Design and Performance Analysis of a New Class of Rate Compatible Serially Concatenated Convolutional Codes

Alexandre Graell i Amat, Member, IEEE, Guido Montorsi, Senior Member, IEEE, and Francesca Vatta, Member, IEEE

Abstract—In this paper, a novel class of serially concatenated convolutional codes (SCCCs) is addressed. In contrast to standard SCCC, where high rates are obtained by puncturing the outer code, the heavy puncturing is moved to the inner code, which can be punctured beyond the unitary rate. We derive analytical upper bounds on the error probability of this code structure by considering an equivalent code construction consisting of the parallel concatenation of two codes, and address suitable design guidelines for code optimization. It is shown that the optimal puncturing of the inner code depends on the outer code, i.e., it is interleaver dependent. This dependence cannot be tracked by the analysis for standard SCCC, which fails in predicting code performance.

Based on the considerations arising from the bounds analysis, we construct a family of rate-compatible SCCC with a high level of flexibility and a good performance over a wide range of code rates, using simple constituent codes. The error rate performance of the proposed codes is found to be better than that of standard SCCC, especially for high rates, and comparable to the performance of more complex turbo codes.

Index Terms—Serially concatenated convolutional codes, iterative decoding, rate-compatible codes, turbo codes, performance bounds.

I. INTRODUCTION

RATE-COMPATIBLE codes were introduced for the first time in [1], where the concept of punctured codes was extended to the generation of a family of rate-compatible punctured convolutional (RCPC) codes. Two codes of different rate, belonging to a rate-compatible code family, are said to be rate-compatible if the higher rate code is embedded into the lower rate code of the family. Namely, a higher rate code is obtained from a lower rate code of the family by adding some more puncturing (up to the desired (higher) rate), while keeping unaltered the bits which were already punctured in the lower rate code. In other words, the rate-compatibility restriction requires that the rates are organized in a hierarchy, where all code bits of a high rate punctured code are used by all the lower rate codes. This technique can be used in automatic repeat request / forward-error correction (ARQ/FEC) schemes (see, e.g., [2] and [3]), where, if the transmission is unsuccessful, information or parity bits are not simply repeated, but additional code bits of a lower rate code are transmitted, until the code is powerful enough to achieve error free decoding. Furthermore, rate-compatible codes can be used to provide unequal error protection (UEP) (see, e.g., [4] and [5]), since the rate-compatibility allows rate variation within a data frame to achieve unequal error protection. The concept of rate-compatible codes was later extended to parallel concatenated convolutional codes (PCCCs) [6, 7] and to serially concatenated convolutional codes (SCCCs) [8].

A new class of hybrid serially concatenated codes was proposed in [9] with bit error performance between that of PCCCs and SCCC. A similar concept was presented in [10, 11] to obtain well performing rate-compatible SCCC families. In contrast to standard SCCC, codes in [10, 11] were obtained puncturing both inner code parity bits and systematic bits, thereby obtaining rates beyond the outer code rate. Some design criteria to obtain good rate-compatible SCCC families were also discussed in [10, 11]. However, the considerations in [10, 11] were limited to heuristic design guidelines, with no theoretical analysis support. On the other hand, the analysis for standard SCCC [12] fails in predicting the performance of this code structure, hence it is not suited to derive proper design guidelines. Therefore, a deeper and more formal insight on the performance of this new class of SCCC is required, in order to provide suitable design guidelines aimed at code optimization. The main goal of this paper is to fill this gap to provide a performance analysis of this new class of concatenated codes.

The error probability performance of turbo-like codes is characterized by a steep transition region, the so-called waterfall region, followed by a more gently sloping plateau region, referred to as the error floor region. Code design criteria in the error floor region are usually based on union bounds to the error probability, since they allow to connect the performance of the code to some code parameters (essentially the weight spectrum) [13]. On the other hand, the design of turbo-like
codes in the waterfall region is mainly based on extrinsic information transfer (EXIT) charts or related techniques [14, 15], since it is hard to relate code performance in the low signal to noise ratio region with code parameters to derive code design criteria. Moreover, in most cases design criteria for the waterfall and error floor regions do not coincide. In fact, optimizing the performance in the waterfall region leads in general to a bad performance in the error floor region and vice versa.

In this paper, by properly redrawing the SCCC as the parallel concatenation of two codes, we derive the analytical upper bounds on the error probability, based on the union bound, using the concept of uniform interleaver. We then propose suitable design criteria for code optimization in the error floor. The analysis shows that the optimal puncturing of the inner code depends on the outer code and, therefore, it is interleaver dependent. This explains why the analysis for standard SCCCs fails in predicting the performance of this code structure. Based on the considerations arising from the union bounds analysis, we address design guidelines to obtain well-performing rate-compatible SCCC families. The proposed codes offer very good performance over a range of rates, including very high ones, using simple constituents. They perform better than standard SCCCs and compare favorably to more complex turbo codes.

It is worth mentioning that bounds improving the union bounds have been presented, e.g., in [16] and [17]. The bounds in [16] and [17] are tighter than the union bound at low SNR values while they coincide with the union bound for high SNR values. Unfortunately, from these bounds it seems very difficult to extract code design criteria in the waterfall region, which still remains an open problem. For these reasons we have elected to use the union bound argument, and we have focused on the puncturing design optimization in the error floor region.

The remainder of the paper is organized as follows. In Section II, we describe the new class of concatenated codes addressed in the paper. In Section III, upper bounds on the error probability are derived, and design criteria are addressed. Design guidelines to obtain well-performing SCCC families are discussed in Section IV. In Section V simulation results are given and some conclusions are drawn in Section VI.

