ell_\psi\}$ and $\{\ell'_1,\ldots,\ell'_\chi\}$, respectively; note that the elements of each set are arranged in increasing order. For the sake of clarity, we drop the indeterminate variables W, U and Z, since they are not of importance for the remainder of this section. For example, we adopt the notation $\mathcal{Y}(L)$ and y_ℓ instead of $\mathcal{Y}(W,U,Z,L)$ and $y_\ell(W,U,Z)$, respectively.

Exploiting the property of binomial series, we can rewrite (25) as

$$\begin{aligned} f(L) &= \mathcal{Y}(L)\cdot\sum_{k=0}^{\infty}\mathcal{X}^k(L) \\ &= \left(y_{\ell_1}L^{\ell_1}+\ldots+y_{\ell_\psi}L^{\ell_\psi}\right)\cdot\sum_{k=0}^{\infty}\mathcal{X}^k(L). \end{aligned} \tag{27}$$

The terms of $\mathcal{X}^k(L)$ for an arbitrary power k can be computed using the multinomial formula

$$\begin{aligned} \mathcal{X}^k(L) &= \left(\sum_{\ell=\ell'_1}^{\ell'_\chi} x_\ell L^\ell\right)^k \\ &= \sum_{\substack{k_1,\ldots,k_\chi \\ k_1+\ldots+k_\chi=k}} \left[\frac{k!}{k_1!\ldots k_\chi!}\left(x_{\ell'_1}L^{\ell'_1}\right)^{k_1}\cdot\ldots\cdot\left(x_{\ell'_\chi}L^{\ell'_\chi}\right)^{k_\chi}\right] \\ &= \sum_{\substack{k_1,\ldots,k_\chi \\ k_1+\ldots+k_\chi=k}} \left[\frac{k!}{k_1!\ldots k_\chi!}\left(x_{\ell'_1}^{k_1}\cdot\ldots\cdot x_{\ell'_\chi}^{k_\chi}\right)\cdot L^{(\ell'_1k_1+\ldots+\ell'_\chi k_\chi)}\right], \end{aligned} \tag{28}$$

where the sum is taken over all nonnegative integers k_1,. . .,k_χ for which $k_1+\ldots+k_\chi=k$.

If we are only interested in enumerating paths of length N, represented by monomials in $f(L)$ containing the term L^N, a process based on (27) and (28) could be followed. In particular:

1) We select a value for ℓ from the ordered set $\{\ell_1,\ldots,\ell_\psi\}$ provided that it is $\ell\le N$. Initially we set $\ell=\ell_1$, therefore we only consider the product $y_{\ell_1}L^{\ell_1}$ in $\mathcal{Y}(L)$.
2) We focus on $\mathcal{X}^k(L)$ in (27), where k takes such values that $\mathcal{X}^k(L)$ contains monomials having L raised to the power of $(N-\ell_1)$. Consequently, the product in (27) will result in monomials containing L^N.
3) Based on (28), we derive the desired values of k by solving the Diophantine equation [30]

$$\ell'_1k_1+\ell'_2k_2+\ldots+\ell'_\chi k_\chi = N-\ell_1 \tag{29}$$

for k_1, k_2, ..., k_χ. We then add the elements of each set $\{k_1, k_2, \ldots, k_\chi\}$ of nonnegative solutions in order to obtain the desired values of k, that is

$$k = k_1 + k_2 + \ldots + k_\chi, \quad \text{for each set of solutions } \{k_1, k_2, \ldots, k_\chi\}. \tag{30}$$

4) We substitute each set $\{k_1, k_2, \ldots, k_\chi\}$ as well as the corresponding sum k into (28) so as to obtain only those monomials that contain L raised to the power of $(N - \ell_1)$.
5) We complete the process by multiplying the sum of monomials in $\mathcal{X}(L)$ with $y_{\ell_1} L^{\ell_1}$.

The same process is repeated until all values which are less than or equal to N from the set $\{\ell_1, \ldots, \ell_\psi\}$ are assigned to ℓ.

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE International Conference on Communications (ICC'93)*, Geneva, Switzerland, May 1993, pp. 1064–1070.

