(iv) Find the quantisation levels by adding displacements in opposite directions:
\[ q_{k+1} = m + \mu (m_{k+1} - m_k) \]
where \( \mu \) is an empirically chosen constant (0 < \( \mu < 2 \)).

The modification in the MBTS algorithm is in the last step where the quantisation levels are calculated. In the BTS algorithm the quantisation levels are selected at cluster centres. In MBTS, they are displaced from the cluster centres to facilitate the representation of more colours within a cluster with closer quantisation values.

**Results and conclusion:** An illustration of how the MBTS eliminates colour impulses and produces sharper details is given in Fig. 1. The original image in Fig. 1a is quantised to 16 colours by the BTS algorithm and dithered using the Floyd-Steinberg (F-S) algorithm (Fig. 1b). The resulting image loses a significant amount of detail, as evidenced by the disappearance of the lettering on the globe. Furthermore, the colour of certain areas is unfaithful. When the same procedure is repeated with the MBTS (Fig. 1c) a more faithful result is obtained. Although the detail and the lettering on the globe is preserved, the overall reproduction of colours is also closer to the original.

In this work, the BTS algorithm has been modified to take the subsequent dithering into account. The colour palette has been changed to accommodate more colours when error diffusion dithering is applied. Although the total squared error of the MBTS is higher than that of the BTS, the palette generated by the MBTS gives distinctly superior results when dithering is applied. This illustrates the potential for improvement in quantiser design.

Our present work focuses on the joint design of quantisers and ditherers.

**References**


**Iterative decoding of serially concatenated convolutional codes**

S. Benedetto and G. Montorsi

**Indexing terms:** Convolutional codes, Decoding

Serial concatenation of convolutional codes separated by an interleaver has recently been shown, through the use of upper bounds to the maximum likelihood performance, to be competitive with parallel concatenated coding schemes known in the literature as 'turbo codes'. The most important feature of turbo codes consists in their relatively simple, yet high performance, iterative decoding algorithm. The authors propose a new iterative decoding algorithm for serial concatenation, and show that the new coding scheme can yield a significant advantage with respect to turbo codes.

**Introduction:** Since their appearance in [1], 'turbo codes' have been the object of great interest, and consequently of wide investigation, in the coding community. The practical importance of turbo codes stems from the fact that they achieve considerable coding gains yet admit iterative soft decoding algorithms whose complexity is not significantly higher than that of the decoder for single constituent codes.

The idea of constructing high performance coding schemes through the concatenation of simpler codes dates back from [2]. Although classical and widely employed concatenated codes make use of a Reed Solomon code as an outer code and of a convolutional code as an inner code, serial concatenation of convolutional codes has also been proposed and analysed [3]. In classical concatenated coding theory, however, the possible insertion of an interleaver between outer and inner codes was considered only as a way of randomising the errors produced by the inner encoder, and not, as in turbo codes, to construct a new, more powerful, integrated coding scheme.

**References**


**Iterative decoding of serially concatenated convolutional codes**

S. Benedetto and G. Montorsi

**Indexing terms:** Convolutional codes, Decoding

Serial concatenation of convolutional codes separated by an interleaver has recently been shown, through the use of upper bounds to the maximum likelihood performance, to be competitive with parallel concatenated coding schemes known in the literature as 'turbo codes'. The most important feature of turbo codes consists in their relatively simple, yet high performance, iterative decoding algorithm. The authors propose a new iterative decoding algorithm for serial concatenation, and show that the new coding scheme can yield a significant advantage with respect to turbo codes.

**Introduction:** Since their appearance in [1], 'turbo codes' have been the object of great interest, and consequently of wide investigation, in the coding community. The practical importance of turbo codes stems from the fact that they achieve considerable coding gains yet admit iterative soft decoding algorithms whose complexity is not significantly higher than that of the decoder for single constituent codes.

The idea of constructing high performance coding schemes through the concatenation of simpler codes dates back from [2]. Although classical and widely employed concatenated codes make use of a Reed Solomon code as an outer code and of a convolutional code as an inner code, serial concatenation of convolutional codes has also been proposed and analysed [3]. In classical concatenated coding theory, however, the possible insertion of an interleaver between outer and inner codes was considered only as a way of randomising the errors produced by the inner encoder, and not, as in turbo codes, to construct a new, more powerful, integrated coding scheme.