II. A NEW CLASS OF SERIALLY CONCATENATED CONVOLUTIONAL CODES

The block diagram of an SCCC (C) is shown in Fig. 1. We consider the concatenation of two convolutional codes. The information sequence \( u \) of length \( N \) bits is first encoded by an encoder \( C_a \) of rate \( R_a \), punctured by a fixed puncturer \( P_a \) and interleaved before being forwarded to encoder \( C_b \) of rate \( R_b \). The interleaver length is denoted by \( N_x \). While no assumptions are made for \( C_a \), we require \( C_b \) to be systematic and recursive\(^1\). The output of \( C_b \) is punctured by puncturers \( P_{b}^{s} \) and \( P_{b}^{p} \), for systematic bits and parity bits, respectively. The resulting punctured systematic bits \( x_{b}^{s} \) and parity bits \( x_{b}^{p} \) are finally multiplexed to form the transmitted codeword \( x \)

of length \( N \) bits. Encoder \( C_a \) together with puncturer \( P_a \) are referred to as the outer code \( C_O \) and encoder \( C_b \) together with puncturers \( P_{b}^{s} \) and \( P_{b}^{p} \) are referred to as the inner code \( C_I \). Consequently, we denote by \( R_O \) and \( R_I \) the rates of \( C_O \) and \( C_I \), respectively. Also, we denote by \( R \) the rate of the SCCC.

In contrast to standard SCCCs, where high rates are obtained by concatenating a heavy punctured outer code with a rate \( R_I \leq 1 \) inner code (typically rate-1) such that the rate of the SCCC is less than or equal to the rate of the outer code \( R \leq R_O \), we allow the inner encoder in Fig. 1 to be punctured beyond the unitary rate, i.e., the overall code rate can be larger than the outer code rate.

A puncturer \( P \) is defined by a puncturing pattern \( p \) with a certain pattern length, also called period, \( N_p \). For each puncturer, we also define the permeability rate \( \rho \) \((0 \leq \rho \leq 1)\) as the fraction of bits remaining after puncturing. In this paper, we refer to puncturing patterns as vectors, rather than matrices. A rate-compatible code family can be then described by a single vector with entries listing the indices of the bits positions to be punctured in a specific order. For example, for the puncturing pattern \( p = [4, 2, 1, 3] \), with \( N_p = 4 \), bit in position 4 is punctured first, followed by bits in positions 2 and 1. Therefore, we obtain the series of binary puncturing patterns \([1, 1, 1, 1], [1, 1, 1, 0], [1, 0, 1, 0], [0, 0, 1, 0], [0, 0, 0, 0] \), which represent a rate compatible code family with permeability rates 1/3, 1/4, 1/2, and 1/4, respectively. For block lengths longer than \( N_p \), the puncturing patterns are just repeated with period \( N_p \). \( P_{b}^{s} \) and \( P_{b}^{p} \) have puncturing patterns \( p_{a} \), \( p_{b}^{s} \) and \( p_{b}^{p} \) of lengths \( N_a \), \( N_{b}^{s} \) and \( N_{b}^{p} \), and permeability rates \( \rho_{a} \), \( \rho_{b}^{s} \) and \( \rho_{b}^{p} \), respectively.

With these definitions the rate of the resulting SCCC is given by

\[
R = \frac{K}{N} = R_O R_I = \frac{R_a}{\rho_{a}} \frac{1}{\rho_{b}^{s} + \left(\frac{1-R_{b}}{R_{b}}\right) \rho_{b}^{p}} \quad (1)
\]

As it will be shown later, this code structure offers superior performance to that of standard SCCCs, especially for high rates. High-rate standard SCCCs suffer from a high error floor, due to the heavy puncturing of the outer code, which leads to a poor code in terms of free distance. On the other hand, the code structure discussed here allows for a constant block length of the outer code, maintaining its distance spectrum properties and thus keeping the interleaver gain for all code rates. The result is a much lower error floor. Moreover, this structure is well suited for the construction of rate-compatible code families.

III. ANALYTICAL UPPER BOUNDS ON THE ERROR PROBABILITY

The design of concatenated codes with interleavers involves the choice of the interleaver and the constituent encoders. Unfortunately, a joint optimization is prohibitively complex.

In [13] Benedetto and Montorsi proposed a method to evaluate the error probability of PCCCs independently from the interleaver used. The method consists in a decoupled design, in which one first designs the constituent encoders, and then tailors the interleaver on their characteristics. To achieve

\(^1\)According to [12] the inner encoder must be recursive to yield an interleaver gain.
this goal, the notion of uniform interleaver was introduced in [13]. The use of the uniform interleaver drastically simplifies the performance evaluation of turbo codes. Following this approach, design criteria for the construction of serially concatenated codes were derived in [12].

The performance of the code in Fig. 1 depends on $P_a$, $P_b^I$ and $P_b^O$, i.e., on puncturing patterns $p_a$, $p_b^I$ and $p_b^O$, and, subsequently, on the permeability rates $\rho_a$, $\rho_b^I$ and $\rho_b^O$, which should be properly optimized. In this section, we clarify some relevant aspects of this code structure providing the clues for code optimization and addressing design guidelines to properly select $\{C_a, P_a\}$, and $\{C_b, P_b^I, P_b^O\}$. To this purpose, we derive the analytical upper bounds on the bit and frame error probabilities, following the concept of uniform interleaver.

It is worth stressing that in the analysis of this section, we do not treat the code structure of Fig. 1 as a standard SCCC. Therefore, we do not apply the considerations in [12], but rather derive a new bound and new design criteria. The analysis in [12] would consider $C_0$ and the puncturers $P_b^I$ and $P_b^O$ as a single entity (the inner code $C_1$) therefore diluting the contribution of the inner code systematic bits and parity bits to the performance. Here, the key idea is to decouple the contribution of inner code systematic bits and inner code parity bits to the error probability bound to better identify how to choose $\{p_b^I, \rho_b^I\}$ and $\{p_b^O, \rho_b^O\}$. In fact, we shall show that to obtain good SCCCs in the form of Fig. 1, the selection of the inner code puncturing directly depends on the outer code (and therefore it is interleaver dependent), which has a crucial impact on performance. This dependence cannot be taken into account by the upper bounds derived in [12] for SCCCs.

