Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding Schemes

Sergio Benedetto, *Senior Member, IEEE*, and Guido Montorsi, *Member, IEEE*

***Abstract*— A parallel concatenated coding scheme consists of two simple constituent systematic encoders linked by an interleaver. The input bits to the first encoder are scrambled by the interleaver before entering the second encoder. The codeword of the parallel concatenated code consists of the input bits to the first encoder followed by the parity check bits of both encoders. This construction can be generalized to any number of constituent codes. Parallel concatenated schemes employing two convolutional codes as constituent codes, in connection with an iterative decoding algorithm of complexity comparable to that of the constituent codes, have been recently shown to yield remarkable coding gains close to theoretical limits. They have been named, and are known as, "turbo codes." We propose a method to evaluate an upper bound to the bit error probability of a parallel concatenated coding scheme averaged over all interleavers of a given length. The analytical bounding technique is then used to shed some light on some crucial questions which have been floating around in the communications community since the proposal of turbo codes.**

***Index Terms*— Turbo codes, concatenated codes, iterative decoding.**

I. Introduction And Motivations

KNOWLEDGE of the fact that increasing the codeword length n of block codes (or the constraint length of convolutional codes) leads to better performance dates back to Shannon theory. It is also well known that the complexity of maximum-likelihood (ML) decoding algorithms increases with n, up to a point where decoding becomes physically unrealizable.

Thus the research in coding theory has seen many proposals aiming at constructing powerful codes with large equivalent block lengths structured so as to permit breaking the ML decoding into simpler partial decoding steps. Iterated codes [1], product codes and their extension [2], concatenated codes [3] and their generalized version [4], and large constraint-length convolutional codes with suboptimal decoding strategies, like sequential decoding, are nonexhaustive examples of these attempts, and some of them have been successfully employed in applications where large coding gains are required, such as, for example, deep-space communication.

Manuscript received February 27, 1995; revised October 17, 1995. This work was supported by European Space Agency and by CNR under Progetto Finalizzato Trasporti, subproject "Prometheus." The material in this paper was presented in part at the IEEE International Symposium on Information Theory, Whistler, BC, Canada, September 17–22, 1995.

The authors are with the Dipartimento di Elettronica, Politecnico di Torino, 10129 Torino, Italy.

Publisher Item Identifier S 0018-9448(96)01685-9.

The most recent successful attempt consists of the so-called "turbo codes" [5], whose astonishing performance has given rise to a large interest in the coding community. They are *parallel concatenated codes* (PCC) (see Fig. 1 for the case of block PCC) whose encoder is formed by two (or more) *constituent* systematic encoders joined through an interleaver. The input information bits feed the first encoder and, after having been interleaved by the interleaver, enter the second encoder. The codeword of the parallel concatenated code consists of the input bits to the first encoder followed by the parity check bits of both encoders.

The suboptimal iterative decoder is modular, and consists of a number of equal component blocks formed by concatenating the decoders of the constituent codes (CC) separated by the same interleaver used in the encoder. Each decoder produces weighted soft decoding of the input sequence. By increasing the number of decoding modules, and thus the number of decoding iterations, bit error probabilities as low as 10^{-5} at $E_b/N_0 = 0.0$ dB have been shown by simulation [6]. A version of turbo codes employing two eight-state convolutional codes as constituent codes, an interleaver of 32×32 bits and an iterative decoder performing two and a half iterations with a complexity of the order of nine times the ML Viterbi decoding of each constituent code is presently available on a chip yielding a measured bit error probability of $9.0 \cdot 10^{-7}$ at $E_b/N_0 = 3$ dB [7].

Bandwidth-efficient versions of turbo codes, compared to trellis-coded modulation schemes have also been proposed [8], as well as turbo codes based on block (instead of convolutional) codes [9], [10].

A careful examination of the literature shows that, rather than being a sudden apparition, turbo codes are the result of a clever intuition building on several concepts already available. We can cite in general the literature on product and concatenated codes in relation with the idea of parallel concatenation, the pioneering work on symbol-by-symbol maximum *a posteriori* decoding of linear codes [11] and the proposals in [12]–[14] of the soft-decisions Viterbi algorithm in relation to the way of implementing the iterative decoder. Very close to turbo codes are also the separable "filters" described in [15] to iteratively decode multidimensional codes.

As for the applications, it must be mentioned that PCC's, like all codes with very long codewords, suffer from one important drawback, namely the delay due to the interleaver and to the iterative decoding (as an example, the previously mentioned "chip" has a latency of 2318 bits). This prevents them from being used in applications where the combination of decoding delay and data rate leads to intolerable delays, like digital telephony. A broad range of applications still remains, such as digital audio and video broadcasting, data packet transmission, and space applications. It is also worthwhile mentioning that the interleaver inherently present in the PCC can prove beneficial for transmission on fading channels [8].

Since the successful proposal of turbo codes, neither a good theoretical explanation of the codes behavior/performance nor an adequate comprehension of the role and relative importance of the PCC ingredients (constituent codes and interleaver) have appeared.

In terms of performance of PCC's, apart from the measurements on the chip [7], what is known is essentially due to simulation [5], [8], [15]–[18], which, in itself, is not at all a simple task, as it requires a huge amount of computer time to obtain reliable results down to bit error probabilities like 10^{-6}.

As a consequence, a number of basic questions are still unanswered:

1) What is the performance of the ML decoder ?
2) What are the relative contributions of the constituent codes and of the interleaver length in determining the PCC performance ?
3) For a given interleaver length, how sensitive is the performance to the interleaver choice ?
4) How crucial is the use of recursive (feedback) systematic convolutional codes (as opposed to nonrecursive ones) as constituent codes of the PCC scheme ?
5) How close are the proposed suboptimal iterative decoding algorithms to ML decoding?

Answering these questions is certainly important from a theoretical point of view. Some of them, however, have significant practical relevance as well. For example, questions 1 and 5 can encourage (or discourage) the search for improved decoding algorithms, and question 2 may lead to the optimization of the PCC for given system constraints such as delay or complexity. Question 3, in turn, is related to the importance of the interleaver optimization, a topic which has already received some "cut-and-try" attention [17]–[19]. Finally, question 4 has been discussed in [20] where the authors seem to believe that recursive convolutional codes have superior merits in themselves, rather than only when used as CC of a PCC.

Formidable complexity obstacles discourage classical theoretical analysis of the PCC's. As an example, the code implemented in VLSI in [7], when seen as a whole convolutional code, consists of an equivalent time-varying convolutional code with 2^{1030} states, thus preventing any analytical evaluation of the main performance parameters.

In this paper, we will try to shed some light on the theoretical comprehension of PCC's. We will propose answers to the previous questions, some of which may be only preliminary yet indicative of the right direction.

In particular, we will define and evaluate an upper bound to the *average performance* of the ML soft decoder for a PCC, stemming from characteristics of the CC's. Owing to its definition, the average performance, expressed in terms of bit error probability, turns out to be independent from the particular interleaver used, and helps in assessing what can be gained with given CC's and with an interleaver of a given length.

We will also present simulation results for PCC's with differently chosen interleavers of the same length and compare them with the proposed bound. The results show that "random" interleavers offer performance close to the average ones evaluated through the upper bound, independent, to a large extent, from the particular interleaver. Bad interleavers are very easy to avoid in practice.

Moreover, we will show that recursive convolutional codes, although providing almost the same performance as nonrecursive codes when used alone, are indeed crucial when embedded in a PCC as CC's.

Finally, by comparing our bound on ML performance with simulation results based on iterative decoding, we will give heuristic evidence that the suboptimal algorithms can come very close to the optimum.

To help the reader, we will accompany the description with frequent examples, and will start from the simpler case of PCC schemes using block codes as CC's (parallel concatenated block code (PCBC)) leaving for the final sections the more complicated case of parallel concatenated convolutional codes (PCCC).

II. Notations and Definitions

Notations and definitions will be introduced for the case of parallel concatenated block codes, the extension to convolutional codes being straightforward.

Given an (n, k) systematic block code C, its well-known weight enumerating function (WEF) is

$$B^C(H) \triangleq \sum_{i=0}^{n} B_i H^i$$

where B_i is the (integer) number of codewords with Hamming weight (number of ones) i and H is a dummy variable. The WEF of a code can be used to compute the exact expression of the probability of undetected errors and an upper bound to the word error probability [21].

We define the *input-redundancy weight enumerating function* (IRWEF) of the code as

$$A^C(W, Z) \triangleq \sum_{w,j} A_{w,j} W^w Z^j$$

where $A_{w,j}$ denotes the (integer) number of codewords generated by an input information word of Hamming weight w whose parity check bits have Hamming weight j, so that the overall Hamming weight is $w + j$.

The IRWEF makes explicit in each term of the WEF the separate contributions of the information and of the parity-check bits to the total Hamming weight of the codewords, and thus provides additional information on the (Hamming) weight profile of the code. It will prove crucial in the following when dealing with parallel concatenated codes (PCC), since the two input words to the constituent encoders, the second being obtained by interleaving the first, share the same Hamming weight, so that the redundant bits generated by the two encoders derive from terms of the IRWEF with the same input weight w.

We mention also that the IRWEF characterizes the whole encoder, as it depends on both input information words and codewords, whereas the WEF only depends upon the code. As a consequence, the WEF is related to the word error probability of the code, whereas the IRWEF provides information on the bit error probability.

Obviously, the following relationship holds true:

$$B^C(H) = A^C(W = H, Z = H)$$

with

$$A^C(H,H) = \sum_{w,j} A_{w,j} H^{w+j} = \sum_k B_k H^k$$

where

$$B_k = \sum_{w+j=k} A_{w,j}\ .$$

Example 1: The $(7,4)$ Hamming code has the following WEF:

$$B^C(H) = 1 + 7H^3 + 7H^4 + H^7.$$

Splitting the contribution of the information and redundancy bits to the total codeword weight we obtain the IRWEF of the code

$$\begin{aligned} A^C(W,Z) = 1 &+ W(3Z^2 + Z^3) + W^2(3Z + 3Z^2) \\ &+ W^3(1 + 3Z) + W^4 Z^3. \end{aligned} \tag{1}$$

◇

Consider now the *conditional weight enumerating function* $A_w^C(Z)$ of the parity check bits generated by the code C corresponding to the input words of weight w. It can be obtained from the IRWEF as

$$A_w^C(Z) = \sum_j A_{w,j} Z^j = \frac{1}{w!} \cdot \left.\frac{\partial^w A^C(W,Z)}{\partial W^w}\right|_{W=0}$$

so that we can also write the inverse relationship

$$A^C(W,Z) = \sum_w W^w A_w^C(Z). \tag{2}$$

Both IRWEF and the $A_w^C(Z)$ can be used with the union bound to compute an upper bound to the bit error probability for ML soft decoding of the code over a channel with additive white Gaussian noise in the form

$$\begin{aligned} P_b(e) &\le \left.\frac{W}{k}\frac{\partial A^C(W,Z)}{\partial W}\right|_{W=Z=e^{-R_c E_b/N_0}} \\ &= \left.\sum_{w=1}^{k} \frac{w}{k} W^w A_w^C(Z)\right|_{W=Z=e^{-R_c E_b/N_0}} \\ &= \left.\sum_m D_m H^m\right|_{H=e^{-R_c E_b/N_0}} \end{aligned} \tag{3}$$

with

$$D_m \triangleq \sum_{j+w=m} \frac{w}{k} A_{w,j} \tag{4}$$

where R_c is the code rate.

The second and third line of (3) represent two equivalent expressions to bound the bit error probability. The first expression keeps distinct the contributions of information words with different weight w, whereas the second sums the contributions according to the overall weight m of the codeword through the coefficient D_m defined in (4).

A tighter bound can also be obtained from (3) [22] exploiting the inequality

$$\operatorname{erfc}(\sqrt{x+y}) \le \operatorname{erfc}(\sqrt{x})e^{-y}.$$

It assumes the form

$$\begin{aligned} P_b(e) \le \frac{W}{2k} &\operatorname{erfc}\left(\sqrt{\frac{d_{\min} R_c E_b}{N_0}}\right) \\ &\cdot e^{\frac{d_{\min} R_c E_b}{N_0}} \left.\frac{\partial A^C(W,Z)}{\partial W}\right|_{W=Z=e^{-R_c E_b/N_0}} \end{aligned} \tag{5}$$

which admits of course a further development like (3).

For parallel concatenated convolutional codes the information and code sequences are semi-infinite and, as a consequence, the summation (explicit in (3) and implicit in (5)) must be truncated to a finite value. For the bound in the second line of (3) the truncation involves the computation of the complete conditional weight distributions up to a given information weight, whereas for the bound in the third line the truncation leads to the computation of the weight multiplicities of the unconditional weight distribution up to a given overall weight of the code sequences. Computing algorithms and a comparison of the two approximations are discussed in [23].

