Design and Performance of Turbo Gallager Codes

Giulio Colavolpe

***Abstract*—The most powerful channel-coding schemes, namely, those based on turbo codes and low-density parity-check (LDPC) Gallager codes, have in common the principle of iterative decoding. However, the relative coding structures and decoding algorithms are substantially different. This paper shows that recently proposed novel coding structures bridge the gap between these two schemes. In fact, with properly chosen component convolutional codes, a turbo code can be successfully decoded by means of the decoding algorithm used for LDPC codes, i.e., the belief-propagation algorithm working on the code Tanner graph. These new turbo codes are here nicknamed "turbo Gallager codes." Besides being interesting from a conceptual viewpoint, these schemes are important on the practical side because they can be decoded in a fully parallel manner. In addition to the encoding complexity advantage of turbo codes, the low decoding complexity allows the design of very efficient channel-coding schemes.**

***Index Terms*—Belief propagation (BP), iterative decoding, low-density parity-check (LDPC) codes, turbo codes.**

I. Introduction

IN RECENT years, great attention has been devoted to powerful coding schemes which achieve near-Shannon limit performance with affordable decoding complexity. In 1993, turbo codes were first introduced [1], [2]. A turbo encoder is the parallel concatenation of two simple constituent encoders interconnected through an interleaver. The corresponding decoding process is based on an *iterative* algorithm in which each component decoder takes advantage of the *extrinsic* information produced by the other decoder at the previous step. This iterative decoding process usually employs *soft-output* component decoders based on the algorithm by Bahl *et al.* (BCJR) [3] or its simplified logarithmic versions [4]. After the invention of turbo codes, this decoding technique was extended to serially concatenated convolutional codes (SCCCs). SCCCs are based on a serial concatenation, through an interleaver, of an outer code and an inner recursive code [5].

The extraordinary success of turbo codes has stimulated the rediscovery of another class of codes exhibiting similar performance and characteristics [6]. These codes, called low-density parity-check (LDPC) codes, were first introduced by Gallager [7] in their original *regular* version. In terms of performance, regular LDPC Gallager codes are only slightly inferior to parallel and serial concatenated convolutional codes. However, in their *irregular* version, they exhibit an impressive performance outperforming the best known turbo codes [8]–[10].

LDPC codes are linear codes specified by a *sparse* parity-check matrix. A (d_v, d_c)-regular LDPC code, as originally defined by Gallager, is a binary linear code such that every code bit participates in exactly d_v parity-check equations, and every check equation involves exactly d_c code bits. In other words, the corresponding parity-check matrix $\mathbf{H}$ has d_v ones in each column and d_c ones in each row. As originally suggested by Tanner [11], LDPC codes are well represented by *bipartite graphs* in which one set of nodes, the *variable nodes*, represents the elements of a codeword, and the other set of nodes, the *check nodes*, corresponds to the set of parity-check constraints which define the code. In the following, we will denote these bipartite graphs as *Tanner graphs*. Regular LDPC codes are such that all nodes of the same type have an equal number of edges. On the contrary, for irregular LDPC codes, the node degree of each node in each set is not equal, but chosen according to an optimized distribution [8]–[10].

As already mentioned, some irregular LDPC codes have better performance with respect to turbo codes for equal codeword length. In addition, they have other advantages over turbo codes. In fact, *belief propagation* (BP), the iterative decoding algorithm for LDPC codes which works on the code Tanner graph, can be fully parallelized and potentially implemented at high speed. Moreover, in [12], low-complexity decoders that closely approximate BP in performance have been designed for these codes, extending the original work in [7]. In the following, all these decoders, i.e., BP and the low-complexity decoders described in [12], will be referred to as *message-passing* decoders. On the negative side, one major drawback of LDPC codes is represented by their high encoding complexity [13].

In this paper, we will describe and analyze two classes of coding schemes, originally proposed in [14], which can be easily decoded at high speed by means of message-passing decoders. The first class is represented by single convolutional codes with some specific algebraic properties which ensure that message-passing decoders can operate successfully on the code Tanner graph. In this case, a message-passing schedule, specifically tailored for this class of codes, will be described. Since message-passing decoding algorithms are very simple and characterized by a decoding complexity which does not directly depend on the code constraint length, they can be used to decode convolutional codes with a large constraint length, and hence, potentially characterized by a large free distance.[1]

Paper approved by R. D. Wesel, the Editor for Coding and Communication Theory of the IEEE Communications Society. Manuscript received October 11, 2002; revised January 2, 2004, and April 9, 2004. This paper was presented in part at the Third International Symposium on Turbo Codes and Related Topics, Brest, France, September 2003.

The author is with the Dipartimento di Ingegneria dell'Informazione, Università di Parma, I-43100 Parma, Italy (e-mail: giulio@unipr.it).

Digital Object Identifier 10.1109/TCOMM.2004.836566

[1]The decoding complexity of a BCJR or a Viterbi algorithm grows exponentially with the code constraint length, therefore, these algorithms cannot be practically used for codes with a very large free distance.

The second class of codes is based on code concatenation. The resulting coding schemes and the corresponding decoding algorithms combine the advantages of turbo codes and LDPC codes. In fact, the coding structures are very simply based on the parallel or serial concatenations of simple convolutional codes through interleavers, in which the component convolutional codes are constrained to have the above-mentioned specific algebraic properties which ensure that message-passing decoders can work successfully on the Tanner graph of the overall code. In this way, combining the coding structure of parallel or serial concatenated codes and the decoding algorithm for LDPC codes, a new architecture results, which can be effectively implemented with simple coding and decoding operations. In terms of decoding complexity, the proposed schemes represent a valid alternative to classical turbo codes and give a solution to the *encoding problem* of LDPC codes [13].

The proposed codes are different from low-density convolutional codes proposed in [15] which are time-varying, periodical, binary convolutional codes.