Denote by $w_m$ the minimum weight of an input sequence generating an error event of code $C_a$, and by $h_m$ the minimum weight of the codewords of $C$. Also, let $A_{w,h}^{C_a}$ denote the Input-Output Weight Enumerating Function (IOWEF), the number of codewords in code $C$ with input weight $w$ and output weight $h$. Similarly, we define $A_{w,h}^{C_0}$ and $A_{w,h}^{C_1}$ for codes $C_0$ and $C_1$, respectively. The bit error probability of an SCCC over an additive white Gaussian noise channel can be upper bounded by [12]

$$P_b(e) \leq \frac{1}{2} \sum_{h=h_m}^{N_0/2} \sum_{w=w_m}^{K} \frac{w}{K} A_{w,h}^{C,w,h} \text{erfc} \left( \sqrt{\frac{hRE_b}{N_0}} \right)$$

(2)

where $N_0/2$ is the two-sided noise power spectral density, and $E_b$ is the energy per information bit.

Likewise, the frame error probability is upper bounded by

$$P_w(e) \leq \frac{1}{2} \sum_{h=h_m}^{N} \frac{A_{w}^{C} \text{erfc} \left( \sqrt{\frac{nRE_b}{N_0}} \right)}{K}$$

(3)

where $A_{w}^{C} = \sum_{w=w_m}^{K} A_{w,h}^{C}$.

$A_{w,h}^{C}$ can be calculated by replacing the actual interleaver with the uniform interleaver [13] and exploiting its properties. The uniform interleaver transforms a codeword of weight $l$ at the output of the outer encoder into all its distinct $\binom{N_0}{l}$ permutations. As a consequence, each codeword of the outer code of weight $l$, through the action of the uniform interleaver, enters the inner encoder generating $\binom{N}{l}$ codewords of the inner code. The IOWEF of the overall SCCC can then be evaluated from the knowledge of the IOWEFs of $C_0$ and $C_1$ [12]:

$$A_{w,h}^{C} = \sum_{l=0}^{N_0/2} A_{w,l}^{C_0} A_{l,h}^{C_1} \binom{N}{l}$$

(4)

To evaluate the performance of the code structure of Fig. 1, instead of proceeding as in [12] using (4), it is more suitable to refer to Fig. 2, which properly redraws the encoder of Fig. 1 for the derivation of the upper bound. In Fig. 2, $C_1 = C_0$, $P_0 = P_a$, and $C_2$ is now the rate $\frac{R_U}{1-R_U}$ convolutional code corresponding to the parity part of $C_0$. Further, $P_1 = n^{-1}(P_b^O)$, while $P_2$ is identical to $P_b^O$. Therefore, $x_U$ in Fig. 2 is a scrambled version of $x_I$ in Fig. 1 (i.e., it gives the same contribution to the output weight as $x_I$), while $x_L = x_b^O$. The lengths of $x_U$ and $x_L$ are denoted by $N_U$ and $N_L$, respectively. Note also that the vectors $v$ and $z$ in Fig. 1 and Fig. 2 contain the same bits. For later reference, the three dashed boxes in Fig. 2 define three corresponding codes referred to as the outer code $C_U$ (equal to $C_0$ in Fig. 1), the upper code $C_L$ and the lower code $C_L$, respectively. We call $C_U$ the code $C_1$ punctured by the concatenation of $P_1$ and $P_1$, and $C_L$ the code $C_2$ punctured by $P_2$. $P_1$ and $P_2$ have puncturing patterns $p_1$ and $p_2$ of lengths $N_1$ and $N_2$, and permeability rates $\rho_1$ and $\rho_2$, respectively.

Fig. 2 allows us to decouple the contribution of inner code systematic bits and parity bits to the error probability bound. Now, the serially concatenated code structure under consideration can be interpreted as the parallel concatenation of $C_U$ and $C_L$. Therefore, the weight $h$ of code $C$ codewords can be split into two contributions $j$ and $m$, corresponding to the weights of the codewords of $C_U$ and $C_L$, respectively, such that $h = j + m$. For later use denote by $d_{l}^{C_U}$ and $d_{l}^{C_L}$ the free distance of code $C_U$ and code $C_L$, respectively, and by $R_U$ the rate of $C_U$. Also, let $n^{U}$ be the number of concatenated error events in a codeword of $C_U$ of weight $j$ associated to a codeword of $C_U$ of weight $l$ generated by a weight $w$ information sequence, and define $n_{m}^{U}$ as the maximum value of $n^{U}$. In a similar way we define $n^{L}$ as the number of
concatenated error events in a codeword of $C_L$ of weight $m$ and input weight $l$, and $n_{\pi}$ as its maximum value. With reference to Fig. 2, equation (4) can then be rewritten as

$$A_{w,h}^c = A_{w,j+m} = \sum_{l=d^{UL}_i}^{N_\pi} \sum_{j=d^{UL}_i}^{N_\pi} A_{w,l,j}^{C_L} \left( \begin{array}{c} N_\pi \\ l \end{array} \right) \left( \begin{array}{c} N_\pi \\ l \end{array} \right)_{j+m=h}$$

where $A_{w,l,j}^{C_U}$ denotes the number of codewords of $C_U$ of weight $j$ associated with a codeword of $C_O$ of weight $l$ generated by an information word of weight $w$, and $A_{l,m}^{C_L}$ indicates the number of codewords of $C_L$ of weight $m$ generated by an input weight $l$.

Assume $R = k/n$ with $R_O = k/p$ and $R_I = p/n$. $A_{w,l,j}^{C_U}$ and $A_{l,m}^{C_L}$ can be expressed as [12]

$$A_{w,l,j}^{C_U} \leq \sum_{n_U=1}^{n_{\pi}} \left( \begin{array}{c} N_\pi/p \\ n_U \end{array} \right) A_{w,l,j,n_U}^{C_U}$$

$$A_{l,m}^{C_L} \leq \sum_{n_L=1}^{n_{\pi}} \left( \begin{array}{c} N_\pi/p \\ n_L \end{array} \right) A_{l,m,n_L}^{C_L}$$

where $A_{w,l,j,n_U}^{C_U}$ represents the number of code $C_U$ sequences of weight $j$ associated with a codeword of $C_O$ of weight $l$ generated from an information word of weight $w$, and number of concatenated error events $n_U$. $A_{l,m,n_L}^{C_L}$ represents the number of code $C_L$ sequences of weight $m$, input weight $l$, and number of concatenated error events $n_L$.