[2] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo codes," *IEEE Trans. Commun.*, vol. 44, no. 2, pp. 1261–1271, Oct. 1996.

[3] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 409–429, Mar. 1996.

[4] D. Divsalar, S. Dolinar, R. J. McEliece, and F. Pollara, "Transfer function bounds on the performance of turbo codes," JPL, Cal. Tech., TDA Progr. Rep. 42-121, Aug. 1995.

[5] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Trans. Commun.*, vol. 44, no. 5, pp. 591–600, May 1996.

[6] J. Hagenauer, "Rate compatible punctured convolutional codes and their applications," *IEEE Trans. Commun.*, vol. 36, no. 4, pp. 389–400, Apr. 1988.

[7] D. Haccoun and G. Bégin, "High-rate punctured convolutional codes for Viterbi and sequential decoding," *IEEE Trans. Commun.*, vol. 37, no. 11, pp. 1113–1125, Nov. 1989.

[8] A. S. Barbulescu and S. S. Pietrobon, "Rate compatible turbo codes," *Electronics Letters*, vol. 31, no. 7, pp. 535–536, Mar. 1995.

[9] M. Fan, S. C. Kwatra, and K. Junghwan, "Analysis of puncturing pattern for high rate turbo codes," in *Proc. Military Comm. Conf. (MILCOM'99)*, New Jersey, USA, Oct. 1999, pp. 547–550.

[10] Ö. Açikel and W. E. Ryan, "Punctured turbo-codes for BPSK/QPSK channels," *IEEE Trans. Commun.*, vol. 47, no. 9, pp. 1315–1323, Sep. 1999.

[11] F. Babich, G. Montorsi, and F. Vatta, "Design of rate-compatible punctured turbo (RCPT) codes," in *Proc. Int. Conf. Comm. (ICC'02)*, New York, USA, Apr. 2002, pp. 1701–1705.

[12] I. Land and P. Hoeher, "Partially systematic rate 1/2 turbo codes," in *Proc. Int. Symp. Turbo Codes*, Brest, France, Sep. 2000, pp. 287–290.

[13] Z. Blazek, V. K. Bhargava, and T. A. Gulliver, "Some results on partially systematic turbo codes," in *Proc. Vehicular Tech. Conf. (VTC-Fall'02)*, Vancouver, Canada, Sep. 2002, pp. 981–984.

[14] S. Crozier, P. Guinand, and A. Hunt, "On designing turbo-codes with data puncturing," in *Proc. Canadian Workshop on Inf. Theory*, Montreal, Canada, 2005.

[15] M. A. Kousa and A. H. Mugaibel, "Puncturing effects on turbo codes," *Proc. IEE Comm.*, vol. 149, no. 3, pp. 132–138, Jun. 2002.

[16] D. Divsalar and F. Pollara, "Turbo codes for PCS applications," in *Proc. IEEE International Conference on Communications (ICC'95)*, Seattle, WA, USA, Jun. 1995.

[17] I. Chatzigeorgiou, M. R. D. Rodrigues, I. J. Wassell, and R. Carrasco, "A novel technique for the evaluation of the transfer function of parallel concatenated convolutional codes," in *Proc. Int. Symp. Turbo Codes*, Munich, Germany, Apr. 2006.

[18] S. Benedetto, M. Mondin, and G. Montorsi, "Performance evaluation of trellis-coded modulation schemes," *Proc. IEEE*, vol. 82, no. 6, pp. 833–855, Jun. 1994.

[19] M. Bossert, *Channel Coding for Telecommunications*. John Wiley & Sons, 1999.

[20] W. E. Ryan, "Concatenated convolutional codes and iterative decoding," in *Wiley Encyclopedia on Telecommunications*, J. G. Proakis, Ed. Hoboken, New Jersey: Wiley-Interscience, 2003, pp. 556–570.

[21] I. Sason and S. Shamai, *Performance Analysis of Linear Codes under Maximum-Likelihood Decoding: A Tutorial*. Delft, the Netherlands: Foundations and Trends in Communications and Information Theory, NOW Publishers, 2006.

[22] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 429–445, Mar. 1996.

[23] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimising symbol error rate," *IEEE Trans. Inf. Theory*, vol. IT-20, pp. 284–287, Mar. 1974.