Using a finite number of terms in (5) transforms the upper bound into the approximation

$$P_b(e) \approx \frac{1}{2} \sum_m D_m \operatorname{erfc}\left(\sqrt{m\frac{R_c E_b}{N_0}}\right). \tag{6}$$

In the following, all the results in terms of bit error probability will be computed using (6).

Example 2: The conditional WEF's of the Hamming code $(7,4)$ considered in the previous example are

$$\begin{aligned} A_0^C(Z) &= 1 \\ A_1^C(Z) &= 3Z^2 + Z^3 \\ A_2^C(Z) &= 3Z + 3Z^2 \\ A_3^C(Z) &= 1 + 3Z \\ A_4^C(Z) &= Z^3 \end{aligned}$$

so that the upper bound on the bit error probability computed through (6) and (4) becomes

$$\begin{aligned} P_b(e) \le \frac{3}{2}\operatorname{erfc}\left(\sqrt{\frac{3R_c E_b}{N_0}}\right) &+ 2\operatorname{erfc}\left(\sqrt{\frac{4R_c E_b}{N_0}}\right) \\ &+ \frac{1}{2}\operatorname{erfc}\left(\sqrt{\frac{7R_c E_b}{N_0}}\right). \end{aligned}$$

◇

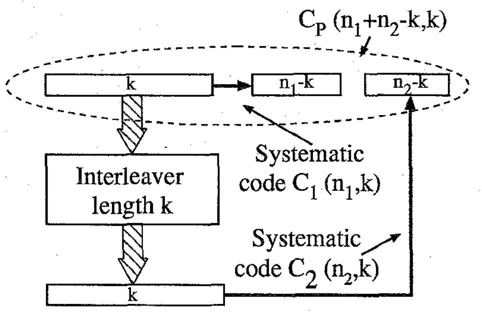
$C_P\ (n_1+n_2-k,k)$
k
n_1-k
n_2-k
Systematic code $C_1\ (n_1,k)$
Interleaver length k
Systematic code $C_2\ (n_2,k)$
k

Fig. 1. Parallel concatenated block code.

III. Parallel Concatenated Block Codes

Consider now a parallel concatenated block code (PCBC) obtained as in Fig. 1. Two linear systematic block codes C_1 with parameters (n_1, k) and C_2 with parameters (n_2, k), the constituent codes (CC), having in common the length k of the input information bits, are linked through an interleaver so that the information part of the second codeword is just a permuted version of the first one. The PCBC codeword is then formed by adding to the input bits the parity-check bits generated by the first and second encoder. The PCBC, that we denote as C_P, is then a $(n_1 + n_2 - k, k)$ linear code as the interleaver performs a linear operation on the input bits.

If w is the (Hamming) weight of the input word, and z_1 and z_2 the weights of the parity check bits introduced by the first and second encoders, respectively, the weight of the corresponding codeword of C_P will be $w + z_1 + z_2$.

We want now to obtain the IRWEF $A^{C_P}(W, Z)$ of C_P starting from the knowledge of those of the constituent codes. For a given interleaver, this operation is exceedingly complicated, as the redundant bits generated by the second encoder will not only depend on the weight of the input word, but also on how its bits have been permuted by the interleaver. The only viable solution, in theory, would be an exhaustive enumeration of all possible cases; in practice, this is an impossible achievement for large k, and this was precisely the reason for lengthy computer simulations.

To overcome this difficulty, we introduce an abstract interleaver called *uniform interleaver*, defined as follows.

Definition 1: A *uniform interleaver* of length k is a probabilistic device which maps a given input word of weight w into all distinct $\binom{k}{w}$ permutations of it with equal probability $1/\binom{k}{w}$. △

From the definition, it is apparent that the conditional weight enumerating function $A_w^{C_2}(Z)$ of the second code becomes independent from that of the first code thanks to the uniform randomization produced by the interleaver.

As a nice consequence of this, we can easily evaluate the conditional weight enumerating function of the PCBC which uses the uniform interleaver as the product, suitably normalized, of the two conditional weight enumerating functions of the constituent codes

$$A_w^{C_P}(Z) = \frac{A_w^{C_1}(Z) \cdot A_w^{C_2}(Z)}{\binom{k}{w}}. \tag{7}$$

Also, from (2) we obtain the IRWEF of the code C_P as

$$A^{C_P}(W, Z) = \sum_{w=1}^{k} W^w A_w^{C_P}(Z). \tag{8}$$

Example 3: The IRWEF of the PCBC constructed using as constituent codes two identical $(7, 4)$ Hamming codes can be obtained plugging the conditional WEF obtained in the previous example into (7) and applying (8)

$$\begin{aligned} A^{C_P}(W, Z) = 1 &+ W(2.25Z^4 + 1.5Z^5 + 0.25Z^6) + \\ &+ W^2(1.5Z^2 + 3Z^3 + 1.5Z^4) + \\ &+ W^3(0.25 + 1.5Z + 2.25Z^2) + W^4 Z^6. \end{aligned} \tag{9}$$

Notice in (9) the presence of fractional coefficients representing the multiplicity of the various terms. They are a direct consequence of the use of the uniform interleaver. ◇

The introduction of the uniform interleaver permits an easy derivation of the weight enumerating functions of the PCBC. However, in practice, one is confronted with deterministic interleavers, which give rise to one particular permutation of the input bits. So, what is the significance of the preceding definitions and equations ?

To answer this question, we prove now the main property of a PCBC which uses the uniform interleaver.

Theorem 1: Let $A^{C_{P_k}}(W, Z)$ be the IRWEF of the code C_{P_k} obtained using the particular interleaver I_k. Then

$$\mathrm{E}_k[A^{C_{P_k}}(W, Z)] = A^{C_P}(W, Z) \tag{10}$$

where E_k means expectation with respect to the whole class of interleavers. ▽

Proof of Theorem 1: The proof makes use of (2) through the following equality chain:

$$\begin{aligned} \mathrm{E}_k[A^{C_{P_k}}(W, Z)] &= \mathrm{E}_k\left[\sum_w W^w A_w^{C_{P_k}}(Z)\right] && (11) \\ &= \sum_w W^w \mathrm{E}_k\left[A_w^{C_{P_k}}(Z)\right] && (12) \\ &= \sum_w W^w A_w^{C_P}(Z) = A^{C_P}(W, Z). && (13) \end{aligned}$$

where the third equality comes from the definition of the uniform interleaver. **QED**

A second result, which comes as a corollary of the previous one from the linear dependency of (6) with respect to the conditional weight enumerating function, is the following.

Corollary 1: The upper bound computed using the IRWEF $A^{C_P}(W, Z)$ coincides with the average of the upper bounds obtainable with the whole class of deterministic interleavers. ▽

The corollary guarantees that, for each value of the signal-to-noise ratio, the performance obtained with the uniform interleaver are achievable by at least one deterministic interleaver.

Example 4: We can check the result (10) by computing, for the simple example of Hamming code previously examined, the IRWEF's of the PCBC's constructed using all the inter-

TABLE I
IRWEF OF THE PARALLEL CONCATENATED CODES BASED ON THE $(7,4)$ HAMMING CODE FOR ALL POSSIBLE INTERLEAVERS

Perm.	$A^{C_{P_k}}(W,Z)$
0123, 0321, 1023, 1320, 3021, 3120	$1+W(3Z^4+Z^6)+W^2(3Z^2+3Z^4)+W^3(1+3Z^2)+W^4Z^6$
0132, 0213, 0231, 0312, 1032, 1203, 1230, 1302, 2013, 2031, 2103, 2130, 2301, 2310, 3012, 3102, 3201, 3210	$1+W(2Z^4+2Z^5)+W^2(Z^2+4Z^3+Z^4)+W^3(2Z+2Z^2)+W^4Z^6$

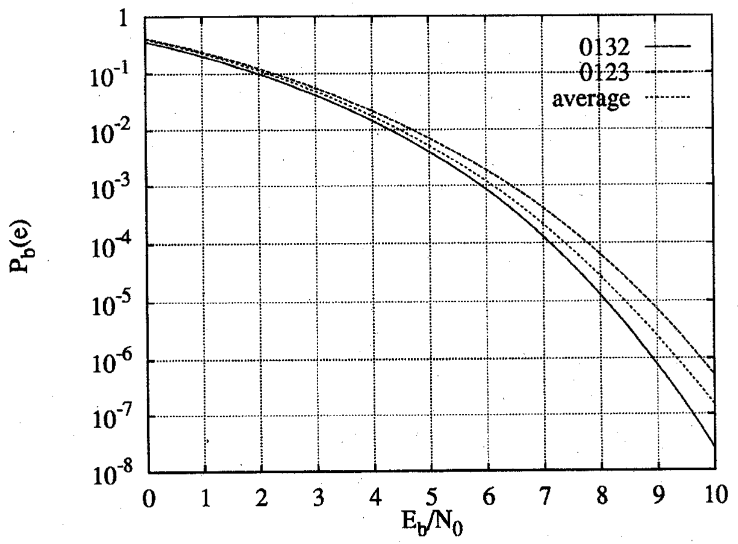

Fig. 2. Upper bounds for Example 4.

leavers originating from the $24 = 4!$ permutations of the input bits. The computed IRWEF's are reported in Table I.

From the table, it is apparent that, for this scheme, only two types of IRWEF are possible:

$$A^{C_{P_1}}(W,Z) = 1 + W(2Z^4+2Z^5) + W^2(Z^2+4Z^3+Z^4) + W^3(2Z+2Z^2) + W^4Z^6 \tag{14}$$

which derives from 18 different permutations and

$$A^{C_{P_2}}(W,Z) = 1 + W(3Z^4+Z^6) + W^2(3Z^2+3Z^4) + W^3(1+3Z^2) + W^4Z^6 \tag{15}$$

which appears six times. The average computed over all possible interleavers yields

$$A^{C_P}(W,Z) = \frac{18}{24}A^{C_{P_1}}(W,Z) + \frac{6}{24}A^{C_{P_2}}(W,Z)$$

which coincides with (9).

The upper bounds obtained substituting (14), (15), and (9) into (6) are plotted in Fig. 2. ◇

Let $n, k, t)$ denote the parameters of a t-error correcting (n,k) code. We have also analyzed the performance achieved by a PCBC using as CC the $(63,57,3)$ and the $(63,51,5)$ BCH codes. The interleavers have lengths $k = 57$ and 51,

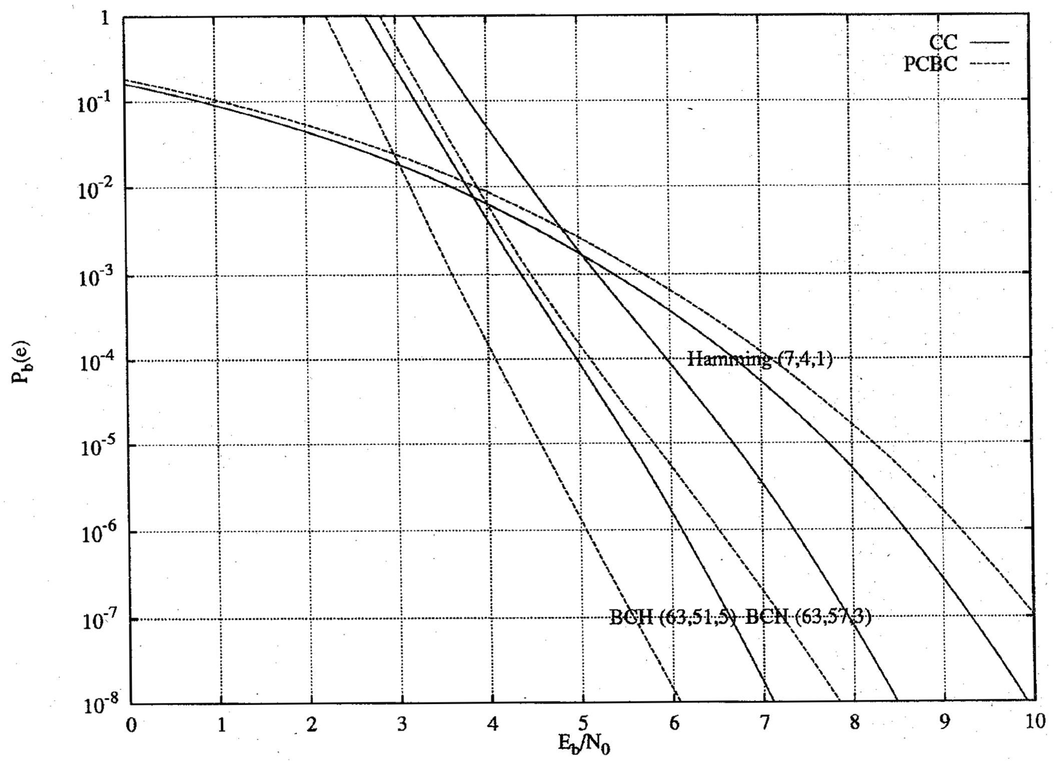

Fig. 3. Upper bounds to the bit error probability of two PCBC's using the $(63, 57, 3)$ and $(63, 51, 5)$ BCH codes as CC's and interleavers of lengths 57 and 51, respectively.

respectively. The results are reported in Fig. 3, where we also plot for comparison the results pertaining to the $(7,4)$ Hamming code of the previous example. In this case, the performance of the PCBC schemes improves over that of the CC's, and a gain of 1 dB is obtainable.