II. Background: Tanner Graphs and Belief Propagation

Message-passing decoding algorithms for LDPC codes are based on an iterative exchange of messages along the edges of a code Tanner graph [7], [11], [16]. This graph, which can be drawn by direct inspection of a parity-check matrix $\mathbf{H}$ of the code, is composed of two classes of nodes: *variable* nodes corresponding to the code bits, and *check* nodes corresponding to the code constraints. Check nodes are connected by *edges* to variable nodes on which they depend. A cycle is a closed path in the graph, and its length is defined as the corresponding number of path edges. The length of the shortest cycle is the *girth* of the graph.

We now review the BP decoding algorithm. Let us denote by Λ a suitable message propagating on an edge of the code Tanner graph. This message represents the log-likelihood ratio related to the code bit corresponding to the variable node from which the considered edge originates. At the mth iteration, we denote by $\Lambda^{(m,v\rightarrow c)}$ a message sent from a variable node to a check node, and by $\Lambda^{(m,c\rightarrow v)}$ a message in the opposite direction. A variable node of degree d_v receives and processes the messages $\Lambda_i^{(m-1,c\rightarrow v)}$, $i = 1,\ldots,d_v$, and sends back to its jth ($j = 1,\ldots,d_v$) neighboring check node the message [7], [12]

$$\Lambda_j^{(m,v\rightarrow c)} = \Lambda^0 + \sum_{\substack{i=1\\ i\neq j}}^{d_v} \Lambda_i^{(m-1,c\rightarrow v)} \tag{1}$$

where Λ^0 is the initial message received by the considered variable node as a function of the channel output corresponding to the considered code bit. When $m = 1$, the variable node simply propagates its initial received message Λ^0. It may be observed that the message $\Lambda_j^{(m,v\rightarrow c)}$ in (1) does not depend on the message $\Lambda_j^{(m-1,c\rightarrow v)}$ previously received on the same edge, i.e., only *extrinsic information* is exchanged.

A check node of degree d_c receives and processes the messages $\Lambda_i^{(m,v\rightarrow c)}$, $i = 1,\ldots,d_c$, and sends back to its jth ($j = 1,\ldots,d_c$) neighboring variable node the message [7], [12]

$$\Lambda_j^{(m,c\rightarrow v)} = 2\tanh^{-1}\left\{\prod_{\substack{i=1\\ i\neq j}}^{d_c} \tanh\left\{\frac{1}{2}\Lambda_i^{(m,v\rightarrow c)}\right\}\right\}. \tag{2}$$

A simplified min-sum version of the updating rule (2) is described in [17]. The decoding algorithm proceeds iteratively until the code parity-check constraints are all verified or a maximum number of iterations is reached.

Although this decoding algorithm is provably optimal for bipartite graphs without cycles [12], in practice, it is necessary to only avoid cycles of length up to four to attain good performance [10]. Other message-passing algorithms can be devised based on different types of messages, possibly taking on values in some finite message alphabet, and different updating rules [12]. All these algorithms can be applied to decode the codes described in Sections III–V.

A message-passing schedule in a factor graph is the specification of the order in which messages are updated. In general, the so-called *flooding schedule* is adopted to decode LDPC codes [18]. In this case, in each iteration, all variable nodes, and subsequently, all check nodes, pass new messages to their neighbors. As can be easily understood, this schedule is well suited for a fully parallel implementation of the decoder.

III. Single Convolutional Code

In this section, we describe a class of convolutional codes which can be decoded by means of the described message-passing algorithms. We consider a single convolutional code truncated to a block code of a given codeword length. Tailbiting can be used to convert the convolutional code to a block code with no rate loss [19]. As mentioned in the previous section, a message-passing decoding algorithm operating on a code Tanner graph is an effective technique if the graph girth is at least six. In the following, we analyze the conditions that a convolutional code must satisfy in order to have a Tanner graph without cycles of length four.[2]

The girth of a code Tanner graph can be easily determined by analyzing the corresponding code parity-check matrix. In fact, a bipartite graph has girth of at least six if and only if (iff) the corresponding parity-check matrix *has no rectangles*, i.e., four one's in two separate rows which define the corners of a rectangle. This property can be easily verified by observing that if a rectangle is present, the row and column indexes of its corners determine the variable and check nodes connected by a cycle of length four.

The condition for the absence of cycles of length four can be directly translated into a condition on the code parity-check equations. For illustration purposes, rate-$k/(k+1)$ systematic

[2]We emphasize that in this paper, the term "Tanner graph" is used to denote a graph directly derived from the code parity-check matrix and without the explicit representation of hidden (state) variable nodes present in the so-called Wiberg-type graphs [17].

codes are considered, but the conditions being described can be easily generalized to rate-k/n convolutional codes.

A rate-$k/(k+1)$ recursive systematic convolutional (RSC) code is described by a polynomial generator matrix of the form

$$\mathbf{G}(D) = [\mathbf{I}_k \mid \mathbf{g}(D)] \tag{3}$$

where $\mathbf{I}_k$ is a $k \times k$ identity matrix, D is a unit-delay operator, and $\mathbf{g}(D)$ is a column vector of k elements of the form

$$\mathbf{g}(D) = \left[\frac{a_1(D)}{b(D)}, \frac{a_2(D)}{b(D)}, \ldots, \frac{a_k(D)}{b(D)}\right]^T \tag{4}$$

where $a_l(D)$, $l = 1, 2, \ldots, k$, and $b(D)$ denote the feedforward and feedback polynomials, respectively. These polynomials may be expressed in the form[3]

$$\begin{aligned} a_l(D) &= \sum_{j=1}^{J_l} D^{\alpha_{l,j}}, \quad l = 1, 2, \ldots, k \\ b(D) &= \sum_{j=1}^{J_{k+1}} D^{\beta_j} \end{aligned} \tag{5}$$

where J_l, $\alpha_{l,j}$, and β_j denote suitable integers which specify the considered code.

