The Super-Trellis Structure of Turbo Codes

Marco Breiling, *Student Member, IEEE,* and
Lajos Hanzo, *Senior Member, IEEE*

***Abstract*—In this contribution we derive the super-trellis structure of turbo codes. We show that this structure and its associated decoding complexity depend strongly on the interleaver applied in the turbo encoder. We provide upper bounds for the super-trellis complexity. Turbo codes are usually decoded by an iterative decoding algorithm, which is suboptimum. Applying the super-trellis structure, we can optimally decode simple turbo codes and compare the associated bit-error rate results to those of iterative algorithms.**

***Index Terms*—Code trellis, iterative decoding, MLSE decoding, turbo codes.**

I. Introduction

Turbo codes, which were proposed by Berrou, Glavieux, and Thitimajshima in 1993 [1], are at time of writing the most power-efficient binary channel codes for medium bit-error rates (BERs) [2]. They form a class of soft-input decodable block codes of relatively large information word lengths (typically longer than 1000 bits), exhibiting various coding rates. The encoder of a turbo code consists mainly of two so-called component encoders, which are joined by an interleaver. The first component encoder encodes the information symbols directly, whereas the second component encoder encodes a permuted version of the same information symbols. The decoding of a long turbo code is typically accomplished using an iterative decoding algorithm, which is related to Gallager's classical probabilistic decoding algorithm [3]. In contrast to the latter algorithm, in a turbo decoder, there are two component decoders operating in unison and passing soft information to each other in a feedback loop. The component decoders must be capable of generating soft outputs corresponding to each information symbol's *a posteriori* probability from the soft inputs corresponding to the *a priori* probability of the respective information and code symbols. State-of-the-art component decoders for carrying out this task are above all the maximum-*a-posteriori* symbol-by-symbol estimation (MAPSSE) algorithm [4] and the soft-output Viterbi algorithm (SOVA) [5]. A vital part of a turbo encoder is its constituent interleaver. Reference [6] shows that if the average performance is evaluated for the range of all possible interleavers, the BER performance of the iterative decoder—although it is suboptimum—converges to the performance of a maximum-likelihood sequence estimation (MLSE) algorithm for medium and high signal-to-noise ratios (SNRs). Further contributions in the above mentioned literature focus on the weight distribution profile of turbo codes [7], [8]. Since the power efficiency of turbo codes deteriorates for low BERs, various methods have been proposed for improving the performance of turbo codes by designing the constituent interleaver appropriately [9], [10]. Conventional turbo codes rely on convolutional component codes, which is what we will restrict ourselves to in this contribution. However, there have also been proposed block component codes in the literature.

Manuscript received December 12, 1997; revised April 4, 2000. This work was supported by the EPSRC, U.K. in the framework of the Contract GR/K 74043 and of the Fraunhofer Gesellschaft, Institut für Integrierte Schaltungen, Erlangen, Germany, of the European Commission and of the Mobile VCE.

M. Breiling was with the Department of Electronics and Computer Science, University of Southampton, SO17 1BJ, U.K. He is now with Lehrstuhl für Nachrichtentechnik II, Universität Erlangen-Nürnberg, D-91058 Erlangen, Germany (e-mail: breiling@LNT.de).

L. Hanzo is with the Department of Electronics and Computer Science, University of Southampton, SO17 1BJ, U.K. (e-mail: lh@ecs.soton.ac.uk).

Communicated by N. Seshadri, Associate Editor for Coding Techniques.
Publisher Item Identifier S 0018-9448(00)07002-4.

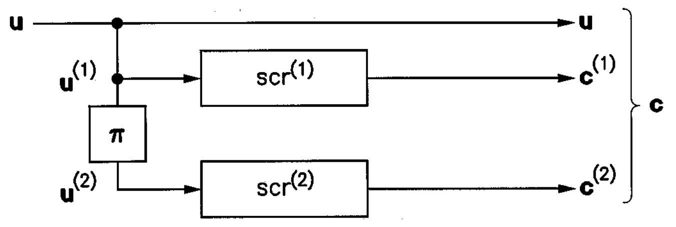


Fig. 1. System model of the turbo encoder.

The outline of the correspondence is as follows. Section II presents a model of the turbo encoder. Section III then provides a derivation of the super-trellis structure of turbo codes. Section IV discusses the complexity of the super-trellises, while Section V gives a comparison between the performance of an optimum decoder for turbo codes and the conventional iterative decoder. Finally, Section VI discusses the results obtained, followed by our conclusions in Section VII.