So far, we have used an interleaver with length k equal to that of the input word of the block code. It is straightforward to extend all previous results to the more general case where we use as basic codeword for a PCBC l consecutive codewords of the constituent code C_1 as in Fig. 4. In particular, the IRWEF of the new constituent (ln_1, lk) code C_1^l is given by

$$A^{C_1^l}(W,Z) = [A^{C_1}(W,Z)]^l \tag{16}$$

and similarly for the second constituent code C_2^l.

The conditional weight enumerating function of the new component code can still be obtained from the IRWEF through

$$A_w^{C_1^l}(Z) = \frac{1}{w!} \cdot \left. \frac{\partial^w A^{C_1^l}(W,Z)}{\partial W^w} \right|_{W=0}. \tag{17}$$

From the conditional weight enumerating functions of the two new constituent codes, owing to the property of the uniform interleaver of length lk, we obtain the conditional weight enumerating function of the $(l(n_1+n_2-k), lk)$ PCBC as

$$A_w^{C_P^l}(Z) = \frac{A_w^{C_1^l}(Z) \cdot A_w^{C_2^l}(Z)}{\binom{lk}{w}}. \tag{18}$$

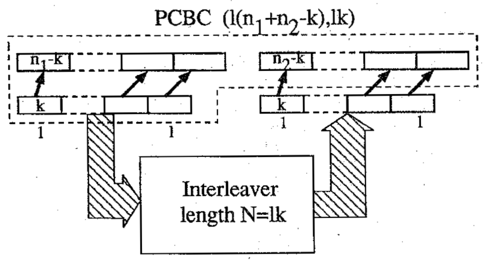

Fig. 4. Parallel concatenated block code with interleaver length lk.

From this point on, the performance of the new PCBC can be obtained as before through (6).

It is important to note that the PCBC obtained in this way is in some sense a generalization of the concept of product codes. In fact, a product code which does not transmit the parity checks on parity check bits can be obtained from the PCBC using a row-column interleaver of length $N = k^2$. Product codes with iterative decoding have been considered in [10].

Example 5: Taking as component code the Hamming code of previous examples we can build different PCBC of increasing complexity by simply grouping l consecutive codewords.

The first coefficients D_m defined in (4) for the resulting PCBC and $l = 1,2,3,4,5$ are reported in Table II. The effect of longer interleavers is clearly apparent from the table. Namely, the multiplicity of the terms which dominate the

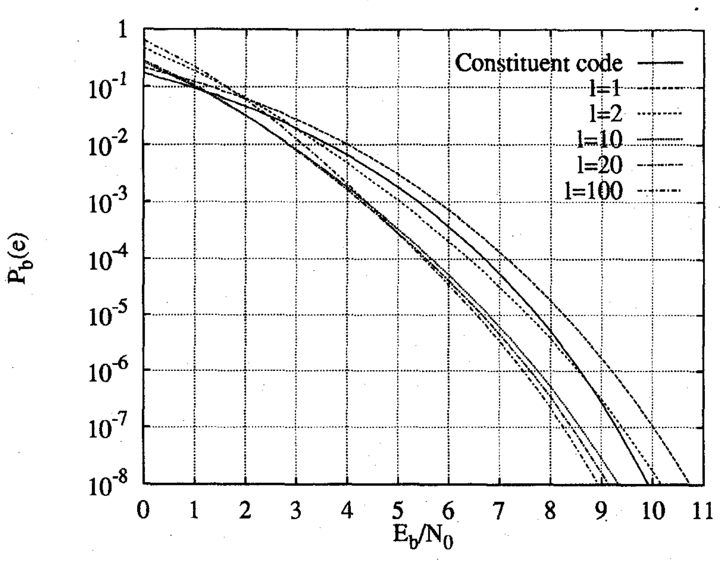

Fig. 5. Upper bounds to the bit error probability referring to Example 5. The different curves refer to interleavers of various lengths lk.

TABLE II
COEFFICIENTS D_m FOR THE EVALUATION OF THE BIT ERROR PROBABILITY OF THE PCBC OBTAINED FROM THE $(7,4)$ HAMMING CODE AND UNIFORM INTERLEAVERS GROUPING $1,2,3,4,5$ CONSECUTIVE INPUT WORDS

Hamming distance	l				
	1	2	3	4	5
3	0.1875	0.02678	0.01022	0.00535	0.00328
4	1.875	0.48214	0.26590	0.18214	0.13815
5	3.75	1.44642	1.06363	0.91071	0.82894
6	1.125	1.20535	0.95259	0.81597	0.73103
7	0.0625	3.83928	3.11412	2.77902	2.57720
8		9.96428	6.63116	5.44900	4.82972
9		23.7053	16.4537	13.6749	12.2146
10	1	33.3928	32.8558	30.8499	30.0122
11		21.5089	51.5566	52.5036	52.2468
12		12.3214	104.044	113.328	117.619
13		7.16071	192.425	219.463	231.044
14		4.40178	311.545	418.307	463.480
15		6.26785	379.400	737.146	894.473
16		1.23214	316.259	1207.75	1620.00
17		0.04464	227.272	2018.09	3016.75
18			148.853	3120.65	5357.44
19			95.6634	4271.70	9176.27
20		1	74.2149	4865.35	14962.0

performance (those with lower Hamming weight) decreases when l increases. Also, Hamming distances not present in the constituent codes show up. The minimum distance of the PCBC is still 3, as for the constituent codes.

Applying the upper bound (6) we obtain the results reported in Fig. 5, where the bit error probabilities for the considered code and various interleaver lengths $l = 1, 2, 10, 20, 100$ are plotted versus the signal-to-noise ratio E_b/N_0. For comparison, the curve of the constituent code is also reported.

The figure shows that a gain of 2 dB can be achieved increasing l at the expense of an increased delay. Also, we

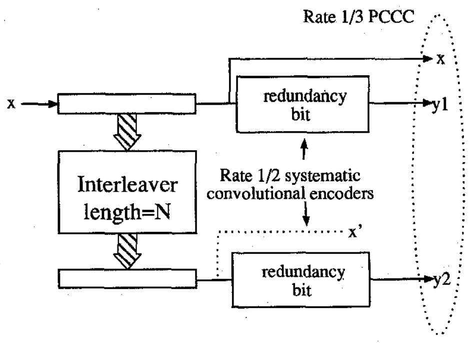

Fig. 6. Encoder structure of a PCCC.

notice from the results that the beneficial effect of increasing the interleaver length tends to decrease the larger l becomes. $\diamond$

The results obtained in the example illustrate an interesting phenomenon that will be observed again: performance improvements for a significant range of signal-to-noise ratios can be obtained with PCC's without increasing (or almost independently from) the minimum distance of the CC's. We ought to mention that independence of bit error probability performance from the code minimum distance for codes with large codewords length was already foreseen in [24].

IV. PARALLEL CONCATENATED CONVOLUTIONAL CODES

The first applications of parallel concatenated coding schemes used convolutional codes as constituent codes. The resulting codes have been named by the authors *turbo codes*, and the main reason for their successful implementation resides in the availability of efficient algorithms for soft iterative decoding [5], [7], [25]. In our context, we will call them parallel concatenated convolutional codes (PCCC). A block diagram showing how they work is presented in Fig. 6. The behavior is similar to that of a PCBC, the main difference being the fact that now the interleaver of length N does not

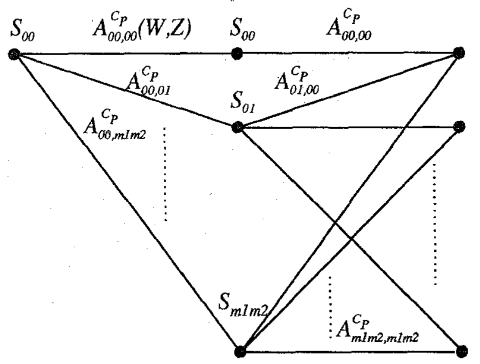

Fig. 7. Hyper-trellis for the PCCC.

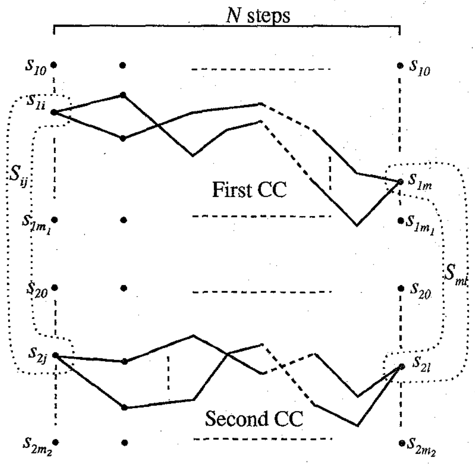

Fig. 8. Associations of states and paths of the trellises of constituent codes to states and paths of the PCCC hyper-trellis.

contain an integer number of input words, since the input sequences are infinite in length.[1] The figure represents the case of a rate 1/3 PCCC obtained from two rate 1/2 CC's. Several generalizations are possible. The number of CC's can be more than 2, and their rates can be different. Also, the final rate of the PCCC can be increased by puncturing the redundant bit sequences at the encoders outputs.

We will break the performance analysis of PCCC into two steps, the first performing an exact analysis to obtain the upper bound to the bit error probability, and the second showing how to obtain an accurate approximation which drastically reduces the complexity analysis.

A. Exact Analysis

Consider a PCCC formed by an interleaver of length N and two convolutional CC's C_1 and C_2 whose trellises have m_1 and m_2 states, respectively.

To examine the full dynamic of the PCCC, we must consider a hyper-trellis with $m_1 \cdot m_2$ states, like the one depicted in Fig. 7.

The state S_{ij} of the hyper-trellis corresponds to the pair of states s_{1i} and s_{2j} for the first and second constituent codes, respectively. Each branch $S_{ij} \rightarrow S_{ml}$ in the hyper-trellis represents all paths which start from the pair of states s_{1i}, s_{2j} and reach the pair s_{1m}, s_{2l} in N steps (see Fig. 8).

Thus when embedded in a PCCC using an interleaver of length N, the constituent convolutional codes contributions to the final codeword (or code sequence) derive from N-truncated versions of their input information sequences, or, equivalently, from trellis paths of length N.

Let $A^{C_P}_{ij,ml}(W, Z)$ be the label of the branch $S_{ij} \rightarrow S_{ml}$ of the hyper-trellis. It represents the IRWEF of the *equivalent parallel concatenated block code* obtained by enumerating the weights of all N-truncated sequences of the PCCC joining the hyper-states S_{ij} and S_{ml}. Once we know all these labels, the performance of the PCCC can be obtained through the standard transfer function bound approach [22] applied to the hyper-trellis.

[1] This observation, and the computing algorithms explained in the following, refer to the case of continuous transmission and decoding. When the PCCC is used as a block code with trellis termination, the same procedure described for PCBC can be applied.

To derive the branch labels of the hyper-trellis we can use the same procedure as that applied in Section III to parallel concatenated block codes, as we have seen that each label is indeed the IRWEF of a particular equivalent block code with information word length equal to N.[2]

We start with the conditional weight enumerating functions $A^{C_k}_{sn}(w, Z)$[3] of the equivalent block codes associated with the constituent convolutional codes. These functions enumerate all possible paths connecting the state s with the state n in N steps for the kth constituent encoder, $k = 1, 2$, and can be obtained from the transfer functions of the CC's as described in the Appendix. From them, we compute the conditional weight enumerating functions $A^{C_P}_{ij,ml}(w, Z)$ of the equivalent parallel concatenated block codes. Owing to the properties of the uniform interleaver, they are simply the normalized product of $A^{C_1}_{im}(w, Z)$ with $A^{C_2}_{jl}(w, Z)$

$$A^{C_P}_{ij,ml}(w, Z) = \frac{A^{C_1}_{im}(w, Z) \cdot A^{C_2}_{jl}(w, Z)}{\binom{N}{w}}. \tag{19}$$

Then, we obtain the IRWEF from the corresponding conditional weight enumerating functions through (2) and, finally, use the transfer function approach to get an upper bound to the bit error probability.