We may also describe a rate-$k/(k+1)$ RSC code by using the corresponding parity-check equation at discrete time m. This description has an immediate visual impact on the code Tanner graph. The encoder for the code described by (3) receives, at discrete time m, the information bits $i_{1,m}, i_{2,m}, \ldots, i_{k,m}$, and produces, assuming without loss of generality $\beta_1 = 0$, a parity bit p_m given by the following parity-check equation:

$$p_m = \sum_{j=2}^{J_{k+1}} p_{m-\beta_j} + \sum_{l=1}^{k} \sum_{j=1}^{J_l} i_{l,m-\alpha_{l,j}}, \quad m = 0, 1, 2 \ldots N-1 \tag{6}$$

where N is the total number of transmitted parity bits and it is understood that, if the lower limit of a sum is greater than the upper limit, no terms are summed. Therefore, $J_1, J_2, \ldots, J_k$, and J_{k+1} denote the number of information and parity bits in each parity check, respectively. In order to simplify the notation, we collect integers $\alpha_{l,j}$ and β_j in the following vectors: $\boldsymbol{\alpha}_l = (\alpha_{l,1}, \alpha_{l,2}, \ldots, \alpha_{J_l})$, $l = 1, 2, \ldots, k$, and $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_{J_{k+1}})$.

The corresponding Tanner graph can be easily built. In fact, for each discrete-time instant m, we have $k+1$ variable nodes corresponding to the information bits $i_{1,m}, i_{2,m}, \ldots, i_{k,m}$ and the parity bit p_m, and a check node representing (6). This check node will be connected to some variable nodes, according to (6). For a given number of decoding iterations, the complexity of a message-passing algorithm working on this graph depends on the number of edges in the graph, and therefore, on $J = J_1 + J_2 + \cdots + J_k + J_{k+1}$, and is independent of the code-constraint length.

Proposition 1: For a systematic rate-$k/(k+1)$ convolutional code, in order to ensure that the bipartite graph has girth at least six, a necessary and sufficient condition is that the index differences $\alpha_{l,j} - \alpha_{l,i}$ ($j > i$, $i, j = 1, 2, \ldots, J_l$, $l = 1, 2, \ldots, k$), and $\beta_j - \beta_i$ ($j > i$, $i, j = 1, 2, \ldots, J_{k+1}$) are all distinct. □

In particular, for a rate-1/2 code, the following proposition holds.

Proposition 2: For a systematic rate-1/2 code:

1) if $J_1 = 1$ ($J_1 = 2$) and $J_2 = 2$ ($J_2 = 1$), the bipartite graph has no cycles;
2) if $J_1 = 2$ and $J_2 = 2$, denoting by $\Delta\alpha \triangleq \alpha_{1,2} - \alpha_{1,1}$ and $\Delta\beta \triangleq \beta_2 - \beta_1$, the graph has girth four if $\Delta\alpha = \Delta\beta$, six if $\Delta\alpha \neq \Delta\beta$ and $\Delta\alpha = 2\Delta\beta$ or $2\Delta\alpha = \Delta\beta$, and otherwise the graph has girth eight;
3) if $J_1 \geq 2$ or $J_2 \geq 2$, the graph can have cycles of at most length six iff all differences $\alpha_{1,j} - \alpha_{1,p}$, $\beta_n - \beta_m$ are distinct.

□

For a general rate-k/n convolutional code, possibly punctured, similar conditions may be derived. As an example, for a systematic convolutional code of rate-$1/n$, in the absence of puncturing, the following proposition holds.

Proposition 3: Considering a systematic convolutional code of rate-$1/n$ as the interlacing of the parity sequences of $n-1$ rate-1/2 subcodes operating on the same information sequence, the condition for the absence of cycles of length four is that the above-mentioned index differences computed on all the rate-1/2 subcodes must be different. □

The proofs of these propositions are omitted for the reasons explained in the following *Remark 1*.

Example 1: Let us consider a rate-1/2 systematic recursive code with generator matrix

$$\mathbf{G}(D) = \left[1, \frac{1+D}{1+D+D^2}\right]. \tag{7}$$

In this case, $J_1 = 2$ and $J_2 = 3$, and the code is defined by the following two vectors: $\boldsymbol{\alpha}_1 = (0, 1)$ and $\boldsymbol{\beta} = (0, 1, 2)$. For this code, the described conditions are not satisfied, because $\beta_3 - \beta_2 = \beta_2 - \beta_1 = \alpha_{1,2} - \alpha_{1,1} = 1$. Hence, the Tanner graph has girth four, as can be easily verified. □

Example 2: Let us now consider a rate-1/2 systematic recursive code with generator matrix

$$\mathbf{G}(D) = \left[1, \frac{1+D^2}{1+D^3}\right]. \tag{8}$$

This code is characterized by $J_1 = 2$, $J_2 = 2$, $\boldsymbol{\alpha}_1 = (0, 2)$, and $\boldsymbol{\beta} = (0, 3)$. According to *Proposition 2*, the corresponding Tanner graph has girth eight. □

Remark 1: Nonrecursive systematic codes are a special case of the codes described in this section. In this case, the conditions in *Propositions 1* and *3* for the absence of cycles of length four (or, equivalently, for the absence of rectangles in the parity-check matrix) define the so-called *convolutional self-orthogonal codes* (CSOCs) [20]–[22]. These codes form a class of nonrecursive systematic codes adopted for satellite communications in the 1960s and 1970s since they admit simple decoding schemes, namely, the so-called hard-input majority-logic decoding and soft-input threshold decoding [20], [22]. The conditions in *Propositions 1* and *3* were recognized as necessary for the applicability of majority logic and threshold decoding [20],

[3]In the following, we employ the usual symbols $+$ and $\sum$ to denote modulo-2 addition.