[24] S. Benedetto and G. Montorsi, "Performance of continuous and blockwise decoded turbo codes," *IEEE Commun. Lett.*, vol. 1, no. 3, pp. 77–79, May 1997.

[25] D. Divsalar and R. J. McEliece, "Effective free distance of turbo codes," *Electronics Letters*, vol. 32, no. 5, pp. 445–446, Feb. 1996.

[26] L. C. Perez, J. Seghers, and D. J. Costello, "A distance spectrum interpretation of turbo codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 1698–1709, Nov. 1996.

[27] I. Chatzigeorgiou, M. R. D. Rodrigues, I. J. Wassell, and R. Carrasco, "Can punctured rate-1/2 turbo codes achieve a lower error floor than their rate-1/3 parent codes?" in *Proc. IEEE Information Theory Workshop (ITW'06)*, Chengdu, China, Oct. 2006.

[28] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.

[29] I. Chatzigeorgiou, M. R. D. Rodrigues, I. J. Wassell, and R. Carrasco, "Pseudo-random puncturing: A technique to lower the error floor of turbo codes," in *Proc. IEEE International Symposium on Information Theory (ISIT'07)*, Nice, France, Jun. 2007.

[30] I. G. Bashmakova, *Diophantus and Diophantine Equations*. Washington DC: The Mathematical Association of America, 1998.

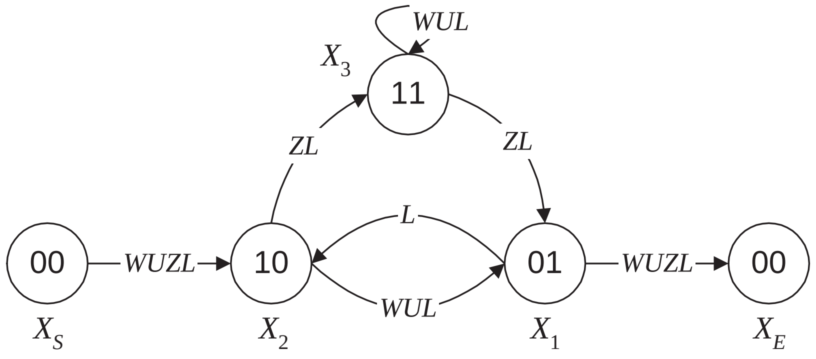

Fig. 1: Weight-labeled state diagram of the rate-1/2 RSC(1,5/7) code.

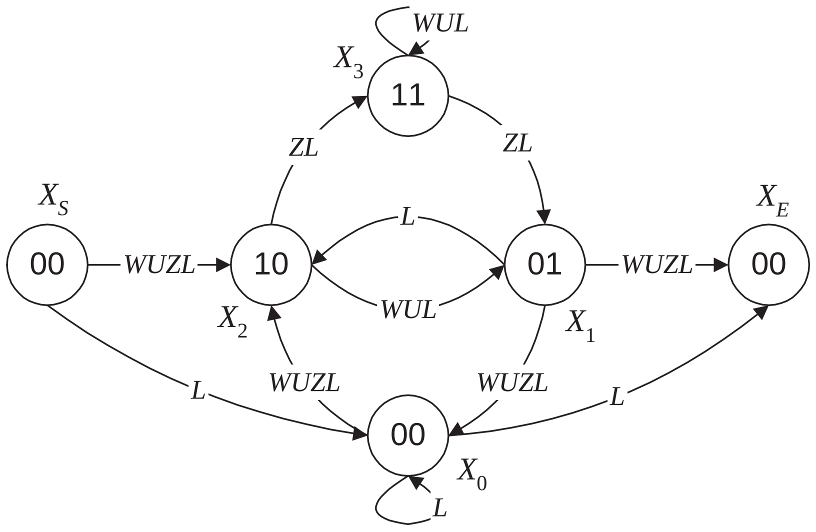

Fig. 2: Augmented state diagram of the nonpunctured rate-1/2 RSC(1,5/7) block code.