An example will clarify the whole procedure.

Example 6: Consider the PCCC obtained by linking two identical 2-state recursive convolutional constituent codes with an interleaver of length $N = 4$. The resulting encoder structure is depicted in Fig. 9.

First, we need to derive, using the algorithm described in the Appendix, the four conditional WEF's $A^C_{ij}(w, Z)$ that enumerate all the possible paths connecting in four steps the

[2] Actually, only the label $A^{C_P}_{00,00}(W, Z)$ describes a linear code containing the all "0" word; the other labels refer to cosets of this code. This has no effect on the analysis.

[3] For clarity, we slightly changed the notation of the conditional weight enumerating function by inserting the conditioning weight w within the parentheses.

TABLE III
CONDITIONAL WEIGHT ENUMERATING FUNCTIONS ENUMERATING ALL POSSIBLE PATHS CONNECTING STATE s_i WITH STATE s_j IN FOUR STEPS FOR THE CONSTITUENT ENCODER OF EXAMPLE 6

$A_{ij}^{C}(w,Z)$					
ij	$w=0$	$w=1$	$w=2$	$w=3$	$w=4$
00	1		$2Z^2+Z^3+3Z$		Z^2
01		$Z+Z^2+Z^3+Z^4$		$2Z^2+2Z^3$	
10		$1+Z+Z^2+Z^3$		$2Z+2Z^2$	
11	Z^4		$Z+2Z^2+3Z^3$		Z^2

TABLE IV
CONDITIONAL WEIGHT ENUMERATING FUNCTIONS LABELING THE HYPER-TRELLIS DESCRIBING THE DYNAMIC OF THE PCCC OF EXAMPLE 6

$A_{ij,ml}^{C_P}(w,Z)$					
ij,ml	$w=0$	$w=1$	$w=2$	$w=3$	$w=4$
00,00	1		$\frac{9Z^2+12Z^3+10Z^4+4Z^5+Z^6}{6}$		Z^4
00,01					
00,10					
00,11		$\frac{Z^2+2Z^3+3Z^4+4Z^5+3Z^6+2Z^7+Z^8}{4}$		$\frac{4Z^4+8Z^5+4Z^6}{4}$	
01,00					
01,01	Z^4		$\frac{3Z^2+8Z^3+14Z^4+8Z^5+3Z^6}{6}$		Z^4
01,10		$\frac{Z+2Z^2+3Z^3+4Z^4+3Z^5+2Z^6+Z^7}{4}$		$\frac{4Z^3+8Z^4+4Z^5}{4}$	
01,11					
10,00					
10,01	$=A_{01,10}^{C_P}(w,Z)$				
10,10	$=A_{01,01}^{C_P}(w,Z)$				
10,11					
11,00		$\frac{1+2Z+3Z^2+4Z^3+3Z^4+2Z^5+Z^6}{4}$		$\frac{4Z^2+8Z^3+4Z^4}{4}$	
11,01					
11,10					
11,11	Z^8		$\frac{Z^2+4Z^3+10Z^4+12Z^5+9Z^6)}{6}$		Z^4

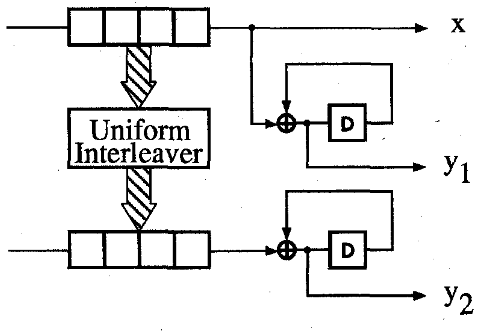

Fig. 9. Parallel concatenated convolutional encoder of Example 6.

state s_i with the state s_j of the CC. The results are summarized in Table III and refer to both identical CC's.

The previous results can be used to construct, through (19), the conditional weight enumerating functions $A_{ij,ml}^{C_P}(w,Z)$ and, through (2), the labeling IRWEF $A_{ij,ml}^{C_P}(W,Z)$ of the hyper-trellis (Table IV).

The hyper-trellis describing the dynamics of this scheme is then a 4-state trellis. The technique that leads to the computation of the performance of this scheme is the same as for a classic time-invariant convolutional encoder.

Thus the average IRWEF $A^{C_P}(W,Z)$ of the PCCC can be found by applying the transfer function technique to this hyper-trellis. ◇

B. An Accurate Approximation

In the preceding example, as the encoder had a very simple structure and the interleaving length was only 4, an analytic approach could be used to determine the exact expression of the average performance of the scheme.

In general, for long interleavers and codes with larger constraint lengths, the hyper-trellis is completely connected, so that the number of branches increases with the fourth power of the number of states (supposed to be equal) of the CC's. Thus although the complexity of the analysis is only related to the CC's and not to the interleaver length, it may become very large. Our experience shows that this is the case for CC's with more than eight states.

To overcome this difficulty, we propose a much simpler analysis. It is based on approximating the complete transfer function of the hyper-trellis with the IRWEF $A_{00,00}^{C_P}(W,Z)$ which labels the branch joining the zero states of the hyper-trellis. It describes all paths which diverge from the zero states of both CC's and remerge into the zero states after N steps.

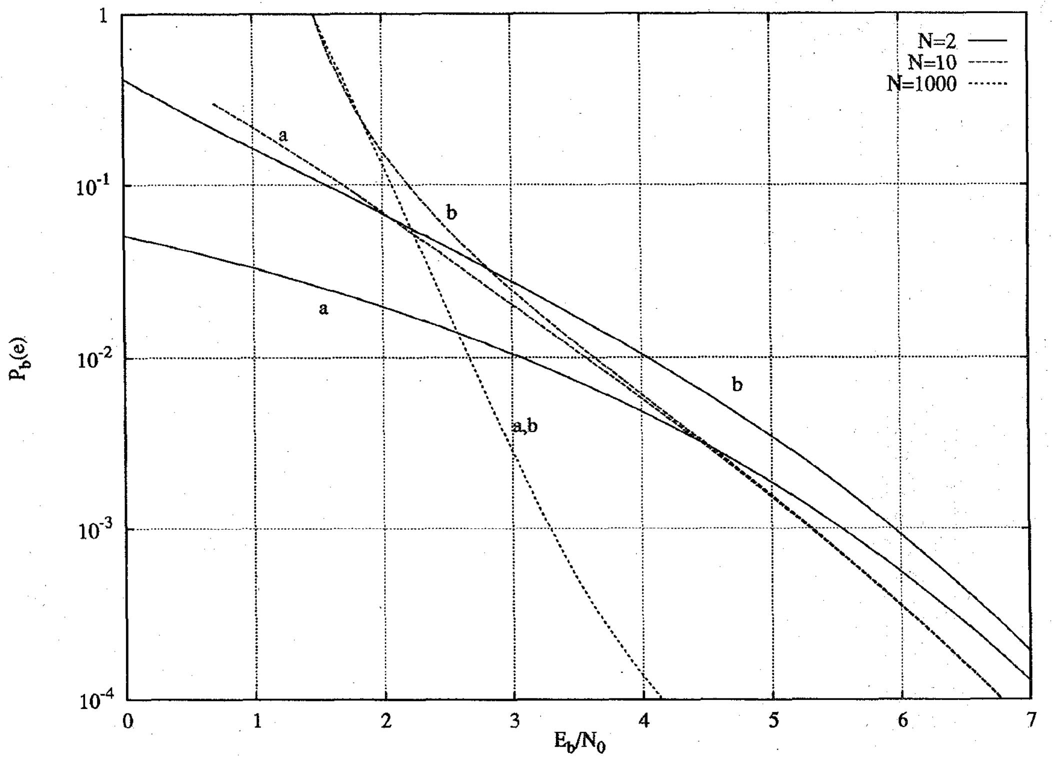

Fig. 10. Comparison between exact and approximate techniques to evaluate the performance of a PCCC. The results refer to the 2-state PCCC of Example 6, with $N = 2, 10, 1000$. Curves labeled "*a*" refer to the approximation, and curves labeled "*b*" to the exact analysis of the hyper-trellis.

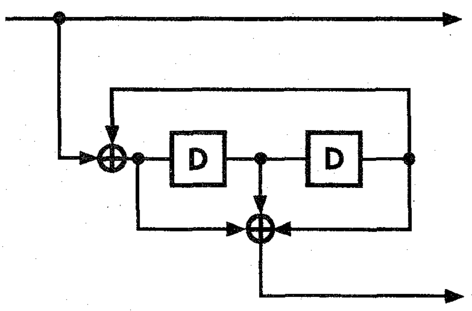

Fig. 11. 4-state recursive convolutional encoder.

To check the accuracy of the approximation technique, we have used the exact and approximate analyses to estimate the performance of the PCCC of Example 6 with different interleaver lengths, namely $N = 2, 10, 1000$. The results are reported in Fig. 10. For $N = 2$, the approximate and exact curves are significantly different above 10^{-4}. They merge around 10^{-2} for $N = 10$, and are completely indistinguishable for $N = 1000$. Actually, this behavior starts from $N = 20$.

In general, we have seen that the approximate method gives accurate results when the interleaver length is significantly larger (say, ten times) than the CC memory. For this reason, since the results that follow refer to this situation, we will use the approximate analysis.

Example 7: Consider a PCCC employing as constituent codes the same 4-state recursive systematic convolutional code with generators $(5, 7)$, free distance 5, and encoder as shown in Fig. 11. We have constructed different PCCC's through interleavers of various lengths, and passed through the previous steps to evaluate their performance.

Upper bounds to the error probability based on the union bound (the transfer function approach for convolutional codes) present a divergence at low values of signal-to-noise ratio. We have checked the influence on the performance bounds of truncating multiple error events of the convolutional code on the basis of their weight. Fig. 12 shows the bit error probability of a PCCC using an interleaver of length 1024, for different truncation weights $(25, 30, 35, 38)$.

Since we did not notice a convergence of the curves, we extended the coefficients D_m defined in (4) by means of an extrapolation based on the exponential law

$$D_m(x) = \exp(\alpha + \beta x).$$

where the parameters α and β have been obtained through a mean-square optimization over the known values of D_m. The curve labeled "extended" in the figure has been obtained in this way. The D_m profile for this code is shown in Fig. 13, together with the curve obtained through extrapolation. The "artificial" decrease in the multiplicities of D_m for large m due to the truncation can be seen from the figure, together with the accuracy of the extrapolation which allows us to include the multiplicities for larger values of m.

The accuracy of the extension procedure has been checked by simulation. For CC's with no more than eight states, it

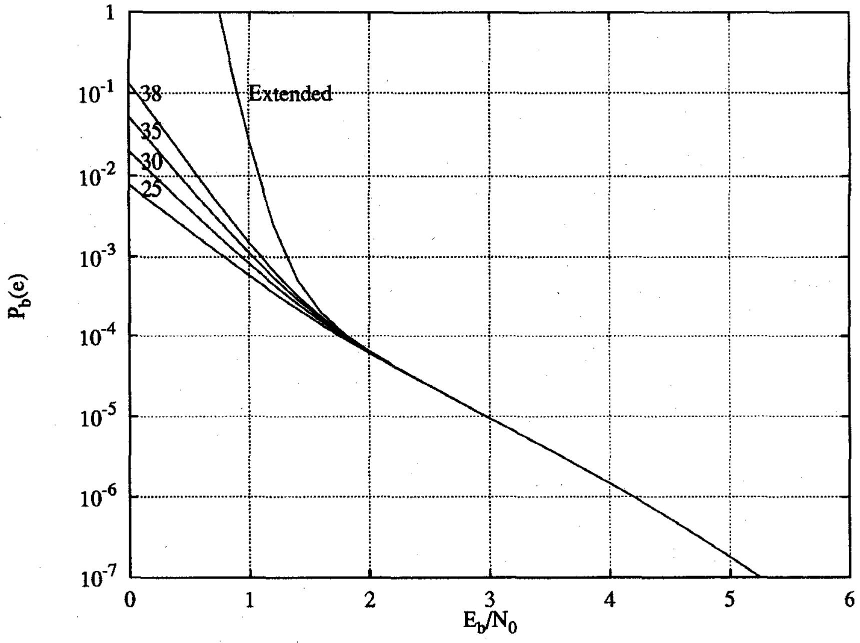

Fig. 12. Influence of the truncation of multiple errors based on their Hamming weight on the performance bound.