[21]. However, this is not surprising, since it can be shown that threshold decoding coincides with the previously described BP algorithm when a single iteration is performed. Similarly, it can be shown that majority-logic decoding is related with the first iteration of the so-called Gallager B algorithm described in [7] and [12]. The codes described in this section may be viewed as a generalization of CSOCs to recursive codes. Proofs of *Propositions 1–3* can be obtained as a straightforward extension of the proofs in [21] for conventional CSOCs, and are therefore omitted. □

Remark 2: The code description is not unique, nor is the corresponding code Tanner graph. As a consequence, even if a given generator matrix does not comply with the conditions for the absence of cycles of length four, it could be possible to find an equivalent code description that does. For instance, this is the case of the code in *Example 1*. In fact, multiplying numerator and denominator by the common factor $1+D$, we obtain the equivalent generator matrix given in (8), whose corresponding Tanner graph has girth eight. This code and its equivalent descriptions will be reconsidered in the numerical results. □

Schedule: When used to decode convolutional codes, message-passing algorithms can also adopt a schedule different from that previously described and usually adopted for LDPC codes. This schedule, referred to as the *wave schedule*, is made possible by the fact that the parity-check matrix of a convolutional code is block triangular, and by the relevant structure of the code Tanner graph. In fact, at each iteration, after variable and check nodes up to discrete-time m have been processed, the edges connecting "future" check nodes may be immediately updated without waiting for the successive iteration. In practice, each check node is "activated" when it can benefit from the updating, at the same iteration, of the messages coming from previous check and variable nodes. This schedule is pictorially exemplified in Fig. 1, with reference to the Tanner graph of a rate-1/2 systematic convolutional code with parity-check equation $p_m = i_m + i_{m-1}$. In the figure, three successive steps of this schedule are shown under the letters (a), (b), and (c). Although this new schedule allows a speed up of the convergence process, it is not compatible with a fully parallel implementation of the decoder. For high-speed communication systems, the flooding schedule could be preferred.

IV. Concatenated Schemes

In this section, we discuss the concatenation of the convolutional codes previously described. The resulting coding schemes are classical turbo codes or serially concatenated codes, but may be decoded as LDPC codes. For this reason, these new concatenated schemes are nicknamed "turbo Gallager codes" (TGCs). More generally, code networks can be designed by concatenating a few convolutional component codes in mixed parallel and serial configurations [5], [23].

TGC are based on the following two key concepts:

1) at the encoder, the use of recursive CSOCs in code networks as component codes;
2) at the decoder, the use of a message-passing algorithm which works on the Tanner graph of the *overall code* by adopting the flooding or the wave schedule.

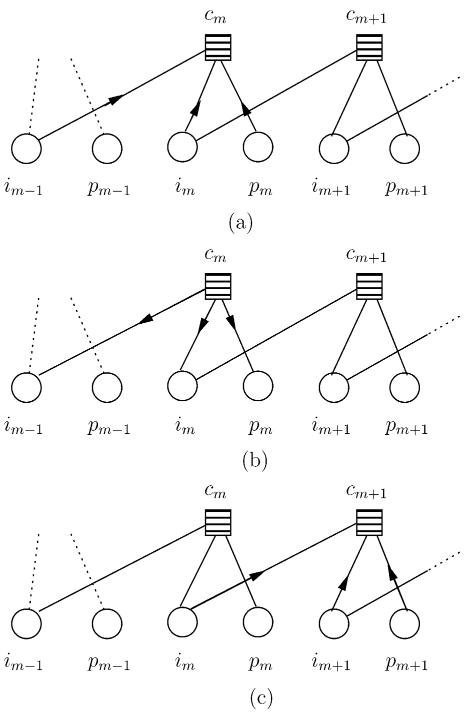


Fig. 1. Wave schedule.

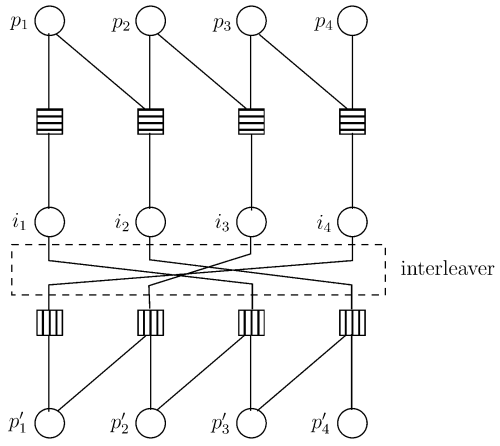


Fig. 2. Overall Tanner graph for a turbo code.

Let us consider the overall Tanner graph of a turbo code. In Fig. 2, we show, as an example, the graph for a rate-1/3 code. The component codes are rate-1/2 systematic codes with $\boldsymbol{\alpha}_1 = (0)$, $\boldsymbol{\beta} = (0, 1)$, and, for simplicity, the code termination is ignored. In the upper part of the graph, the information bits i_m and the parity bits p_m of the first component code are connected with the corresponding check nodes. The information bits are also connected, in a permuted order, with the lower check nodes related to the second component code having parity bits p'_m. The decoding process can be performed by means of a message-passing decoder working on the Tanner graph of the overall code. In this case, the variable nodes *of both codes*, and subsequently, the check nodes *of both codes*, are activated simultaneously. Alternatively, a "turbo-like" decoder such as that described in [2] could be used, in which each soft-output decoder is based on the message-passing algorithm working on the Tanner graph of the corresponding component code, and both decoders iteratively exchange soft information.[4] For a given performance, the convergence speed of a message-passing decoder working on the Tanner graph of the overall code with the flooding or wave schedule was observed to be greater than that of the above-mentioned turbo-like decoder. In fact, each component decoder of this turbo-like scheme must perform some inner iterations in order to pass reliable information to the other decoder. Then, additional iterations on the overall scheme are necessary.