Substituting (6) in (5) we get [12]:

$$A_{w,j+m}^{C_L} \leq \sum_{l=d^{UL}_i}^{N_\pi} \sum_{j=d^{UL}_i}^{N_\pi} \sum_{n_U=1}^{n_{\pi}} \sum_{n_L=1}^{n_{\pi}} \left( \begin{array}{c} N_\pi/p \\ n_U \end{array} \right) \left( \begin{array}{c} N_\pi/p \\ n_L \end{array} \right) \left( \begin{array}{c} N_\pi \\ l \end{array} \right) \left( \begin{array}{c} N_\pi \\ l \end{array} \right)_{j+m=h}$$

$$A_{w,l,j,n_U}^{C_U} \leq \sum_{n_U=1}^{n_{\pi}} \sum_{n_L=1}^{n_{\pi}} \left( \begin{array}{c} N_\pi \\ n_U \end{array} \right) \left( \begin{array}{c} N_\pi \\ n_L \end{array} \right) \left( \begin{array}{c} N_\pi \\ l \end{array} \right) \left( \begin{array}{c} N_\pi \\ l \end{array} \right)_{j+m=h}$$

Finally, substituting (7) into (2), we obtain the upper bound on the bit error probability,

$$P_b(e) \leq \frac{1}{2} \sum_{j+m=h_{m}}^{N} \text{erfc} \left( \sqrt{\frac{(j+m)R_Eb}{N_0}} \right)$$

$$\cdot \sum_{w=w_{m}}^{K} \sum_{l=d^{UL}_i}^{N_\pi} \sum_{j=d^{UL}_i}^{N_\pi} \sum_{n_U=1}^{n_{\pi}} \sum_{n_L=1}^{n_{\pi}} \left( \begin{array}{c} N_\pi \\ n_U \end{array} \right) \left( \begin{array}{c} N_\pi \\ n_L \end{array} \right)$$

$$\cdot \frac{1!}{p^{n_U+n_L}L!} \frac{1!}{L!} \frac{1!}{n_U!n_L!}$$

Equivalently, substituting (7) into (3) the upper bound on the frame error probability is given by

$$P_w(e) \leq \frac{1}{2} \sum_{j+m=h_{m}}^{N} \text{erfc} \left( \sqrt{\frac{(j+m)R_Eb}{N_0}} \right)$$

$$\cdot \sum_{w=w_{m}}^{K} \sum_{l=d^{UL}_i}^{N_\pi} \sum_{j=d^{UL}_i}^{N_\pi} \sum_{n_U=1}^{n_{\pi}} \sum_{n_L=1}^{n_{\pi}} \left( \begin{array}{c} N_\pi \\ n_U \end{array} \right) \left( \begin{array}{c} N_\pi \\ n_L \end{array} \right)$$

$$\cdot \frac{1!}{p^{n_U+n_L}L!} \frac{1!}{L!} \frac{1!}{n_U!n_L!}$$

For large $N_\pi$ and for a given $h = j + m$, the dominant coefficient of the exponentials in (8) and (9) is the one for which the exponent of $N_\pi$ is maximum. This maximum exponent is defined as [12]

$$\alpha(h = j + m) \triangleq \max_{w,l} \left\{ n_U + n_L - l - 1 \right\}$$

For large $E_b/N_0$, the dominating term is $\alpha(h_m)$, corresponding to the minimum value $h = h_m$ [12],

$$\alpha(h_m) \leq 1 - d_{i}^{CO}$$

Substituting (11) in (8) and truncating to the first term of the summation with respect to $h$ we obtain:

$$P_b(e) \lesssim B N_\pi^{-d_i^{CO}} \text{erfc} \left( \sqrt{\frac{h_{m} R_Eb}{N_0}} \right)$$

asymptotic with respect to $E_b/N_0$, where $B$ is a constant that depends on the weight properties of the encoders.

On the other hand, the dominant contribution to the bit and frame error probability for $N_\pi \rightarrow \infty$ is the largest exponent of $N_\pi$, defined as

$$\alpha_M \triangleq \max_{h} \alpha(h = j + m) = \max_{w,l,h} \left\{ n_U + n_L - l - 1 \right\}$$

We consider only the case of recursive convolutional inner encoders. In this case, $\alpha_M$ is given by [12]

$$\alpha_M = - \left\lfloor \frac{d_i^{CO} + 1}{2} \right\rfloor$$

Then, substituting (14) in (8) and truncating to the term of the summation corresponding to $h(\alpha_M)$ we obtain the following result, asymptotic with $N_\pi$ and $E_b/N_0$:

$$P_b(e) \lesssim C N_\pi^{\alpha_M} \text{erfc} \left( \sqrt{\frac{h(\alpha_M) R_Eb}{N_0}} \right)$$

where $C$ is a constant that depends on the weight properties of the encoders and $h(\alpha_M)$ is the weight associated to the highest exponent of $N_\pi$.

Denote by $d_i^{CL}$ the minimum weight of code $C_L$ sequences generated by input sequences of weight $i$. For instance, $d_i^{CL}$ and $d_i^{CL}$ are the minimum weight of code $C_L$ sequences generated by input sequences of weight 2 and 3, respectively. Following similar derivations as in [12] for standard SCCCs,
we obtain the following results for the weight \( h(\alpha M) \) associated to the highest exponent of \( N_p \):

\[
h(\alpha M) = \frac{a_{\text{even}}^C}{2} + d_{\min}^C(f_{\text{even}}^C) \quad \text{if } f_{\text{even}}^C
\]

\[
h(\alpha M) = \frac{(a_{\text{even}}^C - 3)d_{\text{even}}^C}{2} + d_{\min}^C(f_{\text{even}}^C) \quad \text{if } f_{\text{even}}^C
\]

where \( d_{\min}^C(f_{\text{even}}^C) \) is the minimum weight of \( C \) code sequences corresponding to a \( C \) code sequence of weight \( a_{\text{even}}^C \).