TABLE V
COEFFICIENTS D_m FOR THE EVALUATION OF THE BIT ERROR PROBABILITY OF THE PCCC OF EXAMPLE 7 WITH INTERLEAVERS LENGTHS $100, 1000, 10000$

Hamming	N		
distance	100	1000	10000
8	3.8900E-02	3.9881E-03	3.9988E-04
9	7.6590E-02	7.9605E-03	7.9960E-04
10	0.1136	1.1918E-02	1.1991E-03
11	0.1508	1.5861E-02	1.5985E-03
12	0.1986	1.9887E-02	1.9987E-03
13	0.2756	2.4188E-02	2.4017E-03
14	0.4079	2.9048E-02	2.8102E-03
15	0.6292	3.4846E-02	3.2281E-03
16	1.197	6.5768E-02	6.0575E-03
17	2.359	0.1457	1.3697E-02
18	4.383	0.2984	2.8543E-02
19	7.599	0.5472	5.2989E-02
20	12.58	0.9171	8.9441E-02
21	20.46	1.437	0.1403
22	33.31	2.144	0.2082
23	54.65	3.090	0.2957
24	91.23	4.465	0.4177
25	154.9	6.716	0.6133
26	265.5	10.67	0.9577
27	455.6	17.65	1.574
28	779.0	29.61	2.646
29	1327.	49.31	4.430
30	2257.	80.57	7.267
31	3842.	128.6	11.60
32	6556.	201.3	18.04
33	11221	311.5	27.57
34	19261	481.2	41.88
35	33143	748.8	63.94

permits precise estimation of the bound divergence, and will be used in the following to obtain error probability bounds.

We notice, in any case, that all curves merge at a bit error probability around 10^{-4} and stay together from there on.

Let us consider now the performance of different PCCC's when varying interleaver length. In Table V the coefficients D_m needed for the evaluation of the bit error probability of the resulting PCCC for $N = 100, 1000, 10000$ are reported. The effect of longer interleavers is apparent from the table. Namely, the multiplicity of the terms which dominate the performance (those with lower Hamming weights) decreases, when N increases, approximately as $1/N$. As to the free distance, it has increased from 5 (the CC's) to 8.

Finally, in Fig. 14 we present the bit error probabilities of different PCCC's employing the same CC with interleavers of different lengths and, for comparison, the curve of the CC. Gains beyond 4 dB are achievable. The curves in the figure have been extended down to very low values of bit error probability, to show the progressive narrowing of the gap between the curves with different interleaver lengths. This is due to the fact that the free distance of the PCCC's is the same, and thus the curve will eventually merge at high values of signal-to-noise ratio. The curves also show the decrease by a factor 10 of the bit error probability for a factor 10 increase of the interleaver length, as anticipated from the table of coefficients (Table V). $\diamond$

V. COMPARISON BETWEEN ANALYTICAL UPPER BOUNDS ON ML AVERAGE PERFORMANCE AND SIMULATION RESULTS

We deal here with questions 2, 3, and 5 raised in the Introduction.

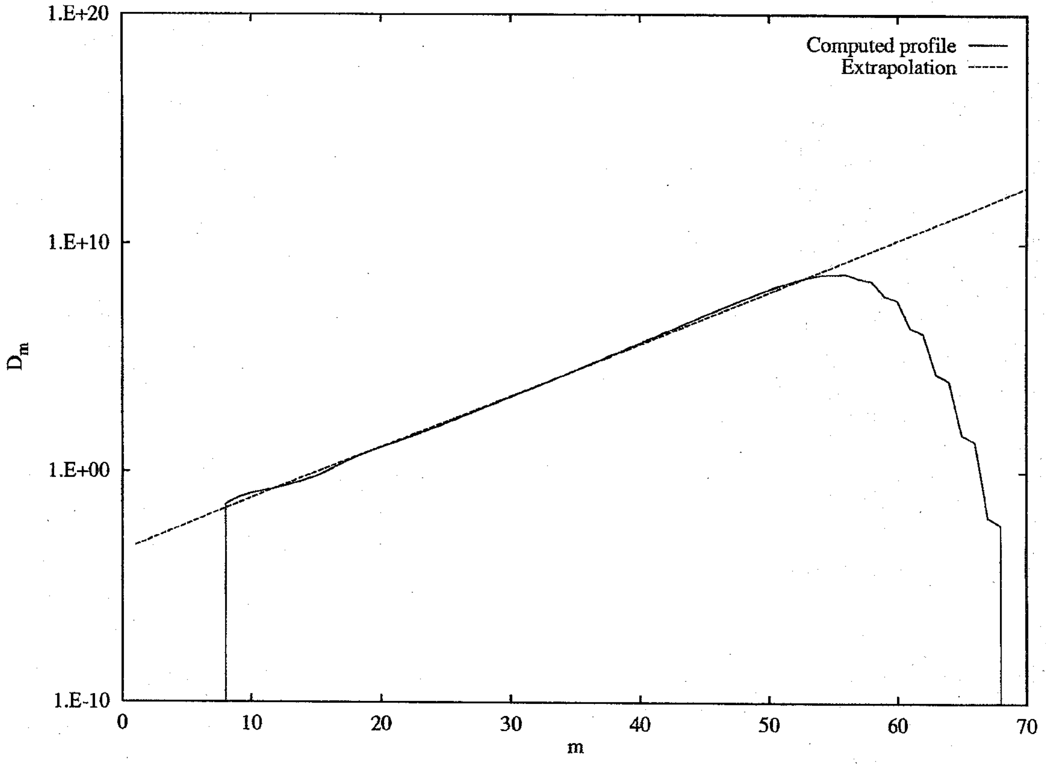

Fig. 13. Profile of the coefficients D_m of the code of Example 7 and its extrapolation ($N = 100$).

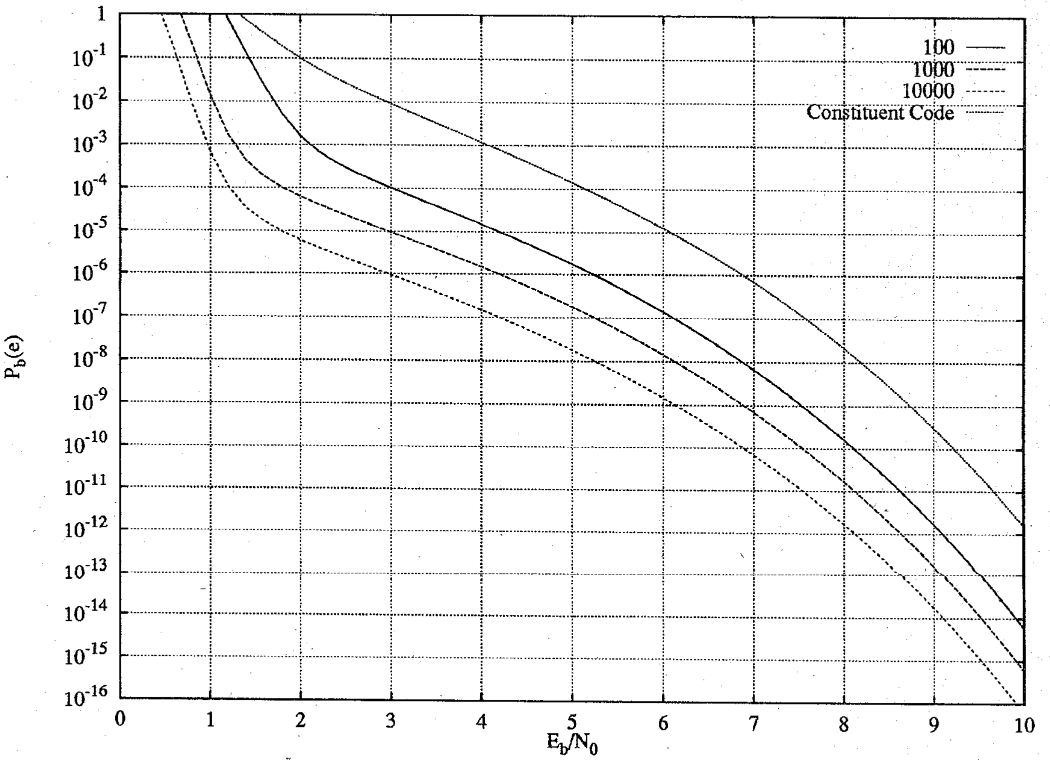

Fig. 14. Average upper bounds to the bit error probability for the PCCC of Example 7 with uniform interleavers of lengths 100, 1000, 10000.

A. The Role of Constituent Codes

We have seen previously that increasing the interleaver length for given CC's lead to noticeable improvements in performance for a wide range of bit error probabilities, even though asymptotically the performance of the PCCC with uniform interleaver are independent from the interleaver length.

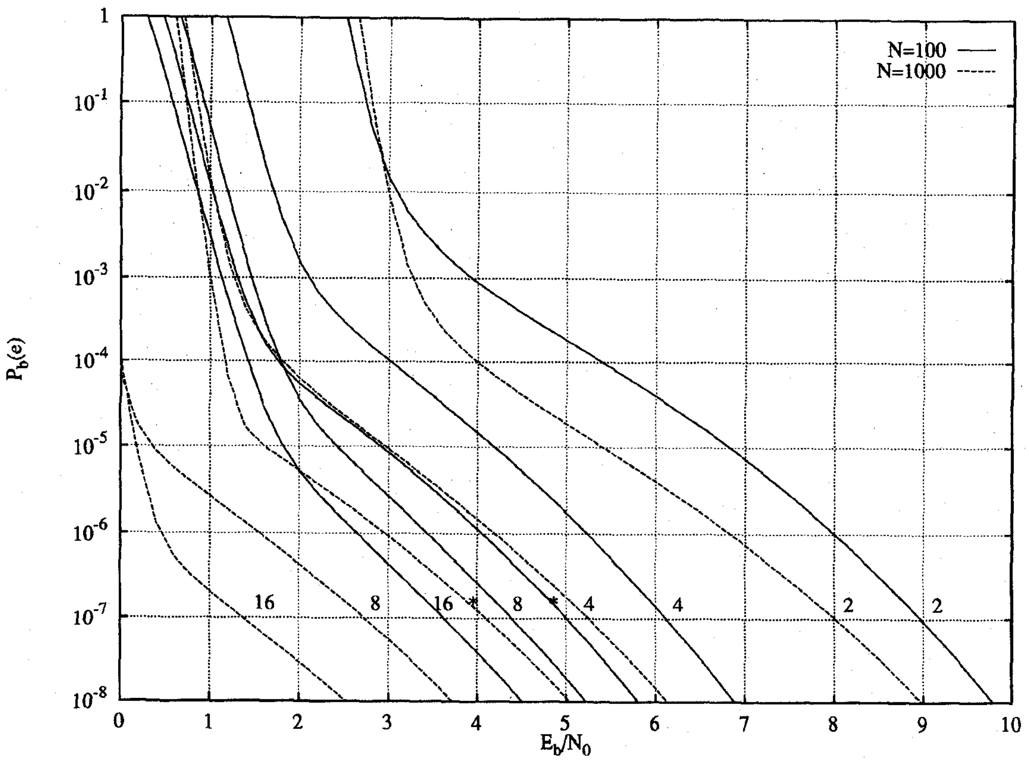

Fig. 15. Average upper bounds to the bit error probability for a PCCC using as CC's two recursive convolutional encoders with 2,4,8, and 16 states and uniform interleavers of length $N = 100$ (continuous curves) and $N = 1000$ (dashed curves).

Let us now consider a uniform interleaver with two lengths $N = 100$ and $N = 1000$ as the building block of PCCC's employing CC's of different constraint length. We will examine the case of rate 1/3 CC's with constraint lengths $2, 3, 4, 5$ and the following generating matrices:

$$2 \rightarrow \left[1, \frac{1}{1+D}\right]$$

$$3 \rightarrow \left[1, \frac{1+D+D^2}{1+D^2}\right]$$

$$4 \rightarrow \left[1, \frac{1+D+D^2+D^3}{1+D+D^3}\right]$$

$$5 \rightarrow \left[1, \frac{1+D+D^2+D^4}{1+D^3+D^4}\right],$$

$$\left[1, \frac{1+D^4}{1+D+D^2+D^3+D^4}\right]^*.$$

For the case of constraint length 5 we have examined two codes: the first uses as denominator of the generating matrix a primitive polynomial, whereas the second (marked with the asterisk) is the one used in the simulation of [5].