**References**


**Iterative decoding of serially concatenated convolutional codes**

S. Benedetto and G. Montorsi

**Indexing terms:** Convolutional codes, Decoding

Serial concatenation of convolutional codes separated by an interleaver has recently been shown, through the use of upper bounds to the maximum likelihood performance, to be competitive with parallel concatenated coding schemes known in the literature as 'turbo codes'. The most important feature of turbo codes consists in their relatively simple, yet high performance, iterative decoding algorithm. The authors propose a new iterative decoding algorithm for serial concatenation, and show that the new coding scheme can yield a significant advantage with respect to turbo codes.

**Introduction:** Since their appearance in [1], 'turbo codes' have been the object of great interest, and consequently of wide investigation, in the coding community. The practical importance of turbo codes stems from the fact that they achieve considerable coding gains yet admit iterative soft decoding algorithms whose complexity is not significantly higher than that of the decoder for single constituent codes.

The idea of constructing high performance coding schemes through the concatenation of simpler codes dates back from [2]. Although classical and widely employed concatenated codes make use of a Reed Solomon code as an outer code and of a convolutional code as an inner code, serial concatenation of convolutional codes has also been proposed and analysed [3]. In classical concatenated coding theory, however, the possible insertion of an interleaver between outer and inner codes was considered only as a way of randomising the errors produced by the inner encoder, and not, as in turbo codes, to construct a new, more powerful, integrated coding scheme.

**References**

PCCCs employing recursive constituent encoders have $w_{ev} = 2$ they present an interleaving gain behaving as $1/N$.

We assume, for analytics, that for an SCCC the limitation to the performance comes principally from error events of the second (inner) code produced by its input sequences of lowest weight. The main difference is that for an SCCC the input sequences to the inner encoder are not unconstrained input information sequence, as they are for a PCCC, but coded sequences produced by the first (outer) code. As a consequence, the lowest weight of these sequences corresponds to the free distance of the outer code, which can be significantly higher than two, thus yielding a higher interleaving gain.

The resulting advantage was demonstrated through the analytical bounds of [4]. To acquire a practical significance, however, this important theoretical result needs the support of a decoding algorithm of the same order of complexity as in turbo decoding, yet which retains the performance advantages. The new algorithm is presented in this letter.

**Iterative decoding algorithm for SCCC:** The suboptimum decoding algorithm is iterative, and presents a complexity not significantly higher than that needed to separately decode the two CCs. As for turbo codes, the core of the new decoding process consists of a maximum-a-posteriori (MAP) decoding algorithm applied to the CCs. As already carried out for PCCC, we will use the sliding-window MAP algorithm described in [7]. A functional diagram of the new iterative decoding algorithm for SCCCs is presented in Fig. 2, where we also show the algorithm that decodes turbo codes, to highlight analogies and differences.

Fig. 2 Iterative decoding algorithm for serially concatenated convolutional codes

- a PCCC, b SCCC

We will explain in detail how the algorithm works, according to the blocks of Fig. 2. The blocks labeled ‘MAP’ are drawn with two inputs and two outputs. The input labeled $I$ represents the logarithm of the probability density function (LPDF) of the unconstrained output symbols of the encoder, and that labeled $O$ the LPDF of unconstrained input symbols. Similarly, the outputs represent the same quantities conditioned to the code constraint as they are evaluated by the MAP decoding algorithm. Unlike the iterative decoding algorithm used for turbo decoding, in which the MAP algorithm only computes the LPDF of input symbols conditioned on the code constraint based on the unconstrained LPDF of input symbols, here we fully exploit the potential of the MAP algorithm. It can, in fact, update both the LPDF of the input and output symbols based on the code constraints.