Note that even if the graphs of each component code have girth six, the overall Tanner graph may have girth four, depending on the interleaver used. As an example, consider a turbo code composed of rate-1/2 convolutional codes. In this case, a sufficient condition to avoid the appearance of cycles of length four in a parallel concatenation is that information symbols which differ for less than $\max_{i,j}\{\alpha_{1,i}-\alpha_{1,j}\}$ have a distance, after the interleaver, greater than or equal to $\max_{i,j}\{\alpha_{1,i}-\alpha_{1,j}\}$. This may be obtained by the use of an S-*random* interleaver with $S=\max_{i,j}\{\alpha_{1,i}-\alpha_{1,j}\}$ [24].

Turbo codes based on CSOCs as component codes were previously proposed in [25], where, however, nonrecursive CSOCs were adopted, and the decoding scheme was "turbo-like" using threshold decoders as soft-output component decoders. The performance of this scheme has a twofold limitation: 1) the use of nonrecursive codes prevents the possibility of an interleaver gain; and 2) the suboptimal threshold decoders further limit the overall performance. This limitation may be overcome by the proposed schemes based on recursive codes and a more efficient decoding.

As a design criterion to select the component codes as well as the type of interleaver, puncturing, and concatenation (parallel, serial, or mixed), the method of density evolution can be used. Density evolution is a tool for jointly analyzing the code and the message-passing decoding algorithm, evaluating the relevant performance when averaged over the ensemble of codes with a common degree distribution of variable and check nodes [10], [12]. This method, which assumes the absence of cycles, analyzes the evolution of the probability density functions of the messages propagating in the graph during decoding, with the aim of determining the critical channel parameter (the so-called *threshold*) which separates the region where reliable transmission is possible from that where it is not [12]. Notice that for TGCs, the condition of absence of cycles for increasing codeword length is not satisfied. Hence, no rigorous claim on the achievable iterative threshold can be made. Nevertheless, density evolution can provide useful information in the code design.

The degree distribution of variable and check nodes, necessary for the application of the density evolution method, can be concisely described by two polynomials $\lambda(x)$ and $\rho(x)$ defined in [9] and [10]. For TGCs, these two polynomials depend on the type of concatenation, interleaver, puncturing (if any), and component codes.

Example 3: Let us consider a rate-1/2 turbo code obtained from two identical rate-1/2 RSC codes by means of puncturing. Let us assume that the interleaver is odd–odd, i.e., bits in odd (even) position remain in odd (even) position after the permutation,[5] and that the odd parity bits of the first encoder and the even parity bits of the second encoder are punctured. When a bit is punctured, the corresponding variable node in the graph disappears, along with the neighboring check nodes and the edges connected to them. As a consequence, by means of simple considerations on the overall Tanner graph, it can be shown that the corresponding degree polynomials are

$$\begin{aligned}\lambda(x) &= \frac{J_1}{J_1+J_2}x^{J_1-1}+\frac{J_2}{J_1+J_2}x^{J_2-1}\\ \rho(x) &= x^{J_1+J_2-1}.\end{aligned} \tag{9}$$

By properly choosing the integers J_1 and J_2, simple irregular codes can be obtained. However, if $J_1=J_2$, a regular code results. As an example, if $J_1=J_2=3$, a (3,6) regular LDPC code is obtained.

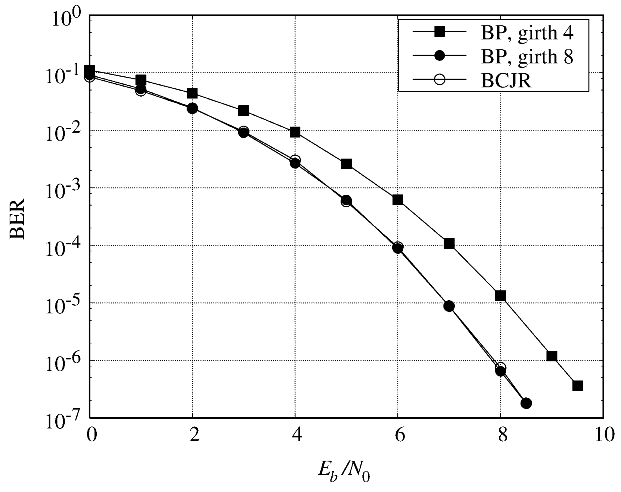


Fig. 3. BER of the convolutional code in *Examples 1* and *2*, by using the BCJR and the BP algorithm.

V. Numerical Results

Computer simulations are used to assess the performance of the proposed codes, whether concatenated or not, in terms of bit-error rate (BER) versus E_b/N_0, E_b being the received signal energy per information bit, and $N_0/2$ the two-sided noise power spectral density. For TGCs, density evolution is used to select the component codes. To this purpose, discretized density evolution [10] is considered with 512-level quantization in the range from -32 to 32.[6] Finally, a comparison between the flooding and the wave schedules in terms of convergence speed is reported.

Single Convolutional Code: The application of the BP algorithm to a single convolutional code is first analyzed, and in Fig. 3, the code in *Examples 1* and *2* is considered (see also *Remark 2*). In this figure, the performance of the optimal BCJR algorithm is also given as a benchmark. For the BP algorithm, computer simulations were performed by using early detection of convergence, i.e., at each iteration the decoder determines if

[4]This turbo-like scheme corresponds to a particular message-passing schedule in the overall graph. In fact, in this case, messages are exchanged in the upper or in the lower part of the graph alternatively.

[5]Examples of such an interleaver are described in [2] and [26]. A dithered relative prime (DRP) interleaver [27] can be also odd–odd for particular choices of the dither functions.

[6]We assume that the transmitted code symbols are ± 1.