Their performance is shown in Fig. 15. Continuous curves refer to the interleaver length 100, while dashed curves are for $N = 1000$. A few comments seem appropriate. At low bit error probabilities, say 10^{-7}, the coding gain yielded by increasing the complexity of the CC's is rather large, namely around 6 dB passing from 2 states to 16 states for both length $N = 100$ and $N = 1000$. This can help in those situations where delay must be kept low, in the sense that interleaver length (and thus delay) can be traded with CC's complexity. As an example, the PCCC based on 8-state codes with $N = 100$ has better performance than the PCCC based on 4-state codes with $N = 1000$, yet reduces the delay. Similar considerations can be developed for higher values of the bit error probability. A comparison between the two 16-state codes show that the code used in [5] (curves marked with an asterisk) is sensibly worse than the one we propose employing a primitive polynomial.

Apart from the effect of the constraint length of constituent codes, there are also important effects due to the *choice* of the CC's for a given constraint length. This "optimization" of the constituent codes is dealt with in a companion paper [26].

B. Maximum-Likelihood and Suboptimal Iterative Decoding

The upper bound (6) is known to be tight for values of the bit error probabilities lower than $10^{-3} - 10^{-4}$. It represents then a good estimate of the average performance of the ML soft decoding of PCC. On the other hand, the practical importance of turbo codes resides in the availability of a simple suboptimal iterative decoding algorithm. To compare its performance with those of ML decoding, we have simulated iterative decoding using a "log-map" soft-output algorithm[4] applied to the PCCC employing two 4-state CC's described in Example 7. The results are reported in Figs. 16 and 17, for $N = 100$ and

[4]This algorithm, a variation on the theme of soft output algorithms like the MAP algorithm described by Bahl [11] and the SOVA algorithm of [14], is described in [27].

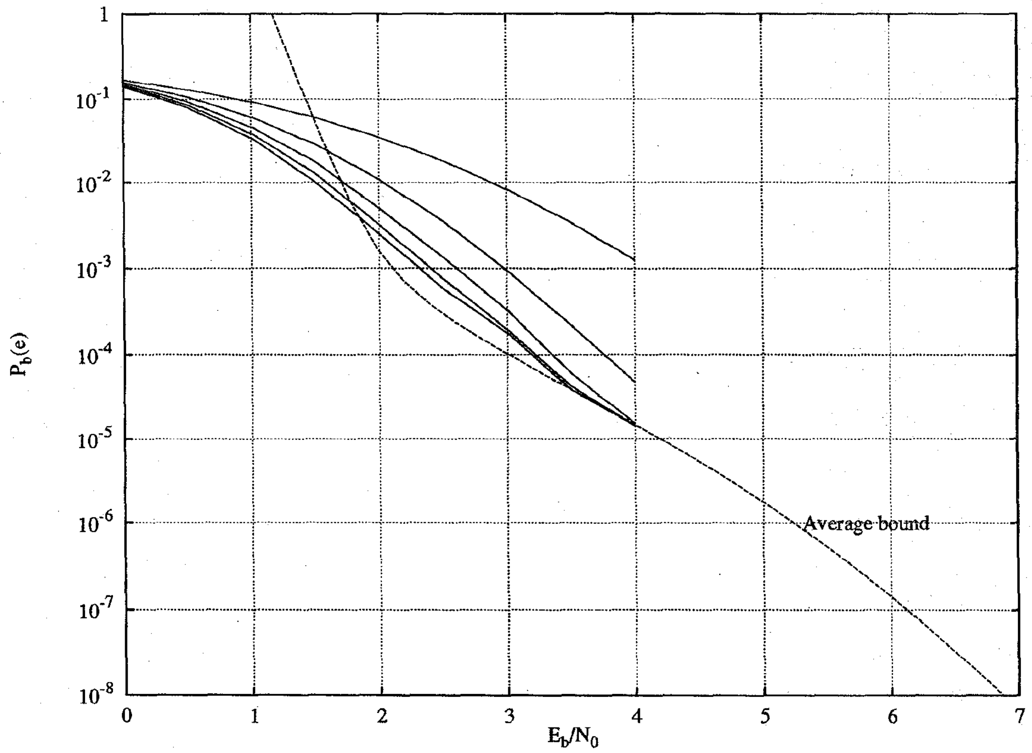

Fig. 16. Comparison between the average upper bound and simulation results for the 4-states rate $1/3$ PCCC of Example 7 with interleaving length $N = 100$. The dashed curve refers to the analytical bound, whereas continuous curves represent simulation results obtained with increasing number of iterations $N_I = 1, 2, 3, 4, 5$.

$N = 1000$, respectively. In the figures, the dashed curves refer to the bound, whereas the continuous curves show the simulation results obtained with an increasing number of iterations, $N_I = 1, 2, 3, 4, 5$ for $N = 100$ and $N_I = 1, \cdots, 10$ for $N = 1000$.

For the simulations, we have used interleavers chosen at random from the set of all permutations. The results were almost independent from the interleaver choice for error probabilities down to 10^{-5}, whereas they started to slightly diverge for lower values, owing to the different values of the free distance yielded by different interleavers. This sheds some light on question 3 of the Introduction, showing that the choice of the interleaver is not critical, as long as one avoids obvious "bad" choices like the replicating interleaver (the one which replicates the input sequence), for medium-high values of the bit error probabilities. A more careful choice yielding higher free distances can pay off for lower values of the error probability.

The results also show that the performance obtained with the uniform interleaver are indeed very close to those obtainable with a practical appropriately chosen interleaver. Also, the convergence of the simulated curves to the ML bound for increasing number of iterations gives a heuristic evidence of the asymptotic optimality of the iterative decoding procedure. This is particularly evident in Fig. 17, where the simulated curves for an increasing number of iterations progressively conform to the ML performance bound.

Finally, it is confirmed that approaching the ML performance requires an increasing number of iterations for low values of the signal-to-noise ratio; four iterations are enough at $P_b(e) = 10^{-5} - 10^{-6}$, whereas we need 8–10 at $P_b(e) = 10^{-3} - 10^{-4}$.

VI. Recursive and Nonrecursive Constituent Encoders

In this section, we deal with question 4 of the Introduction; namely, the role played by recursive systematic convolutional encoders as constituent codes (CC) of the PCCC. Through the analytical upper-bounding technique, we will show that turbo codes do require recursive convolutional encoders to work properly, and that this is a distinctive feature of turbo codes, in the sense that, when considered alone, systematic recursive (SR) and systematic nonrecursive (SNR) convolutional encoders have very similar performance.

Consider a rate $1/3$ PCCC employing as CC's two rate $1/2$ convolutional codes with constraint length 2 whose encoders are shown in Fig. 18. Both are systematic codes[5], the first one (Fig. 18(a)) is recursive while the second (Fig. 18(b)) is not. They have the same transfer function and thus the same error

[5]Using systematic codes is not strictly required; however, it simplifies the decoder and has no effects on attainable performance, so that we will limit ourselves to them.

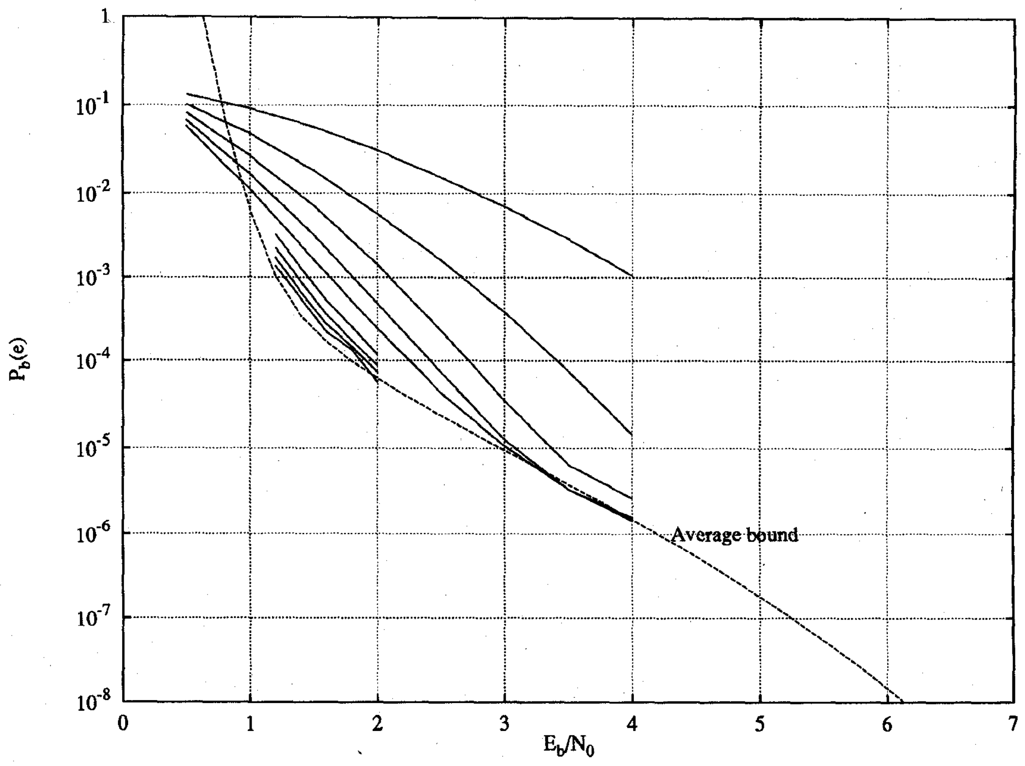

Fig. 17. Comparison between the average upper bound and simulation results for the 4-states rate 1/3 PCCC of Example 7 with interleaving length $N = 1000$. The dashed curve refers to the analytical bound, whereas continuous curves represent simulation results obtained with increasing number of iterations $N_I = 1, \cdots, 10$.

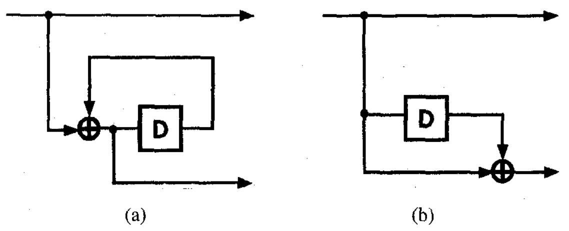

Fig. 18. Two-state recursive (a) and nonrecursive (b) encoders.

event probability.[6] The bit error probability depends on the input–output correspondence of the encoders, and thus is not the same, although the difference is small. Moreover, in this case, as Fig. 19 shows,[7] the systematic nonrecursive encoder behaves better than the recursive one for E_b/N_0 larger than 1 dB.

We now take sequences of length 100 from both CC's, and compute their conditional weight enumerating functions $A_{00}^C(w, Z)$ for weights of the input sequences up to 10. The results are plotted in Figs. 20 and 21, for the recursive and nonrecursive CC, respectively. The differences are apparent. For each weight of the input sequence, the redundancy weights generated by the input sequences of the SR CC span a broad range with a rather uniform multiplicity, while, for the SNR CC the input sequences with small weight generate a small set of redundant weights of low value. These are responsible, after the convolution with themselves which leads to the labels of the branches of the hyper-trellis of the PCCC, for the poor behavior of the concatenated scheme, as will be seen soon. We notice also that, for SNR CC, information sequences with weight $w = 1$ generate error events of finite weight, while, for SR CC, error events start with $w = 2$. This will be proved to be crucial in Section VI-A.

Using the exact bounding technique described in Section IV-B, based on the complete hyper-trellis of the PCCC, we have computed the bit error probabilities for the two types of CC's. The results are reported in Fig. 22 for both schemes employing the SNR code as well as the SR code as CC's. The curves show the bit error probability versus the signal-to-noise ratio for different interleaver lengths.

Curve "A" of Fig. 22 corresponds to the uncoded binary PSK which is reported for reference. Curves "B" and "C" refer to the SNR CC: curve "B" represents the performance of the PCCC obtained by simply duplicating the redundant bit of the CC, while curve "C" derives from the use of an interleaver of length 1000. The results show that the differences with N are marginal (less than half a decibel), and limited to a short range

[6] Actually, this simple case is one of the very few where we can find two equivalent codes, one SR and the other SNR, with the same number of states.

[7] In [20] the authors claim that RS convolutional encoders yield significantly lower error probabilities than SNR codes, even when they have the same transfer function, the key fact being the inapplicability of the Viterbi algorithm for decoding and of the transfer function bound for performance evaluation. To check their conclusions, we have obtained the curve of Fig. 19 using both the transfer function approach and an equivalent block code with very large block size, up to $N = 1000$.