Finally from (14)-(16) we obtain:

\[
P_b(\epsilon) \leq C_{\text{even}}N_p^{-a_{\text{even}}^C/2} \cdot \text{erfc} \left( \frac{\left( \frac{a_{\text{even}}^C d_{\text{even}}^C}{2} + d_{\min}^C(f_{\text{even}}^C) \right) \cdot \text{RE}_b}{N_0} \right)
\]

if \( f_{\text{even}}^C \) is even, and

\[
P_b(\epsilon) \leq C_{\text{odd}}N_p^{-a_{\text{odd}}^C/2} \cdot \text{erfc} \left( \frac{\left( (a_{\text{odd}}^C - 3)d_{\text{odd}}^C + d_{\min}^C(f_{\text{odd}}^C) \right) \cdot \text{RE}_b}{N_0} \right)
\]

if \( f_{\text{odd}}^C \) is odd. Constants \( C_{\text{even}} \) and \( C_{\text{odd}} \) can be derived as in [12] for standard SCCC. Similar expressions can be obtained for the frame error probability.

We can draw from (17) and (18) some important design considerations for the code structure of Fig. 1:

- Similar to standard SCCC, \( \{C_1, P_0\} \) should be chosen so that the output weight enumerating function (OIEWF) of \( C \) is optimized, i.e., the terms \( d_{\text{even}}^C, d_{\text{odd}}^C + 1, \ldots \), must be orderly maximized and their multiplicities minimized.
- The coefficient \( h(\alpha M) \) that multiplies the signal to noise ratio \( E_b/N_0 \) increases with \( d_{\text{even}}^C, d_{\text{odd}}^C, d_{\min}^C(f_{\text{even}}^C) \), and \( d_{\min}^C(f_{\text{odd}}^C) \). \( P_1 \) and \( \{C_2, P_2\} \) should be chosen so that \( h(\alpha M) \) is maximized. For a fixed pair of permeability rates \( \rho_{p_1} \) and \( \rho_{p_2} \), this turns into two design rules: (1) \( \{C_2, P_2\} \) must be chosen so that the IOWEF of \( C_L \) is optimized, i.e., the coefficients \( d_{\text{even}}^L, d_{\text{odd}}^L, d_{\min}^L(f_{\text{even}}^L) \), and \( d_{\min}^L(f_{\text{odd}}^L) \), must be orderly maximized and their multiplicities \( D_{\text{even}}^L, D_{\text{odd}}^L, \ldots \), minimized. (2) \( P_1 \) must be selected to maximize \( d_{\min}^L(f_{\text{even}}^C) \) and successive terms and to minimize their multiplicities.

Finally, to construct the SCCC of Fig. 1, set \( \{C_a, P_a\} = \{C_1, P_0\} \), \( \{C_b^+, P_b^+\} = \{C_2, P_2\} \), \( P_a = \pi(P_1) \), and \( C_b = \{1, C_b^0\} \), where \( 1 \) and \( C_b^0 \) stand for the systematic part and the parity part of \( C_b \) in Fig. 1, respectively. Note that the equality \( P_b^+ = \pi(P_1) \) introduces the dependence of the optimization of the inner code \( C_1 \) on the outer code \( C_0 \) through the interleaver.

A complementary analysis tool for the design of concatenated schemes is to consider the EXIT charts or equivalent plots [14, 15]. EXIT charts predict very well the behavior of iterative decoding schemes in the low SNR region (convergence region) and often lead to design rules that are in contrast with those arising from union bounds, which are more suited for the analysis in the error floor region. Unfortunately, EXIT chart analysis is mainly based on Monte Carlo simulations and does not allow to extract useful code design parameters. For this reason we have not included this technique in the paper. The reader however should be aware that for the careful design of concatenated schemes both aspects must be considered and this implies that comparison of the designed schemes through simulation cannot be avoided. This fact also allows to justify some differences in the simulation results which are not evident from the uniform interleaver analysis. A convergence analysis of this class of SCCC using EXIT charts is discussed in [18].

IV. RATE-COMPATIBLE SERIALLY CONCATENATED CONVOLUTIONAL CODES

The code structure of Fig. 1 is ideal for designing rate-compatible code families, since puncturing is moved to the inner code. To satisfy the rate-compatibility constraint the outer code should be kept fixed for all rates. This implies the use of a fixed puncturing pattern \( p_a \). Then, the design of well-performing rate-compatible code families reduces to choose the puncturing patterns \( p_b^+ \) and \( p_b^- \) and the permeability rates \( \rho_{p_1} \) and \( \rho_{p_2} \) to yield the best performance in both the error floor and the waterfall region. Unfortunately, a joint optimization is not feasible. Here, we consider a different approach, yet practical. The key idea is to optimize the puncturing patterns \( p_b^+ \) and \( p_b^- \) based on the design criteria derived in the previous section, thus optimizing the puncturing order for the error floor. Then \( \rho_{p_1} \) and \( \rho_{p_2} \) can be tuned to yield good performance in the waterfall region. The optimization of puncturings \( p_b^+ \) and \( p_b^- \) can be formulated in terms of the optimization of \( p_1 \) and \( p_2 \) in Fig. 2 as follows:

- **Optimization of \( p_2 \):** We recall that \( p_2 \) gives the order in which the bits must be punctured in puncturer \( P_2 \), so that the resulting punctured code is rate-compatible. The optimization of \( p_2 \) is then performed through a search algorithm that works incrementally, finding one punctured position at a time. The algorithm starts with a binary puncturing vector \( v_2 \) of length \( N_2 \) with all entries set to 1. Then, it finds the first position to be punctured by comparing the \( N_2 \) IOWEFs of all encoders \( C_L \) resulting from the puncturing of \( C_2 \) through all possible \( N_2 \) binary puncturing patterns (all binary vectors of length \( N_2 \) with a single zero) and keeping the puncturing position that gives the best IOWEF (see Section III). This gives the first entry of vector \( p_2 \). To ensure rate-compatibility, this position must be kept while searching the following positions to be punctured. To find the second position to be punctured the IOWEFs of all \( N_2 - 1 \) possible encoders \( C_L \) resulting from the puncturing of \( C_2 \) through all \( N_2 - 1 \) puncturing patterns with two zero entries (one corresponding to the index given by the first entry of vector \( p_2 \)) are compared. The position that gives the best IOWEF is kept and determines the second entry of vector \( p_2 \). This method can be continued until the order of all \( N_2 \) possible bits to be punctured is found. The result is a vector \( p_2 \) of \( N_2 \) indices stating the order in which the parity bits of the lower code \( C_2 \) shall be punctured.