TABLE I
THRESHOLDS FOR A RATE-1/2 TURBO CODE WITH EQUAL COMPONENT CODES, AN ODD–ODD INTERLEAVER AND AN ODD–EVEN PUNCTURING

J	J_1	J_2	E_b/N_0(dB)
3	1 (2)	2 (1)	10.712
4	2	2	3.277
5	2 (3)	3 (2)	1.078
6	2 (4)	4 (2)	0.797
7	2 (5)	5 (2)	0.863
8	2 (6)	6 (2)	1.046
9	2 (7)	7 (2)	1.262
10	2 (8)	8 (2)	1.484

checks are all satisfied, stopping decoding when they are. When the BP algorithm is applied to the Tanner graph with girth four derived from the generator matrix in (7), the performance does not approach the optimal one, even for a maximum allowed number of 50 iterations. On the contrary, on a Tanner graph with girth greater than four, such as the graph which derives from the generator matrix in (8) for the same code, the BP algorithm has a performance which is very close to the optimal one in two to three iterations (the relevant curve in Fig. 3 refers to ten maximum allowed iterations). In general, when the code Tanner graph has girth six, we observed in our simulations that the performance of the BCJR algorithm is always reached. On the contrary, when the code has girth four, the BER never converges to the optimal performance.

Concatenated Schemes: Code Design Based on Density Evolution: As a tool to perform the code design, the density evolution technique was used. The threshold of the signal-to-noise ratio (SNR), i.e., the value above which the expected fraction of incorrect messages approaches zero, assuming absence of cycles, is computed. As already mentioned, these threshold values can be used to select the component codes as well as the type of interleaver, puncturing, and code concatenation. As an example, let us consider the case of turbo codes, such as those described in *Example 3*. In this case, the threshold values are shown in Table I for different values of $J \triangleq J_1+J_2$, along with the corresponding optimal values of J_1 and J_2. It can be noted that an optimal value of J exists and corresponds to the couple $(J_1,J_2)=(2,4)$ (or equivalently, $(J_1,J_2)=(4,2)$). Hence, component codes with these values of J_1 and J_2 represent the better choice in this case. The corresponding threshold value is 0.797 dB, whereas the Shannon limit for binary inputs and rate-1/2 is 0.187 dB. Lower values of the threshold (0.584 dB) may be obtained by using different component codes, i.e., an *asymmetric* turbo code, in order to make the overall code more irregular. As a side result, we observed a significant increase of the threshold values when the component codes are nonrecursive, according to well-known results for turbo codes.

As a final comment, we summarize the results we obtained by applying the threshold analysis to serial concatenated schemes. We found that lower threshold values are obtained for a recursive inner code according to the results in [5]. As a further degree of freedom which can allow lower threshold values, code networks obtained by concatenating component codes in mixed parallel and serial configurations may be considered.

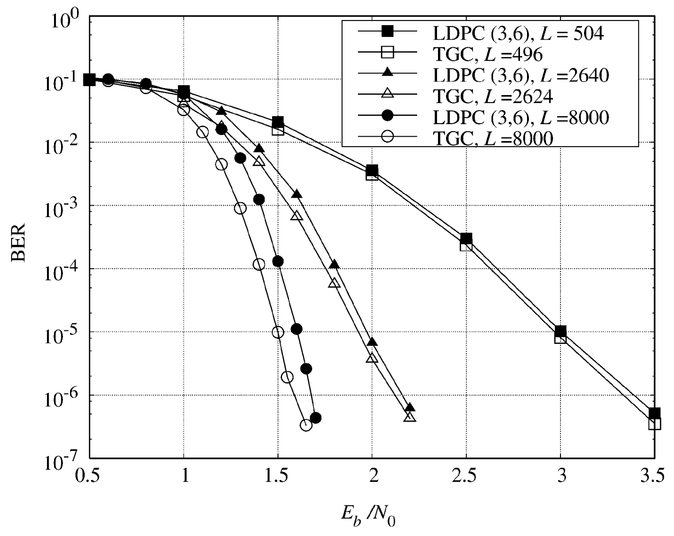


Fig. 4. BER of the considered TGCs and comparison with regular LDPC codes for different codeword lengths L.

Concatenated Schemes: BER Analysis: In *Example 3*, we observed that, with a proper choice of component codes, interleaver type, and puncturing, it is possible to obtain a TGC which is also a (3,6)-regular LDPC code. In Fig. 4, we compare the performance of these (3,6)-regular TGCs with that of classical (3,6)-regular LDPC codes [6], [28] for different codeword lengths L. For the proposed schemes, the component codes have $\boldsymbol{\alpha}_1=(0,3,4)$, $\boldsymbol{\beta}=(0,14,34)$, and we use tailbiting in both component codes, as described in [29] for recursive codes, in order to avoid a rate loss due to code termination. Note that no code search was performed—the component codes are only chosen following *Proposition 3* and simple intuitive considerations. Specifically, as both component codes are punctured, integers β_j must be all even (or all odd) in order to prevent all of the check nodes from disappearing. In addition, integers $\alpha_{1,i}$ are chosen small in order to reduce the probability of appearance of cycles of length four due to the interleaver, whereas larger values of the integers β_j are chosen in order to allow the propagation of a message to a wide region of the graph in a few iterations.

For TGCs, different codeword lengths, namely, $L=496$, 2624, and 8000, which correspond to interleaver lengths of 248, 1312, and 4000, respectively, are considered.[7] We emphasize the simplicity in obtaining codes with different codeword lengths. The component codes remain the same and only the interleaver size changes. Computer simulations were performed by using early detection of convergence. The maximum allowed number of iterations is 400 for TGCs, as well as for (3,6)-regular LDPC codes of length $L=504$, 2340, and 8000, reported in Fig. 4 as a comparison [28]. We observe that the considered TGCs perform slightly better than classical (3,6)-regular LDPC codes, despite their simpler construction and encoding.

In Fig. 5, we compare the performance of TGCs with that of the original rate-1/2 turbo code by Berrou and Glavieux (B&G)

[7]DRP interleavers with $M = 8$ are used [27].