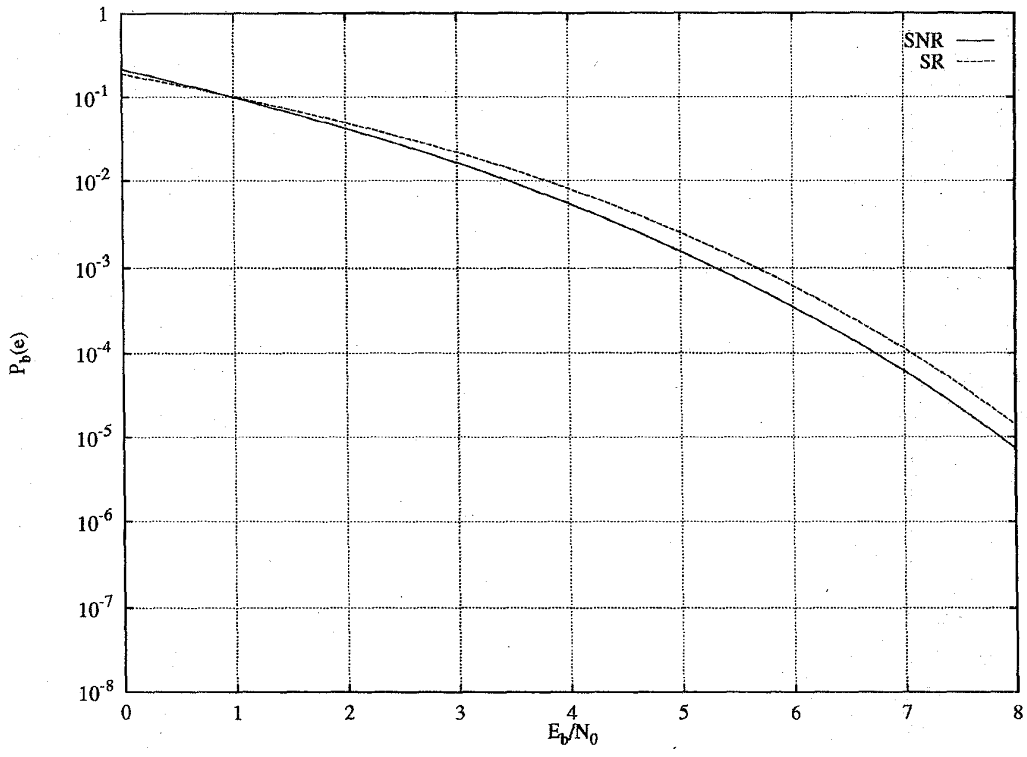

Fig. 19. Bit error probabilities for 2-state SR and SNR encoders.

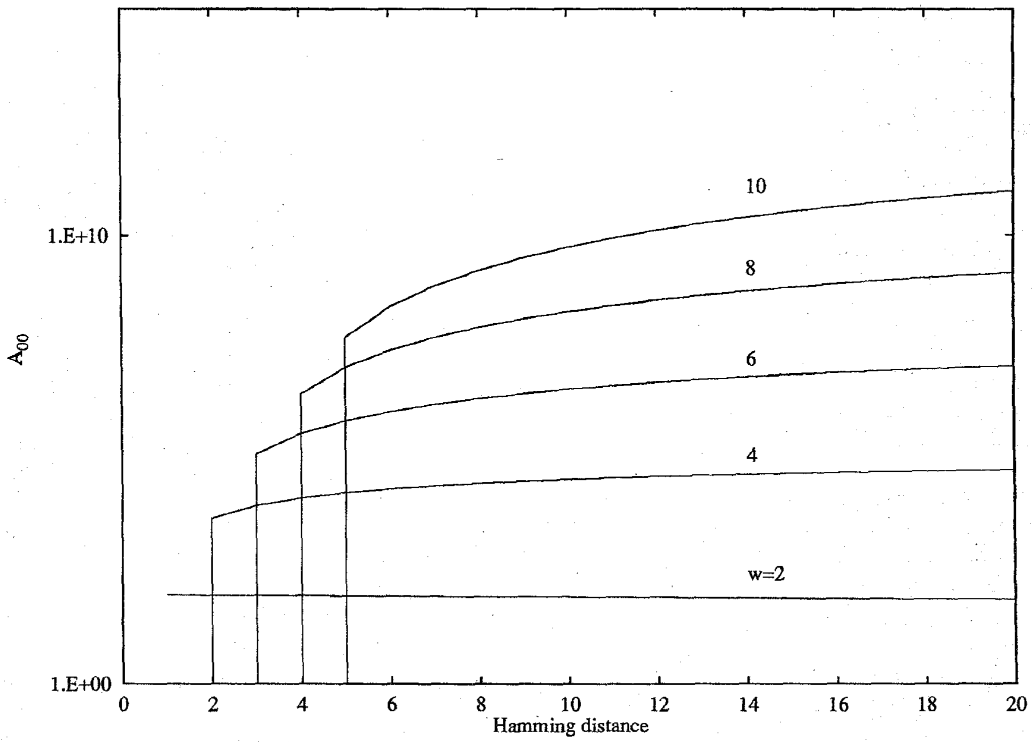

Fig. 20. Coefficients of the conditional weight enumerating functions $A_{00}^C(w, Z)$ for the recursive encoder ($N = 100$).

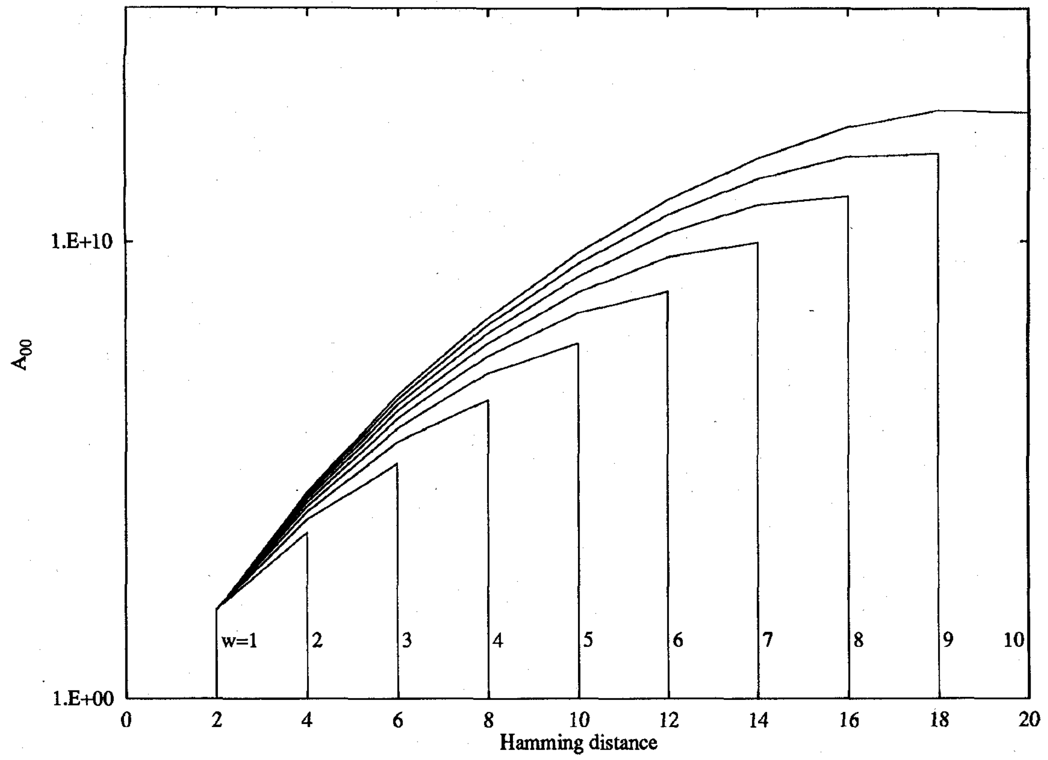

Fig. 21. Coefficients of the conditional weight enumerating functions $A_{00}^{C}(w, Z)$ for the systematic nonrecursive encoder ($N = 100$).

of error probabilities. In fact, the curves practically merge below 10^{-7}.

A completely different behavior for the SR CC is offered by curves "*D*" and "*E*," referring to the same situations as the previous curves "*B*" and "*C*." Here, in fact, we notice a significant improvement for $N = 1000$, yielding a gain of 3 dB at 10^{-5}. Also in this case the curves will merge (being free-distance-independent from N for the uniform interleaver), but this will happen at very low values of the bit error probabilities.

Interestingly enough, for $N = 1$ (compare curves "*B*" and "*D*") the SNR-based PCCC improves over the SR one. This is due to the fact that the same free distance of both rate $1/2$ CC's (recursive and not) is obtained from different contributions of the input and redundant bits, so that duplicating the redundant bits leads to a larger free distance of the SNR-based PCCC. The hierarchy is completely reversed for $N = 1000$.

A. An Approximate Analytical Explanation

We have seen earlier that recursive constituent encoders do play a fundamental role in PCCC's, and that this is due to the *interleaving gain* they yield because of the uniform spread of the weight distributions corresponding to low-weight information sequences. We will give here a simple heuristic explanation of this fact, which illuminates the most important parameters of the CC's in determining the PCCC performance.[8] A more detailed and accurate analytical explanation can be found in [26], where the attention is focused on the optimal design of PCCC with respect to the constituent codes.

Consider a PCCC with an interleaver of length N and two identical convolutional constituent codes C. Let an error event of C have weight $m = w + j$, where w is the number of information bit errors and j the number of redundant bit errors and call $w_{\min}$ the minimum w in any finite-weight error event, and $N_{\min}$ the number of error events with $w_{\min}$ information errors and lowest weight

$$m_{\min} = \min_j [w_{\min} + j]$$

per unit time in C.

The number of possible information error events of weight $w_{\min}$ contained in an interleaver of length N is

$$\binom{N}{w_{\min}}$$

so that the probability under uniform interleaving that the information errors are associated with the lowest weight $m_{\min}$ error event is

$$\frac{N_{\min} \cdot N}{\binom{N}{w_{\min}}} \sim N_{\min} \cdot w_{\min}!\, N^{1-w_{\min}} \tag{20}$$

where the last expression holds for large values of N.

Equation (20) shows that the *interleaver gain* in the error coefficient is proportional to $N^{1-w_{\min}}$, and, consequently, that this parameter $w_{\min}$ is indeed a key design parameter of the CC's. Now, it is easy to see that $w_{\min}$ is equal to 2 for all recursive convolutional encoders, which yields the interleaver

[8] We ought to mention that Dave Forney came to the same conclusions shown hereafter after reading a first version of this manuscript, where they did not show up clearly. The line of thought in this subsection follows closely his comments.

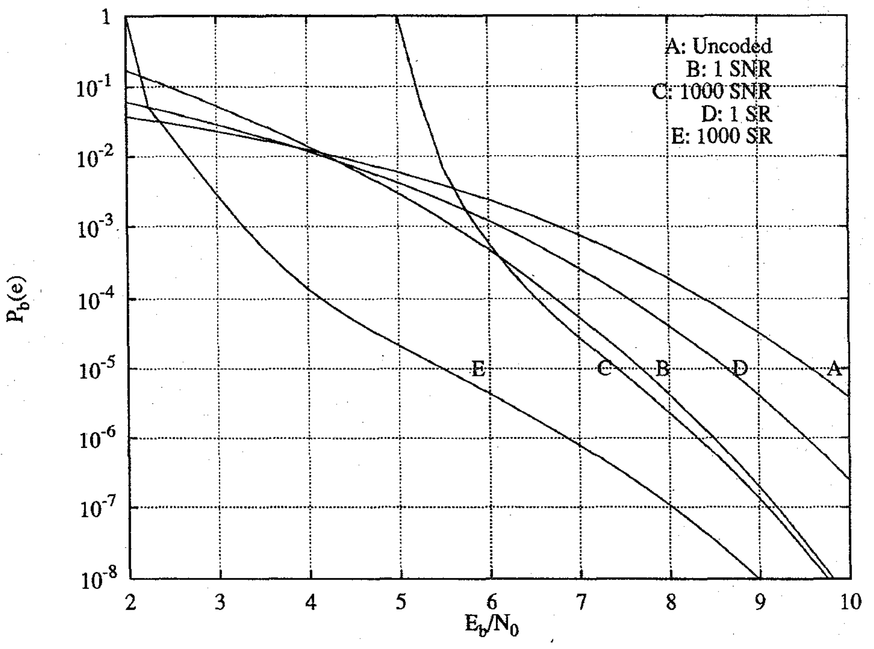

Fig. 22. Bit error probabilities comparison between PCCC using SR and SNR CC's.

gain increase $1/N$ noted in Example 7, and that it is equal to 1 for nonrecursive encoders, which explains the results of Fig. 22 being almost independent from N. It is worthwhile mentioning that $w_{\min}$ is also equal to 1 for block codes, which explains the lower gain obtained with PCBC's of Section II.