The proposed search algorithm is summarized in the flow chart of Fig. 3, where IOWEFs of \( p_2 \) for the best IOWEF, \( j \) is the \( j \)-th entry of vector \( p_2 \) and \( i \) is the
index for the loop over the \( N_2 - (j - 1) \) IOWEFs to be compared to determine entry \( j \) of \( p_2 \).

- Optimization of \( p_1 \): The best puncturing pattern \( p_1 \) for \( C_U \) is found in a similar way by maximizing \( d_{\min}^{C_U}(d_f^{C_O}) \) and successive terms and minimizing the corresponding multiplicities. The algorithm starts with a binary puncturing vector of length \( N_1 \) with all entries set to 1. Then, it finds the first position to be punctured by comparing the terms \( d_{\min}^{C_U}(d_f^{C_O}) \), \( d_{\min}^{C_U}(d_f^{C_O}) + 1 \), ... , and the corresponding multiplicities of all codes \( C_U \) obtained by puncturing \( C_O \) through all possible \( N_1 \) puncturing patterns (all binary vectors of length \( N_1 \) with a single zero) and keeping the one giving the maximum \( d_{\min}^{C_U}(d_f^{C_O}) \) and the lowest multiplicity. If two encoders have the same \( d_{\min}^{C_U}(d_f^{C_O}) \) and corresponding multiplicity, we consider the second term \( d_{\min}^{C_U}(d_f^{C_O}) + 1 \), and so on. This gives the first entry of vector \( p_1 \). Again, to ensure rate-compatibility this position must be kept while searching the following positions to be punctured. To find the second position to be punctured the terms \( d_{\min}^{C_U}(d_f^{C_O}) \), \( d_{\min}^{C_U}(d_f^{C_O}) + 1 \), ... , of all \( N_1 - 1 \) possible encoders \( C_U \) obtained by puncturing \( C_O \) through all possible \( N_1 - 1 \) puncturing patterns with two zero entries (one corresponding to the index given by the first entry of vector \( p_1 \)) are compared. The position that gives the best sequence of \( d_{\min}^{C_U}(d_f^{C_O}) \), \( d_{\min}^{C_U}(d_f^{C_O}) + 1 \), ... , is kept and determines the second entry of vector \( p_1 \). This method is continued until the order of all \( N_1 \) possible bits to be punctured is found. Note that since \( p_1 \) may also puncture outer code information bits, the resulting code might be non-invertible and/or catastrophic. Therefore, at each step, the invertibility and non-catastrophicity of the code must be guaranteed.

- Finally, set \( p_b^o = p_2 \) and \( p_b^o = \pi(p_1) \).

V. Numerical Results

For the examples here, we have chosen \( C_s \) and \( C_o \) to be identical rate-1/2 \( (R_a = R_b = 1/2) \) 4-state systematic recursive convolutional codes, with generator polynomials \((1, 5/7)\) in octal form. \( C_s \) is punctured to rate 2/3 by applying the fixed binary puncturing pattern \([1, 1, 1, 0]\) (i.e., \( p_a = [4, 2, 3, 1] \) and \( p_o = 3/4 \)). With this puncturing pattern the free distance of code \( C_O \) is \( d_f^{C_O} = 3 \). The overall code rate is \( R = 1/3 \). Higher rates are then obtained by puncturing the inner encoder by puncturers \( P_b^o \) and \( P_b^o \) for systematic and parity bits, respectively, as previously discussed.

We have chosen \( N_b^o = N_b^p = 300 \), providing a high resolution for choosing the code rate \( R \). Table I reports the puncturing pattern \( p_1 \) for puncturer \( P_1 \) (the de-interleaved version of \( P_b^o \)), optimized assuming \( p_a \) above as the puncturing pattern of \( P_a \). As stated in Section III the optimization of \( P_1 \) depends on the outer code, thus for each \( p_a \) a new \( p_1 \) must be found. \( p_1 \) was optimized according to the design rules in the previous section. The puncturing pattern \( p_b^o \) is then obtained by interleaving \( p_1 \), \( p_b^o = \pi(p_1) \). Table I lists the indices of the successive bits positions to be punctured (see Section II).

For instance, assuming \( N_x = N_b^o = N_b^p \), if three bits are punctured in \( P_b^o \) (i.e., \( p_b^o = 297/300 \)) the bits in positions \( \pi(101) \), \( \pi(1) \), and \( \pi(193) \) will be punctured. Note that pattern \( p_1 \) in Table I has only 100 entries, while \( N_b^o = 300 \). The reason is that we restrict the rate of the equivalent code \( C_U \) to be \( R_U \leq 1 \), since it must be invertible. Since \( R_U = \frac{R_o}{R_b} \) and being \( R_o = 2/3 \) the maximum number of bits that can be punctured in \( P_b^o \) is \( N_b^o \cdot (1 - R_o/R_U^{\text{max}}) = 300 \cdot (1/3) = 100 \).