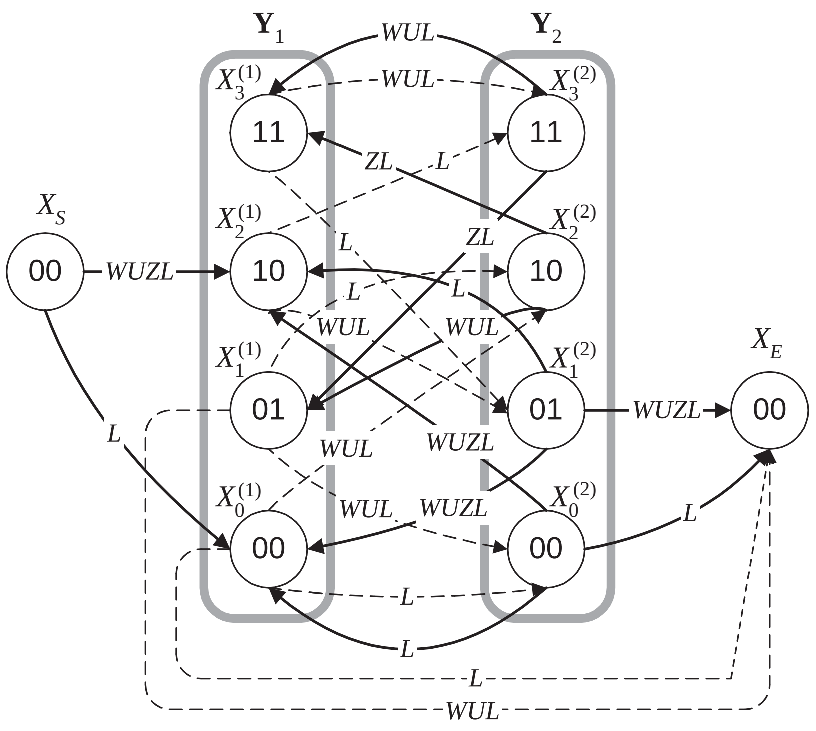

Fig. 3: Augmented state diagram of the punctured rate-2/3 RSC(1,5/7) block code.

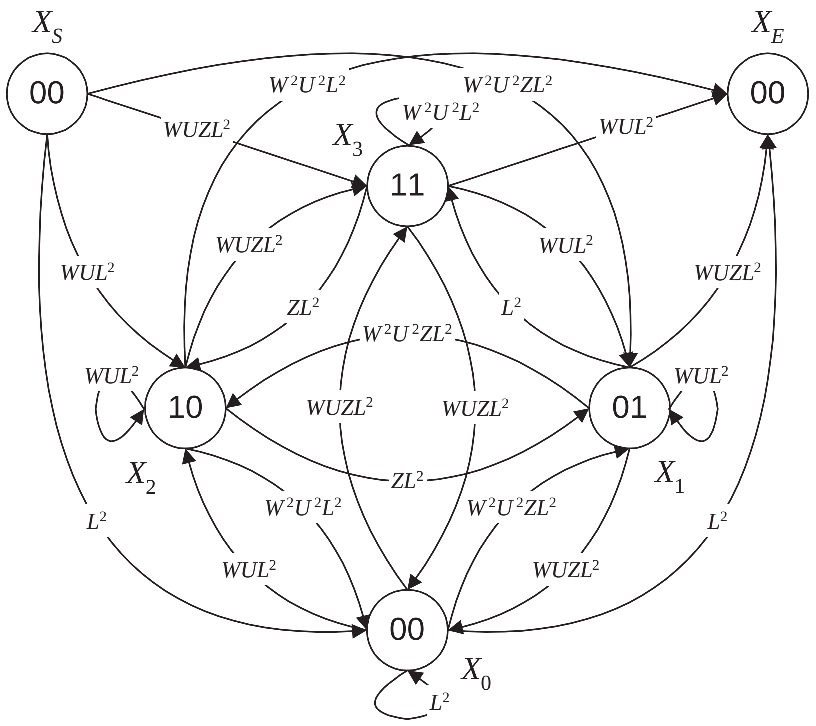

Fig. 4: Collapsed augmented state diagram of the punctured rate-2/3 RSC(1,5/7) block code for even-length input blocks.