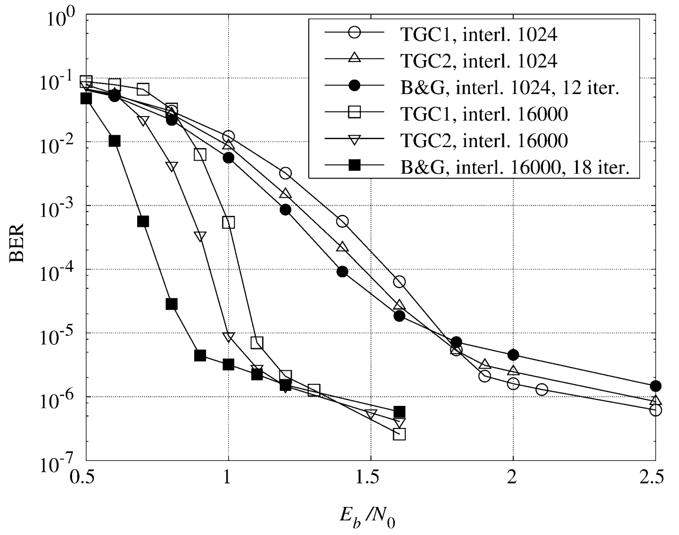


Fig. 5. BER of the considered TGCs and comparison with the original turbo code by B&G [2].

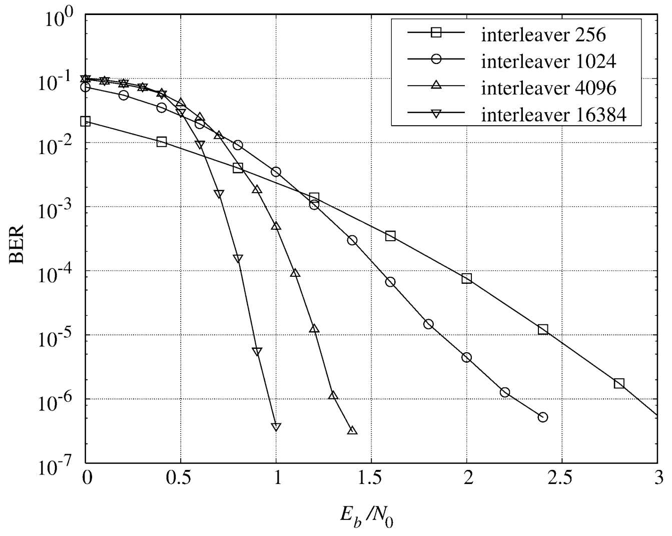


Fig. 6. BER of the considered TGCs in the case of serial concatenation.

[2]. Unlike the code considered in *Examples 1* and *2* and *Remark 2*, it can be shown that it is not possible to rearrange the generator matrix of the component code of the B&G turbo code in order to obtain a bipartite graph with girth greater than four. For this reason, we use different component codes for the considered TGCs. A symmetric TGC, such as that described in *Example 3*, is considered with component codes having $J_1 = 4$ and $J_2 = 2$, since these values correspond to the lowest threshold. This code is denoted TGC1 in Fig. 5. The identical component codes have $\boldsymbol{\alpha}_1 = (0, 3, 4, 13)$ and $\boldsymbol{\beta} = (0, 34)$. We also consider an asymmetric turbo code, denoted as TGC2 in Fig. 5, which corresponds to a threshold of 0.584 dB. In this case, one component code has $\boldsymbol{\alpha}_1 = (0, 3, 4, 13)$ and $\boldsymbol{\beta} = (0, 34)$, whereas the second one has $\boldsymbol{\alpha}_1 = (0, 3, 5, 9, 17, 33)$ and $\boldsymbol{\beta} = (0, 34)$. For these three turbo codes (B&G, TGC1, and TGC2), DRP interleavers with lengths 1024 $(M = 8)$ or 16 000 bits $(M = 16)$ are used. For TGC1 and TGC2, the maximum number of iterations is 200, for interleavers of length 1024, or 800, for interleavers of length 16 000. However, as shown below, the mean number of iterations really necessary is significantly lower. From Fig. 5, it can be observed that, despite the lower level of decoding complexity, TGC1 exhibits a performance degradation of less than 0.3 dB at a BER of 10^{-5}. This performance loss reduces to about 0.15 dB by using TGC2.

A serial code concatenation is considered in Fig. 6, where the performance of an overall code obtained by concatenating a systematic outer code with $\boldsymbol{\alpha}_1 = (0, 1)$ and $\boldsymbol{\beta} = (0)$ (hence, this code in nonrecursive), and a systematic recursive inner code with $\boldsymbol{\alpha}_1 = (0, 3, 4, 13)$ and $\boldsymbol{\beta} = (0, 34)$, is shown for different interleaver lengths. The overall rate of 1/3 is obtained by properly puncturing the inner code. Hence, the codeword length is 1.5 times the interleaver length. A density evolution analysis showed that, corresponding to the degree distribution of this code, the threshold is 0.697 dB. From the figure, it can be observed that by increasing the interleaver length, the performance tends to this limiting value.

Schedule: Finally, in Fig. 7, we compare the wave and flooding schedules in terms of mean value and standard deviation of the number of iterations until convergence versus the SNR. The considered TGC code is TGC1 from Fig. 2 with an interleaver length of 1024 bits. We observe that the wave schedule reduces the number of iterations. However, as already mentioned, this schedule is not compatible with a fully parallel decoder implementation—it represents a viable solution for serial implementations.

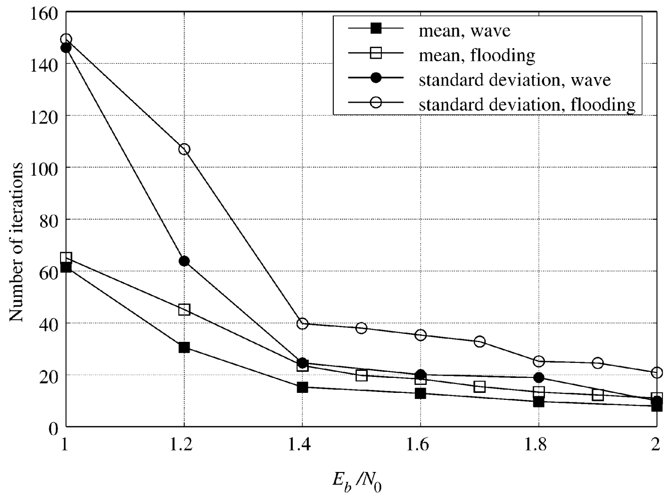


Fig. 7. Mean and standard deviation of the number of iterations for the wave and flooding schedules.