The puncturing pattern \( p_2 \) for puncturer \( P_2 \) is shown in Table II. As explained in Section III the puncturing pattern must be optimized so that \( d_f^{C_L} \), \( d_f^{C_3} \), ... , are maximized and their multiplicities \( \tilde{D}_L^f, \tilde{D}_S^f, \ldots \), are minimized for each puncturing position. The evolution of the values \( d_f^{C_L}, D_f^{C_L} \), for \( i = 2 \) and \( i = 3 \), as a function of the number of bits punctured in \( \tilde{P}_2 \) (\( 0 \leq \tilde{N}_2 \cdot (1 - \rho_2) \leq 300 \)) is reported in Fig. 4. For all \( i \) there are some \( d_f^{C_L} = 0 \) with a corresponding \( D_f^{C_L} \neq 0 \), which means that the corresponding code \( C_L \) is not invertible. Also, given a value of \( d_f^{C_L}, \tilde{D}_L^f \) is an increasing function of the number of punctured bits by construction.

\footnote{A code is said to be invertible if, knowing only the parity-check symbols of a code vector, the corresponding information symbols can be uniquely determined [19].}

\footnote{According to [12] it can be convenient to choose as outer code a nonrecursive encoder. Here, for simplicity, we considered identical encoders for \( C_s \) and \( C_o \). Moreover, it is worth mentioning that the difference in performance is marginal.
TABLE I

PUNCTURING PATTERN $p_1$: IT LISTS THE INDICES OF THE SUCCESSIVE BITS POSITIONS TO BE PUNCTURED IN $x_i^n$ ($p_{b_0}^i = \pi(p_1)$)

<table>
<thead>
<tr>
<th>Number of punctured bits</th>
<th>Bit index</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 10</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>11 - 20</td>
<td>6 7 8 9 10</td>
</tr>
<tr>
<td>21 - 30</td>
<td>11 12 13 14 15</td>
</tr>
<tr>
<td>31 - 40</td>
<td>16 17 18 19 20</td>
</tr>
<tr>
<td>41 - 50</td>
<td>21 22 23 24 25</td>
</tr>
<tr>
<td>51 - 60</td>
<td>26 27 28 29 30</td>
</tr>
<tr>
<td>61 - 70</td>
<td>31 32 33 34 35</td>
</tr>
<tr>
<td>71 - 80</td>
<td>36 37 38 39 40</td>
</tr>
<tr>
<td>81 - 90</td>
<td>41 42 43 44 45</td>
</tr>
<tr>
<td>91 - 100</td>
<td>46 47 48 49 50</td>
</tr>
</tbody>
</table>

TABLE II

PUNCTURING PATTERN $p_2$: IT LISTS THE INDICES OF THE SUCCESSIVE BITS POSITIONS TO BE PUNCTURED IN $x_i^n$ ($p_{b_0}^i = p_2$)

<table>
<thead>
<tr>
<th>Number of punctured bits</th>
<th>Bit index</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 10</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>11 - 20</td>
<td>6 7 8 9 10</td>
</tr>
<tr>
<td>21 - 30</td>
<td>11 12 13 14 15</td>
</tr>
<tr>
<td>31 - 40</td>
<td>16 17 18 19 20</td>
</tr>
<tr>
<td>41 - 50</td>
<td>21 22 23 24 25</td>
</tr>
<tr>
<td>51 - 60</td>
<td>26 27 28 29 30</td>
</tr>
<tr>
<td>61 - 70</td>
<td>31 32 33 34 35</td>
</tr>
<tr>
<td>71 - 80</td>
<td>36 37 38 39 40</td>
</tr>
<tr>
<td>81 - 90</td>
<td>41 42 43 44 45</td>
</tr>
<tr>
<td>91 - 100</td>
<td>46 47 48 49 50</td>
</tr>
</tbody>
</table>

TABLE III

PARAMETERS OF SEVERAL $R = 2/3$ SCCCs WITH $N_b = 300$ BITS AND PUNCTURING PATTERNS $p_1$ AND $p_2$ OF TABLES I AND II, RESPECTIVELY

<table>
<thead>
<tr>
<th>$p_{b_0}^i$</th>
<th>$d_3^L$</th>
<th>$d_0^{C_{max}}$</th>
<th>$d_0^{C}$</th>
<th>$h(\alpha_M)$</th>
<th>$h(m)$</th>
<th>$H_{b_0}$</th>
<th>Markers</th>
</tr>
</thead>
<tbody>
<tr>
<td>20/300</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>5.60E-01</td>
<td>□</td>
</tr>
<tr>
<td>40/300</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4.81E-03</td>
<td>□</td>
</tr>
<tr>
<td>60/300</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>7.12E-03</td>
<td>□</td>
</tr>
<tr>
<td>80/300</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>5.28E-03</td>
<td>□</td>
</tr>
<tr>
<td>100/300</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1.40E-04</td>
<td>□</td>
</tr>
</tbody>
</table>

Listed in Table III are the parameters $d_3^L$, $d_0^{C_{max}}$ ($d_0^{C}$), $h(\alpha_M)$, $h(m)$ and its multiplicity $H_{b_0}$, for several $R = 2/3$ SCCCs of permeability rate $p_{b_0}^i$, can be obtained from (1) fixing $R = 2/3$ and $R_a = R_b = 1/2$. The block length is $K = 200$ bits ($N_b = 300$). The different values of $p_{b_0}^i$ are listed as rational numbers with denominator $N_b$.

Nevertheless, it should be stressed that a sufficient number of systematic bits should be preserved in order to ensure a good behavior for high $E_b/N_0$ values. This can be observed for the curve $p_{b_0}^i = 100/300$, which shows a worse slope. Indeed, for asymptotic values of $E_b/N_0$, the performance is dominated by $h(m)$, the minimum weight of code sequences. Therefore, the best performance for very high signal-to-noise ratios $E_b/N_0$ is obtained for $p_{b_0}^i = 20/300$ (curve with squares in Fig. 5), since the corresponding code has $h_{b_0} = 3$, whereas the worst performance is obtained for $p_{b_0}^i = 100/300$ (curve with circles in Fig. 5), since the corresponding code has $h_{b_0} = 1$.

In the same figure we report the union bounds for codes...
with $\rho^p_b = 20/300$ and $\rho^p_b = 100/300$ computed following the approach for standard SCCCs [12] (curve with solid squares for $\rho^p_b = 20/300$ and with solid circles for $\rho^p_b = 100/300$). It is interesting to observe that the union bound analysis in [12] fails in predicting the performance of the SCCC of Fig. 1.