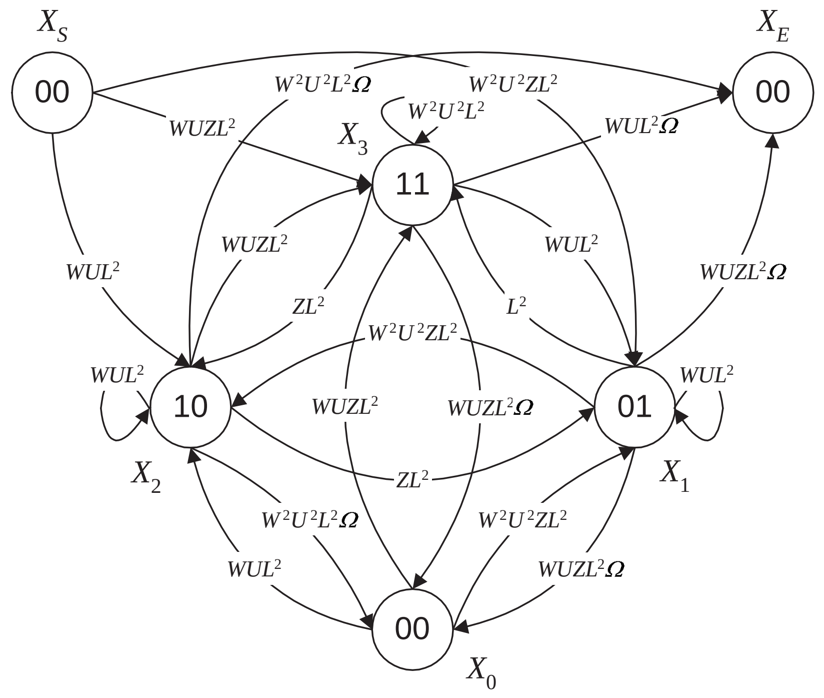

Fig. 5: Simplification of the collapsed augmented state diagram of Fig.4.

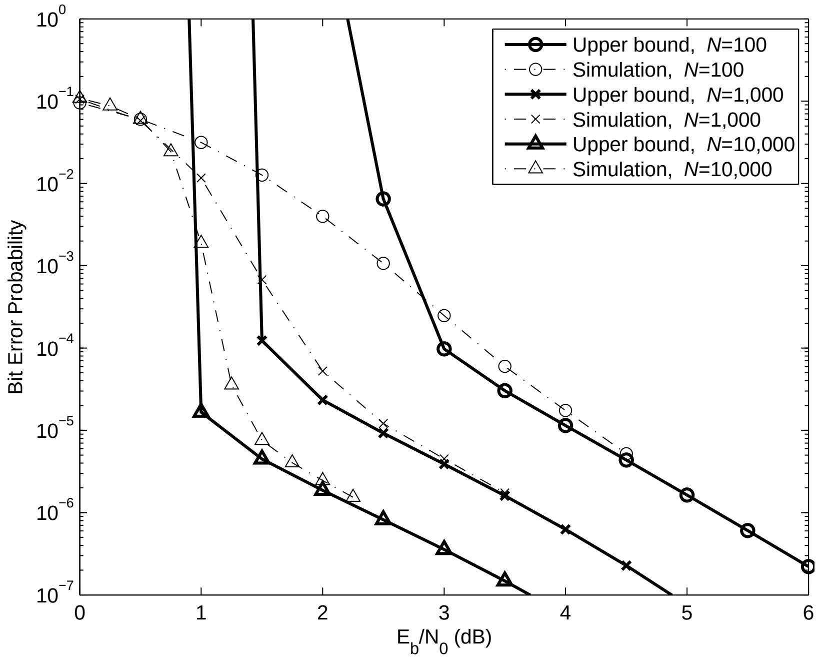

Fig. 6: Comparison between bounds and simulation results of rate-1/2 systematic PCCC(1,5/7,5/7) configurations after 8 iterations.