VI. Conclusion

In this paper, we showed that code networks, designed by concatenating convolutional codes in mixed serial and/or parallel configuration through interleavers, may be effectively decoded by means of the decoding algorithm used for LDPC Gallager codes and working on the code Tanner graph, provided that the component codes are chosen according to proper conditions which guarantee that the graph has girth at least six. For RSC codes of rate $1/n$ and $k/(k+1)$, these conditions are explicitly given, and their analogy with the conditions which define CSOCs were drawn. Hence, the proposed code networks combine the simple coding operations of turbo and serial concatenated codes with the fully parallel decoding implementation of LDPC codes. These codes were compared, in terms of BER performance, with classical regular LDPC codes and the original B&G turbo code showing that, despite the lower overall complexity, very powerful schemes may be obtained.

ACKNOWLEDGMENT

The author acknowledges the contribution of Dr. M. Bertinelli for his help in developing part of the computer programs, and that of M. Franceschini for the density evolution results.

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near-Shannon limit error-correcting coding and decoding: Turbo codes," in *Proc. Int. Conf. Communications*, 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, pp. 1261–1271, Oct. 1996.

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

[4] P. Roberston, E. Villebrun, and P. Hoeher, "Optimal and sub-optimal maximum *a posteriori* algorithms suitable for turbo decoding," *Eur. Trans. Telecommun.*, vol. 8, no. 2, pp. 119–125, Mar./Apr. 1997.

[5] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding," *IEEE Trans. Inform. Theory*, vol. 44, pp. 909–926, May 1998.

[6] D. J. C. MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399–431, Feb. 1999.

[7] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[8] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann, "Practical loss-resilient codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 569–584, Feb. 2001.

[9] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619–637, Feb. 2001.

[10] S.-Y. Chung, G. D. Forney, T. J. Richardson, and R. L. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, pp. 58–60, Feb. 2001.

[11] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 533–547, Sept. 1981.

[12] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, pp. 599–618, Feb. 2001.

[13] ——, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 638–656, Feb. 2001.

[14] G. Colavolpe, "Decoding of Concatenated Self-Orthogonal Convolutional Codes on Graphs," Italian Patent MI2002A001438, June 28, 2002, Int. Patent Applicat. PCT/EP03/06337, Int. Patent WO 2004/004134, June 16, 2003.

[15] A. J. Felström and K. S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Trans. Inform. Theory*, vol. 45, pp. 2181–2191, Sept. 1999.

[16] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Linköping Univ., Linköping, Sweden, 1996.

[17] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, pp. 498–519, Feb. 2001.

[18] F. R. Kschischang and B. J. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 219–231, Feb. 1998.

[19] H. H. Ma and J. K. Wolf, "On tail biting convolutional codes," *IEEE Trans. Commun.*, vol. COM-34, pp. 104–111, Feb. 1986.

[20] J. L. Massey, *Threshold Decoding*. Cambridge, MA: MIT Press, 1963.

[21] J. P. Robinson and A. J. Bernstein, "A class of binary recurrent codes with limited error propagation," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 106–113, Jan. 1967.

[22] G. C. Clark, Jr and J. B. Cain, *Error-Correction Coding for Digital Communications*, 3rd ed. New York: Plenum, 1988.

[23] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Soft-input soft-output modules for the construction and distributed iterative decoding of code networks," *Eur. Trans. Telecommun.*, vol. 9, no. 2, pp. 155–172, Mar./Apr. 1998.

[24] D. Divsalar and F. Pollara, "Turbo codes for PCS applications," in *Proc. IEEE Int. Conf. Communications*, Seattle, WA, June 1995, pp. 54–59.

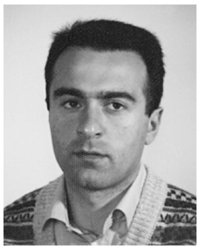
[25] S. Riedel and Y. V. Svirid, "Iterative ("turbo") decoding of threshold decodable codes," *Eur. Trans. Telecommun.*, vol. 6, no. 5, pp. 527–534, Sept./Oct. 1995.

[26] A. S. Barbulescu and S. S. Pietrobon, "Interleaver design for turbo codes," *Electron. Lett.*, vol. 30, pp. 2107–2108, Dec. 1994.

[27] S. Crozier and P. Guinand, "High-performance low-memory interleaver banks for turbo codes," in *Proc. IEEE Vehicular Technology Conf.*, Atlantic City, NJ, Oct. 2001, pp. 2394–2398.

[28] D. J. C. MacKay. Regular LDPC Online Database. [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/

[29] P. Ståhl, J. B. Anderson, and R. Johannesson, "A note on tailbiting codes and their feedback encoders," *IEEE Trans. Inform. Theory*, vol. 42, pp. 529–534, Feb. 2002.



Giulio Colavolpe was born in Cosenza, Italy, in 1969. He received the Dr. Ing. degree *(cum laude)* in telecommunication engineering from the University of Pisa, Pisa, Italy, in 1994 and the Ph.D. degree in information technology from the University of Parma, Parma, Italy, in 1998.

Since 1997, he has been with the University of Parma, where he is now an Associate Professor of Telecommunications. In 2000, he was a Visiting Scientist with the Insitut Eurécom, Valbonne, France. His main research interests include digital transmission theory, channel coding, and signal processing. His reesearch activity has led to more than 60 scientific publications in leading international journals and conference proceedings and several industrial patents.