In Fig. 6 we give FER curves (solid curves with empty markers) together with the union bounds on the error probability (dashed curves with empty markers) for the rate-2/3 SCCCs of Table III. Here, the block length is $K = 2000$ bits. The curves were obtained with a log-map SISO algorithm and 10 decoding iterations and assumed random interleavers. The puncturing patterns of Tables I and II were applied periodically. The simulation results show a very good agreement with the analytical bounds. We observe that lower error floors can be obtained by increasing $\rho^p_c$. For example, the code $\rho^p_b = 8/30$ shows a gain of 1.4 dB at $\text{FER} = 10^{-5}$ with respect to the code $\rho^p_b = 2/30$. However this gain tends to vanish for very high $E_b/N_0$, where the term $h_m$ is dominant (note the crossing of the two curves at $E_b/N_0 = 8$ dB). In the same figure we plot the union bound of the SCCC with $\rho^p_b = 20/300$ computed using the approach in [12] (the dashed curve with solid squares). The bound does not match the simulated curve.

On the other hand, the performance in the waterfall region can be explained in part by looking at the cumulative function $\sum_{i=1}^{d} A_i^{C}$ of the output distance spectrum of the serially concatenated code $C$. The codes for which the cumulative function of the average distance spectrum is minimum perform better at low SNRs, since, in this region, the higher distance error events have a nontrivial contribution to error performance. The cumulative functions of the codes listed in Table III are traced in Fig. 7. The worst performance for low signal-to-noise ratios $E_b/N_0$ is obtained for $\rho^p_b = 20/300$ (curve with squares in Fig. 5), since the corresponding code has the maximum cumulative function of the average distance spectrum, whereas the best performance is obtained for $\rho^p_b = 100/300$ (curve with circles in Fig. 5), since the corresponding code has the minimum cumulative function of the average distance spectrum. This is in agreement with the simulation results of Fig. 6.

For comparison purposes, we also report in Fig. 6 the simulated performance of the rate-2/3 PCCC proposed in [20], of the rate-2/3 SCCC proposed in [8] and the simulated performance of a standard SCCC consisting of an outer 8-state code punctured to rate-2/3 and inner accumulator. The PCCC code in [20] was obtained by optimally puncturing the mother code specified in the wideband code-division multiple-access (WCDMA) and CDMA2000 standards, consisting of the parallel concatenation of two rate-1/2, 8-state, convolutional codes. The SCCC in [8] is the same as our baseline code (two rate-1/2, 4-state, systematic recursive encoders), but puncturing is limited to inner code parity bits. As it can be observed in Fig. 6, the proposed SCCC shows a significant gain in the error floor region with respect to the more complex code in [20]. On the other hand, the code in [8] performs much worse than our code since puncturing is limited to parity bits, which penalizes code performance in the error floor as discussed above. The proposed code with $\rho^p_b = 80/300$ performs also slightly better than the standard SCCC of similar complexity. Note that the standard SCCC construction is not adapted to rate-compatibility.

Fig. 8 depicts the FER of rate-9/10 SCCCs as a function of the permeability rate $\rho^p_b$ for several values of $E_b/N_0$. The minimum of each curve indicates the permeability rate that minimizes the FER for that particular $E_b/N_0$. The figure shows that the optimal permeability rates depend on the region of interest. For high $E_b/N_0$ (i.e., error floor region), to achieve low error rates more parity bits should be kept, i.e., $\rho^p_b$ must be increased. On the other hand, for low $E_b/N_0$ (i.e., convergence region) $\rho^p_b$ and $\rho^p_c$ should be more evenly balanced.

Finally, in Fig. 9 we compare the simulated performance of several SCCCs of rate $R = 9/10$ with the analytical upper bounds. The block length is $K = 2000$ bits. Similar to the case $R = 2/3$ the curves show that lower error floors are obtained by keeping more parity bits in $C_1$, i.e., for higher $\rho^p_b$. Nevertheless, it should be stressed that some inner code systematic bits must be kept in order to allow convergence of the decoding process. For comparison purposes, we also report
in the same figure the performance of the rate-9/10 PCCC proposed in [20] and that of a standard SCCC consisting of an outer 8-state code punctured to rate 9/10 and inner accumulator. A gain of 2 dB at FER $10^{-5}$ is obtained for the code with $\rho_p = 16/300$ with respect to the code in [20]. Further, the proposed codes perform much better than the standard SCCC, which shows a high error floor due to the heavy puncturing of the outer code. In the same figure we report the union bound for SCCCs in [12] for the code with $\rho_p = 16/300$ (dashed curve with solid circles). The predicted floor is much higher than the actual floor. On the other hand, the analysis of Section III correctly predicts code performance. Indeed, the bound in [12] and the bound of Section III coincide for $R = 1/3$, i.e., when no puncturing is applied to the inner code. However, for higher rates, the bound in [12] is not able to capture the performance of the code structure of Fig. 1. The disagreement with respect to the simulation results is more significant for higher rates, as it can be observed in Figs. 6 and 9.

VI. CONCLUSIONS

In this paper, we provided a performance analysis of a novel class of serially concatenated convolutional codes, where the inner code can be punctured beyond the unitary rate. We derived the union bounds on the error probability of this code structure by considering an equivalent code construction consisting of the parallel concatenation of two codes. The analysis showed that the optimization of the inner code puncturing depends on the outer code. This dependence cannot be tracked by the analysis for standard SCCCs, which explains that they fail in predicting the performance of this code structure.

Based on the union bounds analysis, we derived design criteria for code optimization and design guidelines for the construction of well-performing rate-compatible code families. The proposed codes show superior performance than standard SCCCs, especially for high rates, and perform better than more complex PCCCs.

The code analyzed in this paper, due to its simplicity and versatility, was chosen for the implementation of a very high speed (1Gbps) Adaptive Coded Modulation modem for satel-
lite application. The interested reader can find implementation details in [21].

REFERENCES