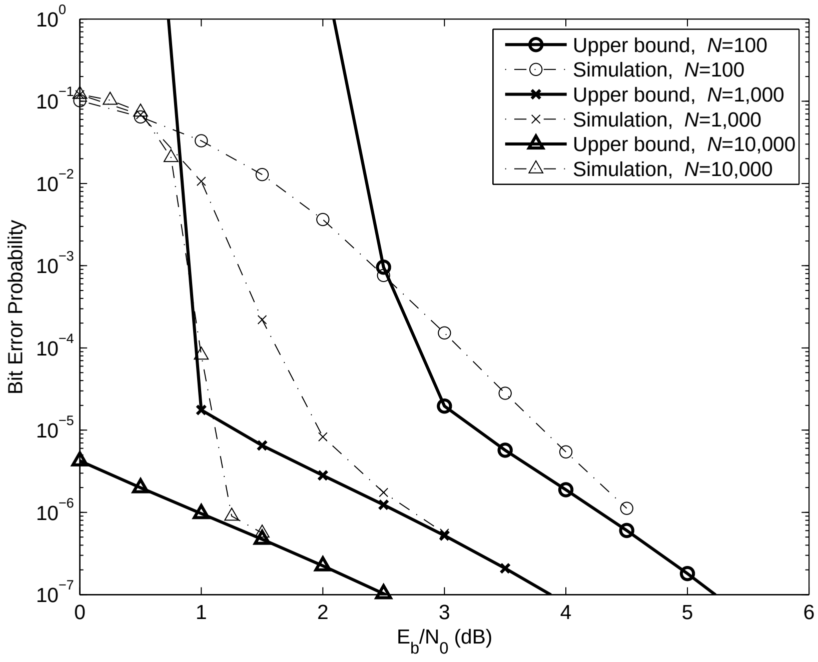

Fig. 7: Comparison between bounds and simulation results of rate-1/2 systematic PCCC(1,17/15,17/15) configurations after 8 iterations.

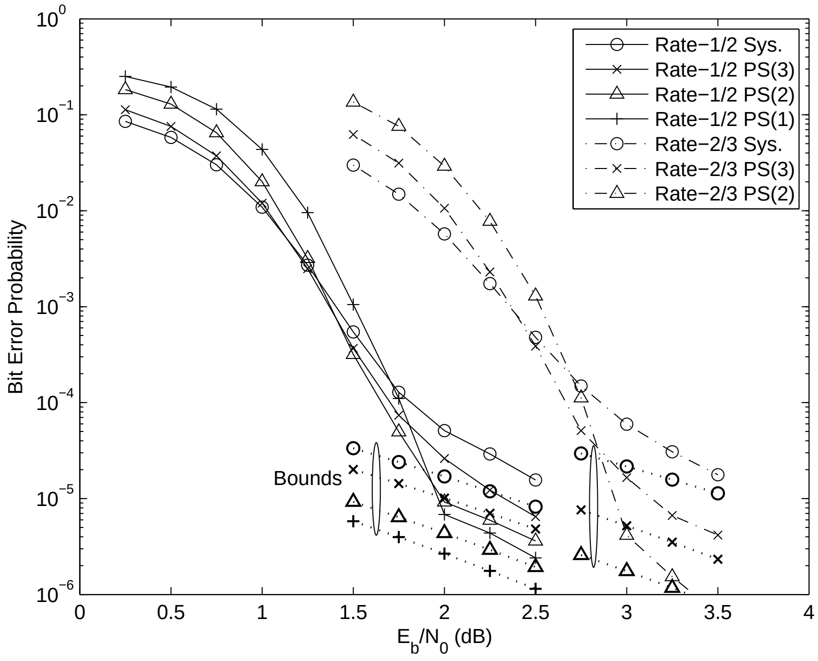

Fig. 8: Simulation results after 10 iterations for various punctured rate-1/2 and rate-2/3 PCCC configurations. An interleaver size of 1,000 bits is used.

TABLE I: Coefficients D_d for the calculation of the union bound on the BEP of two systematic rate-1/2 PCCCs. Three interleaver sizes are considered.