We assume that the pair $(t, o)$ of symbols, labelling each branch of the code trellis, is independent at the input of the MAP decoder, so that their joint LPDF is given by:

$$LPDF(t, o) = LPDF(t) \cdot LPDF(o)$$

During the first iteration of the SCCC algorithm, the block ‘MAP inner code’ is fed with the demodulator soft output, consisting of the LPDF of symbols received from the channels, i.e., of output symbols of the inner encoder. The LPDF is processed by the first MAP decoder that computes the LPDF relative to the input symbols conditioned on the inner code constraints. This information, from which we subtract the unconstrained input LPDF to obtain the ‘extrinsic’ information as in turbo decoding, is passed through the inverse interleaver (block labelled ‘$\pi^{-1}$’). To simplify the description we assume that the interleaver acts on symbols instead of bits. In the actual decoder, we deal with bit, LPDF, and interleaver. As the input symbols of the inner code, after inverse interleaving, correspond to the output symbols of the outer code, they are sent to the MAP outer code block in the upper entry, which corresponds to output symbols. The outer MAP decoder, in turn, processes the LPDF of the unconstrained output symbols and computes the LPDF of both output and input symbols based on the code constraints. The LPDF of input symbols (the MAP information) will be used in the final iteration to recover the information bits, whereas the LPDF of output symbols, after subtraction and interleaving, is fed back to the MAP inner decoder to start the second iteration.

**Fig. 3 Simulated performances of rate 1/3 SCCC and PCCC**

- PCCC, - - - - SCCC

**Performance of the decoding algorithm:** To show the performance of the SCCC decoded using the new algorithm, we have simulated a rate 1/3 SCCC using two four-state recursive convolutional codes, the first (outer code) with rate 1/2 and the second (inner code) with rate 2/3, joined by an interleaver of length $N = 2048$. Since the interleaver operates on coded sequences produced by the outer rate 1/2 encoder, its length of 2048 bits corresponds to a delay of 1024 information bits. The simulation results are shown as dashed curves in Fig. 3 in terms of bit error probability against $E_b/N_0$ for a number of iterations ranging from one to seven. The good convergence of the decoding algorithm is manifest.

In this figure, we also report the simulation results pertaining to a rate 1/3 PCCC formed by two equal four-state rate 1/2 recursive convolutional constituent codes, joined by an interleaver of length $N = 1024$, which induces the same delay on the information bits, using the iterative turbo-decoding algorithm and the same numbers of iterations.

The comparison between the two set of curves is striking, as they show the great advantage of SCCCs over PCCCs in terms of interleaving gain. In fact, the change of slope that characterises the error probability curves of turbo codes does not take place with SCCCs, or, at least, it appears more than one order of magnitude later. The gain in $E_b/N_0$ is $>0.65$dB at $P_e = 10^{-4}$. It should also be noted that the two CCs are optimised for the turbo code according to [6], whereas no optimisation is performed for SCCCs.

References


Authorized licensed use limited to: Politecnico di Torino. Downloaded on March 12, 2010 at 12:47:08 EST from IEEE Xplore. Restrictions apply.
Parallel decoding of turbo codes

Y. Chang

Introduction: Turbo codes are the most exciting and potentially important development in coding theory in many years. They were first introduced in 1993 by Berron et al. [1], and have received the attention of many researchers. In this Letter a new type of turbo code, based on GF(2') with a corresponding decoding algorithm, is formulated in a more simple fashion by passing information from one decoder to the next using probability functions, as opposed to channel values that need to be normalised. No heuristically determined correction parameters are necessary for stable decoding. In this way, binary turbo codes can be decoded in a parallel operational manner.

Q-ary turbo code: Let \( \{d_i\} = (d_1, ..., d_i, ..., d_N) \), \( d_i \in GF(2') \); \( k = 1, 2, ..., N \) denote a q-ary (q = 2) information sequence of length \( N \). Transform \( \{d_i\} \) into a \( N \) length binary sequence and feed it into an encoder for a binary recursive systematic convolutional (RSC) code of coding rate \( R = 1/2 \). Then we obtain a \( N \) length binary parity check sequence, which is also an \( N \) length q-ary parity check sequence. We call this type of code a q-ary RSC code. A q-ary turbo code can be obtained through a parallel concatenation of two or more q-ary RSC codes. The interleaving and puncturing should operate in a symbol-by-symbol manner, not in a bit-by-bit manner. Thus the interleaver size of the q-ary turbo code is \( i \) times less than that of the corresponding binary code. The trellis structure of the q-ary turbo code is obtained from the corresponding binary code where \( i \) successive nodes are joined into one node. Therefore the decoding algorithm of a q-ary turbo code is also a parallel decoding algorithm of the binary code.