To quote Dave Forney[9] "Turbo codes seem to turn the conventional design principles on their head; they make error coefficients more important than minimum distance !" Indeed, the main CC's design parameters to optimize a PCCC for a given interleaver length are $w_{\min}$ and $m_{\min}$, and the last can also be significantly larger than the free distance of the code.

VII. Prospects of Research

There are several important open questions and topics for future research, including those in the following list.

1) We have shown through examples that the iterative soft-output decoding scheme employed by turbo codes approaches the ML performance bound for increasing number of iterations. An important theoretical question concerns the convergence of the suboptimal algorithm to the ML decoding. Does it converge ? Under what conditions ?
2) We have shown in some cases that a reasonable choice of the interleaver leads to performance close to the average, and that the interleaver choice is not critical, as opposed to its length, which is indeed the main reason for the good performance of PCC's. However, the interleaver plays a role in determining the free (or minimum) distance of the PCC, and, consequently, the asymptotic performance of the code. Constructive algorithms to find "good" interleavers of a given length, as well as theoretical results on limits to the achievable free distance for a given interleaver length and CC's would be important, especially for PCCC's employing short interleavers. A few, very preliminary steps in this directions have been taken in [17], [19], [25], [28] using "cut-and-try" approaches.
3) For some applications, a short decoding delay is a must. In these cases, the best compromise between interleaver length and CC's complexity certainly deserves attention, as does the optimization of the CC's based on the new principles outlined in Section VI-A.
4) Several iterative decoding algorithms have already been proposed. A comparison of them, as well as the search for new ones aiming at reducing the number of iterations (and consequently delay) required would be an important achievement.
5) We have proposed here a bounding technique valid for both block and convolutional PCC's. While simple and good iterative soft-decoding algorithms for PCCC have been proposed and implemented, the same is not true for PCBC's. A first solution for the particular case of product block codes has been proposed recently in [10]. However, a complexity analysis is still missing, and other alternatives need to be explored.
6) In certain applications, bandwidth efficiency is required, and the best compromise between coding gain and bandwidth efficiency found so far is trellis-coded modulation. A pragmatic approach to joint turbo coding and modulation was presented in [8], with promising results. A satisfactory approach to overall optimization as available for TCM is nevertheless still to come.
7) It is a common belief that the performance of PCC degrade significantly if one tries to increase the rate. This

[9] Private communication.

conclusion is based on simulation of PCC employing punctured CC's. Other solutions should be tried, such as using higher rate CC's. The analytical tools presented here give a way to analyze them.

8) We already mentioned that the interleaver inherently present in the coding scheme might prove beneficial for channels affected by fading. Some results obtained by simulation in this direction are contained in [8]. Extension of the analytical bounding technique presented here for AWGN channels to fading channels would be important for both analysis and design purposes.

VIII. Conclusions

We have proposed for the first time a method to evaluate the bit error probability of a parallel concatenated coding scheme independently from the interleaver used. Crucial was the introduction of a probabilistic interleaver called *uniform interleaver* which permits an easy derivation of the weight enumerating function of the parallel concatenated code starting from the weight enumerating functions of the constituent codes. The two cases of parallel concatenated block codes and parallel concatenated convolutional codes were considered. This analytical bounding technique was then used extensively to clarify some relevant aspects of this interesting and promising coding technique.

Appendix
Evaluation of the Conditional Weight Enumerating Function $A_w^C(Z)$ of the Equivalent Block Code

In this Appendix we show how to compute the conditional weight enumerating function $A_{sn}^C(w, Z)$ which describes the equivalent block code obtained as the set of sequences of the constituent code (CC) C leading to the CC trellis from state s_s to state s_n in N steps. For simplicity, we will only show the derivation of $A_{00}^C(w, Z)$, which will be called $A_w^C(Z)$ for brevity.

We consider then for each constituent convolutional code an equivalent block code whose trellis representation is the truncation at step N of the trellis of the convolutional code, and whose codewords lead the trellis into the identity state at step N. Our goal is to derive the IRWEF of such a block code, starting from the knowledge of a suitably defined *error events enumerating function* of the convolutional code.

With such a definition, the number of codewords of the equivalent block code can be very large, but not infinite as for the convolutional code, so that a short closed-form expression for its IRWEF does not exist. For this reason, we will use an algorithmic approach that allows the evaluation of the most significant terms of the IRWEF.

Let us consider Fig. 23. By our previous hypotheses on the block code which approximates the constituent convolutional code, any codeword belonging to the block code is obtainable by combining the set of error events of the convolutional code with suitable sequences of "0" so that the total length equals N.

As an example, a single error event of length l smaller than N produces all codewords with $N - l$ zeros positioned before

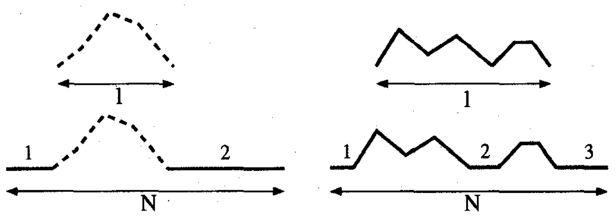

Fig. 23. How to obtain codewords of the equivalent block code from error events of the convolutional code.

and after the error event. All these codewords share the same input and redundancy weight, so that they can be grouped together. The multiplicity $K[l, 1]$ of the block codewords produced from this single error event in the IRWEF of the block code equals the number of partitions of $N - l$ into two numbers:

$$K[l, 1] = \binom{N - l + 1}{1} = N - l + 1.$$

Similarly, a single pair of error events with total length l produces all the codewords with zeros before, after, and between the two error events. Thus the multiplicity $K[l, 2]$ of the codewords produced from this single pair in the IRWEF of the block code equals the number of partitions of $N - l$ into three numbers

$$K[l, 2] = \binom{N - l + 2}{2}.$$

Proceeding this way, one can obtain the general expression for the multiplicity of codewords produced by a single combination of n error events with total length l

$$K[l, n] = \binom{N - l + n}{n}.$$

Let $T^C(W, Z, L, \Omega)$ be the transfer function of the convolutional code which enumerates all paths in the trellis leaving the zero state at step 1, and remerging into the zero state at or before step N, with possible remerging into the zero state at other steps in between, subject to the constraint that, after remerging, they leave the zero state immediately at the successive step[10]

$$T^C(W, Z, L, \Omega) = \sum_{i,j,m,n} T_{i,j,m,n} W^i Z^j L^m \Omega^n \qquad (21)$$

where $T_{i,j,m,n}$ is the number of paths in the trellis produced by an input sequence of weight i, with weight of the redundant bits equal to j, length m, and n remergings with the zero state (and hence concatenating n error events).

As for the case of block codes, we define the conditional transfer function $T_w^C(Z, L, \Omega)$ as

$$T_w^C(Z, L, \Omega) = \sum_{j,m,n} T_{w,j,m,n} Z^j L^m \Omega^n. \qquad (22)$$

Passing now to the codewords of the equivalent block code, we notice that each path of length m and number of remergings n

[10]Examples of these concatenations of single error events were shown in Fig. 23.

belonging to $T_w^C(Z, L, \Omega)$ gives rise to $K[m, n]$ codewords with the same input and redundancy weights, so that the conditional IRWEF $A_w^C(Z)$ of the equivalent block code can be obtained as

$$A_w^C(Z) = \sum_j A_{wj} Z^j \tag{23}$$

with

$$A_{wj} \triangleq \sum_{m,n} K[m, n] T_{w,j,m,n}.$$

An efficient algorithm able to compute the most significant terms[11] of the transfer function $T^C(W, Z, L, \Omega)$ has been implemented, elaborating on the algorithm described in [29] to evaluate the transfer function of a convolutional code, and then yielding as output the conditional IRWEF of the equivalent block code.

[11] We consider only the code sequences with Hamming weights smaller than a given threshold.

REFERENCES

[1] P. Elias, "Error-free coding," *IEEE Trans. Inform. Theory*, vol. PGIT-4, pp. 29–37, Sept. 1954.

[2] S. Hirasawa, M. Kasahara, Y. Sugiyama, and T. Namekawa, "Modified product codes," *IEEE Trans. Inform. Theory*, vol. IT-30, no. 2, pp. 299–306, Mar. 1984.

[3] G. D. Forney Jr, *Concatenated Codes.* Cambridge, MA: M.I.T. Press, 1966.

[4] S. Hirasawa, M. Kasahara, Y. Sugiyama, and T. Namekawa. "Certain generalizations of concatenated codes—Exponential error bounds and decoding complexity," *IEEE Trans. Inform. Theory*, vol. IT-26, pp. 527–534, Sept. 1980.

[5] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: turbo-codes," in *Proc. ICC'93*, (Geneve, Switzerland, May 1993), pp. 1064–1070.

[6] D. Divsalar and F. Pollara, "Turbo codes for PCS applications," in *Proc. ICC'95* (Seattle, WA, June 1995).

[7] Comatlas, Chateaubourg, France. *CAS 5093 Turbo-Code Codec*, 3.7 ed., Aug. 1994. Data sheet.

[8] S. Le Goff, A. Glavieux, and C. Berrou, "Turbo-codes and high spectral efficiency modulation," in *Proc. ICC'94* (New Orleans, LA, May 1994).

[9] S. Benedetto and G. Montorsi, "Average performance of parallel concatenated block codes," *Electron. Lett.*, vol. 31, no. 3, pp. 156–158, Feb. 1995.

[10] R. Pyndiah, A. Glavieux, A. Picart, and S. Jacq, "Near optimum decoding of product codes," in *Proc. GLOBECOM '94*, (San Francisco, CA, Nov. 1994), vol. 1, pp. 339–343.

[11] 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.

[12] C. Berrou, P. Adde, A. Ettiboua, and S. Faudeil. "A low complexity soft-output Viterbi decoder architecture," in *Proc. ICC'93* (Geneva, Switzerland, May 1993).

[13] G. Battail, "Pondération des symboles décodeés par l'algorithme de Viterbi," *Ann.s Télécomm.*, vol. 42, nos. 1–2, pp. 31–38, 1987.

[14] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *Proc. GLOBECOM'89* (Dallas, TX, Nov. 1989), pp. 47.1.1–47.1.7.

[15] J. H. Lodge, R. J. Young, P. Hoeher, and J. Hagenauer, "Separable MAP 'filters' for the decoding of product codes and concatenated codes," in *Proc. ICC'93* (Geneva, Switzerland, May 1993), pp. 1740–1745.

[16] J. D. Andersen, "The TURBO coding scheme," in *Proc. ISIT'94* (Trondheim, Norway, June 1994).

[17] P. Jung and M. Nasshan, "Performance evaluation of turbo codes for short frame transmission systems," *Electron. Lett.*, vol. 30, no. 2, pp. 111–113, Jan. 1994.

[18] P. Robertson, "Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes," in *Proc. GLOBECOM '94* (San Francisco, CA, Nov. 1994), vol. 3, pp. 1298–1303.

[19] P. Jung and M. Nasshan, "Dependence of the error performance of turbo-codes on the interleaver structure in short frame transmission systems," *Electron. Lett.*, vol. 30, no. 4, pp. 287–288, Feb. 1994.

[20] G. Battail, C. Berrou, and A. Glavieux, "Pseudo-random recursive convolutional coding for near-capacity performance ," in *Proc. GLOBECOM'93, Communication Theory Mini-Conf.* (Houston, TX, Dec. 1993).

[21] S. Benedetto, E. Biglieri, and V. Castellani, *Digital Transmission Theory.* Englewood Cliffs, NJ: Prentice-Hall, 1987.

[22] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding.* New-York: McGraw-Hill, 1979.

[23] S. Benedetto and G. Montorsi, "Analysis and results on parallel concatenated coding schemes with multiple interleavers," in *Proc. 3rd Int. Symp. on Communication Theory and Applications* (Lake District, UK, July 1995).

[24] G. Battail, "Construction explicite de bons codes longs," *Ann. Télécommun.*, vol. 44, nos. 7–8, pp. 392–404, 1989.

[25] C. Berrou and A. Glavieux, "Turbo codes: General principles and applications," in *Proc. 6th Tirrenia Int. Workshop on Digital Communications* (Tirrenia, Italy, Sept. 1993).

[26] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes," accepted for publication in *IEEE Trans. Commun.*

[27] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Soft-ouput decoding algorithms in iterative decoding of parallel concatenated convolutional codes," accepted for ICC' 96.

[28] P. Robertson, "Understanding turbo codes—Are they capable of near Shannon limit error correction ?" in *Proc. 6th JCCC Joint Conf. on Communications and Coding* (Selva di Val Gardena—Wolkenstein, Italy, Mar. 1994).

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