	PCCC(1,5/7,5/7)			PCCC(1,17/15,17/15)		
d	$N = 100$	$N = 1,000$	$N = 10,000$	$N = 100$	$N = 1,000$	$N = 10,000$
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	$4.45 \cdot 10^{-4}$	$4.49 \cdot 10^{-6}$	$4.49 \cdot 10^{-8}$	0	0	0
4	$2.35 \cdot 10^{-5}$	$2.39 \cdot 10^{-8}$	$2.39 \cdot 10^{-11}$	$9.20 \cdot 10^{-5}$	$9.56 \cdot 10^{-8}$	$9.59 \cdot 10^{-11}$
5	$1.11 \cdot 10^{-2}$	$1.16 \cdot 10^{-4}$	$1.16 \cdot 10^{-6}$	0	0	0
6	$8.54 \cdot 10^{-2}$	$8.93 \cdot 10^{-3}$	$8.99 \cdot 10^{-4}$	$1.13 \cdot 10^{-2}$	$9.91 \cdot 10^{-4}$	$9.98 \cdot 10^{-5}$
7	$9.23 \cdot 10^{-2}$	$1.00 \cdot 10^{-3}$	$1.01 \cdot 10^{-5}$	$4.84 \cdot 10^{-2}$	$5.38 \cdot 10^{-4}$	$5.43 \cdot 10^{-6}$
8	0.257	$2.38 \cdot 10^{-2}$	$2.39 \cdot 10^{-3}$	$6.18 \cdot 10^{-2}$	$3.98 \cdot 10^{-3}$	$3.99 \cdot 10^{-4}$
9	0.394	$3.99 \cdot 10^{-3}$	$4.00 \cdot 10^{-5}$	0.143	$1.55 \cdot 10^{-3}$	$1.58 \cdot 10^{-5}$
10	0.995	$4.36 \cdot 10^{-2}$	$4.03 \cdot 10^{-3}$	0.259	$8.43 \cdot 10^{-3}$	$8.02 \cdot 10^{-4}$
11	2.303	$1.92 \cdot 10^{-2}$	$1.89 \cdot 10^{-4}$	0.606	$4.58 \cdot 10^{-3}$	$4.52 \cdot 10^{-5}$
12	6.178	0.199	$1.795 \cdot 10^{-2}$	1.247	$1.68 \cdot 10^{-2}$	$1.37 \cdot 10^{-3}$
13	15.471	0.121	$1.198 \cdot 10^{-3}$	3.134	$1.44 \cdot 10^{-2}$	$1.34 \cdot 10^{-4}$
14	37.189	0.820	$7.280 \cdot 10^{-2}$	7.591	$4.31 \cdot 10^{-2}$	$2.91 \cdot 10^{-3}$
15	91.838	0.582	$5.653 \cdot 10^{-3}$	18.195	$5.37 \cdot 10^{-2}$	$4.85 \cdot 10^{-4}$

TABLE II: Rate-1/2 and 2/3 configurations for PCCC(1,5/7,5/7). The interleaver size N is 36 bits.

Configuration	$\mathbf{P}_1^r$	$\mathbf{P}_2^r$	$\mathbf{P}_3^r$	d_{eff}	E_b/N_0 for $P_b^u = 10^{-6}$
Rate-1/2 Sys.	$[1\,1\,1\,1]$	$[1\,0\,1\,0]$	$[0\,1\,0\,1]$	6	6.438 dB
Rate-1/2 PS(3)	$[1\,1\,1\,0]$	$[1\,0\,1\,1]$	$[0\,1\,1\,0]$	6	6.243 dB
Rate-1/2 PS(2)	$[0\,1\,0\,1]$	$[1\,1\,1\,0]$	$[1\,1\,0\,1]$	7	6.291 dB
Rate-1/2 PS(1)	$[0\,0\,1\,0]$	$[1\,1\,0\,1]$	$[1\,1\,1\,1]$	7	5.959 dB
Rate-2/3 Sys.	$[1\,1\,1\,1]$	$[1\,0\,0\,0]$	$[0\,0\,1\,0]$	2	7.313 dB
Rate-2/3 PS(3)	$[1\,0\,1\,1]$	$[0\,1\,0\,0]$	$[0\,1\,1\,0]$	4	7.168 dB
Rate-2/3 PS(2)	$[1\,1\,0\,0]$	$[0\,0\,1\,1]$	$[0\,1\,1\,0]$	4	6.786 dB