Decoding algorithm: Fig. 1 shows the structure of a turbo decoder with two parallel decoders. Let \( \{X_i\}, \{Y_i\} \), and \( \{Y_0\} \) denote the received sequences of a q-ary information sequence, the corresponding q-ary parity check sequence produced by the first component encoder, and the corresponding q-ary parity check sequence produced by the second component encoder, respectively. Let \( \{A(d_i)\} \) and \( \{Z_i\} \) denote the output sequence of DEC1 and the feedback sequence provided by DEC2, respectively. Then the input sequence to DEC1 can be expressed as \( R_i^e = (R_{i-1} ... R_i) \), where \( R_i \) is determined by the 4th components of \( \{X_i\}, \{Y_i\} \), and \( \{Z_i\} \). And the input sequence to DEC2 can be expressed as \( R_i^f = (R_{i-1} ... R_i) \), where \( R_i \) is a permutation of \( A(d_i) \) determined by the interleaving operation. Assume DEC1 has known \( p(z_i|d_i \not= 0) \) for all \( i \in GF(q) \) and all \( k \). Then using the Bath et al. algorithm [2], DEC2 can obtain (see the appendix for further details)

\[
Pr\{d_i = |R_i^f\} = p(z_k|d_i = 0)\frac{Pr\{d_k = 1\}}{W_{R_i}^2}
\]

where \( p(z|d = 0), p(x|d = 1) \), and \( W_{R_i}^2 \) are determined by DEC2, the channel, and DEC1, respectively. If \( p(A(d)|d = 0, i \in GF(q) \) are known by DEC2, then the following can be obtained:

\[
Pr\{d_i = |R_i^f\} = p(A(d_i)|d_i = 0)\frac{Pr\{d_k = 1\}}{W_{R_i}^2}
\]

where \( W_{R_i}^2 \) is determined by DEC2. So the decoder can make a decision by

\[
d_i = \begin{cases} 0 & \text{if } Pr\{d_i = |R_i^f\} > Pr\{d_i = |R_i^f|\} \\ 1 & \text{for all } j \in GF(q) \end{cases}
\]

In our algorithm, the following equations are used (see the appendix for further details):

\[
p(A(d_i)|d_i = 0) = Pr\{d_i = |R_i^f|/p(z_k|d_i = 0) = i \}
\]

and

\[
p(z_k|d_i = 0) = Pr\{d_i = |R_i^f|/p(A(d_i)|d_i = 0) = i \}
\]

Thus, if \( p(z_k|d_i = 0) = 1 \) for all \( i \in GF(q) \) in the first iteration, the iterative decoding can proceed.

Fig. 2 Bit error rate performance over AWGN channel

(i) 4-ary code (\( N = 99 \)), (ii) binary code (\( N = 192 \))

(iii) binary code (\( N = 99 \)), (iv) binary code (\( N = 100 \))

Ending iterations: DEC1 and DEC2 are deterministic machines. For a given \( \{X_i\}, \{Y_i\} \) and \( \{Y_0\} \), if \( \{Z_i\} \) no longer changes after some iterations, \( \{d_i\} \) will no longer change. Define

\[
SZ = \frac{1}{N} \sum_{h=0}^{N-1} \sum_{k=0}^{q-1} [p(z_k|d_k = i)]^2
\]

Let \( SZ \) and \( SZ_{\alpha} \) denote \( SZ \)'s value of \( \alpha \)th and \( (q-1) \)th step iterations, respectively. For a small enough positive number \( \delta \), if

\[
\frac{SZ_{\alpha} - SZ_{\alpha-1}}{SZ_{\alpha}} \leq \delta
\]

is satisfied, then iterative operation can be stopped. In our simulations, \( \delta = 10^{-5} \) was used.

Results: Simulations were performed for (37,21) half rate binary turbo code and 4-ary turbo code over the AWGN channels. 2PSK modulation was assumed. An 'odd-even' block interleaver [3] of 9 x 11 and maximum number of 18 iterations were used for both binary and 4-ary codes. The results are included in Figs. 2 and 3. For comparison, the results for binary turbo codes of length \( N = 100 \) [4] and \( N = 192 \) [3] are also included in Fig. 2. Both of them were obtained using Robertson's algorithm [4] after 6 and 10 iterations, respectively.