II. System Model and Terminology

Fig. 1 shows the model of a conventional turbo encoder. The encoder consists of three parallel branches connected to the encoder input, which generate the three constituent parts of the codeword $\boldsymbol{c} = (\boldsymbol{u}; \boldsymbol{c}^{(1)}; \boldsymbol{c}^{(2)})$. The vector of K binary encoder input symbols used for generating a codeword is referred to as the *information word* $\boldsymbol{u} = (u_1; \ldots; u_K)$. This information word $\boldsymbol{u}$ directly forms the systematic part of the codeword. The other two branches, referred to as the first and the second *component*, respectively, contain the scramblers $\mathsf{scr}^{(1)}$ and $\mathsf{scr}^{(2)}$, which are also often referred to as component encoders, processing the information symbols in order to generate the component codewords $\boldsymbol{c}^{(1)}$ and $\boldsymbol{c}^{(2)}$ constituting the parity part of the codeword $\boldsymbol{c}$. Throughout the correspondence, we use the following terminology. Capital letters denote random variables, whereas lower case letters represent their specific manifestations. Variables with a superscript, such as $\bullet^{(1)}$ and $\bullet^{(2)}$, are associated with the first and second component, respectively. The first component scrambler $\mathsf{scr}^{(1)}$ is fed with the information symbols $\boldsymbol{u}^{(1)} = \boldsymbol{u}$ obeying their original ordering. The second component scrambler $\mathsf{scr}^{(2)}$ is fed with a *permuted* version $\boldsymbol{u}^{(2)}$ of the information word. The third branch contains therefore an interleaver, which is going to play a major role in our forthcoming elaborations. The vector containing the element-wise transpositions performed by the interleaver is denoted by $\boldsymbol{\pi} = (\pi_1; \ldots; \pi_K)$. Interleaving of $\boldsymbol{u}^{(1)} = \boldsymbol{u}$ is achieved by permuting the vector elements generating $\boldsymbol{u}^{(2)}$, such that $u^{(2)}_{\pi_i} = u^{(1)}_i$ for $i = 1, \ldots, K$. For the sake of simplicity, we assume identical scramblers in both components. As usual, the scrambler is a binary shift register with a feedback branch as exemplified in Fig. 2. This structure is also often referred to as a recursive systematic convolutional (RSC) code. Any such scrambler represents a finite state machine (FSM) $\mathbb{F}_\mathrm{s} = (\mathcal{U}_\mathrm{s}; \mathcal{C}_\mathrm{s}; \mathcal{S}_\mathrm{s}; \gamma_\mathrm{s})$, whose input value set is binary $\mathcal{U}_\mathrm{s} \in \{0; 1\}$ and the corresponding output value set is also binary $\mathcal{C}_\mathrm{s} \in \{0; 1\}$. We note here explicitly that the subscript s refers to the state machine of the component scrambler. For a scrambler of memory M, the state set $\mathcal{S}_\mathrm{s} = \{0; \ldots; 2^M - 1\}$ consists of $\|\mathcal{S}_\mathrm{s}\| = 2^M$ legitimate states, which correspond to the states of the scrambler shift register, i.e., to vectors of M binary elements. The mapping $\gamma_\mathrm{s} : (\mathcal{S}_\mathrm{s} \times \mathcal{U}_\mathrm{s}) \to (\mathcal{S}_\mathrm{s} \times \mathcal{C}_\mathrm{s})$ represents the state transitions,

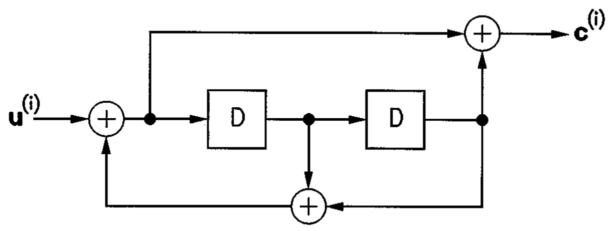


Fig. 2. Example of a component scrambler or component encoder with feedback and feed-forward polynomials expressed in octal form as 7 and 5, respectively.

where for every pair (s, u) with $s \in \mathcal{S}_s$ (predecessor state) and $u \in \mathcal{U}_s$ (scrambler input) a pair (s', c) with $s' \in \mathcal{S}_s$ (successor state) and $c \in \mathcal{C}_s$ (scrambler output) is specified. An FSM can be graphically characterized by its state transition diagram or, if we also incorporate the elapse of time, by its trellis. Fig. 3 displays the trellis of the scrambler of Fig. 2. It consists of identical trellis segments, since the scrambler FSM is time-invariant. The labels along the trellis transitions, e.g., $1 \rightarrow 0$, denote the scrambler input and output symbols, respectively, which are associated with the specific state transition.

Any FSM $\mathbb{F}(t) = (\mathcal{U}(t); \mathcal{C}(t); \mathcal{S}(t); \gamma(t))$—which can be potentially also time-variant, as indicated by (t)—exhibits the Markovian property, i.e., the state transition at a given discrete time instant $t+1$ with the associated ending state S_{t+1} and output symbol C_{t+1} depends only on the starting state S_t and the input symbol U_{t+1} (and possibly on the time t, if the FSM is time-variant, which manifests itself in a trellis structure that is different at different time instants t in the sequence of consecutive trellis stages—an issue to be augmented in more depth at a later stage), but not on any of the previous states $S_{t'}$ or input symbols $U_{t'+1}$ with $t' < t$. Observe that a transition and its associated input and output symbols have the same time index as the *terminating* state of the transition.

In our forthcoming elaborations, we will make use of the following three terms. The *pre-history* of an FSM as regards to time t represents the state transitions and their associated FSM input, that the FSM traversed through before and including the time instant t, i.e., the trellis section left of S_t in Fig. 3. By contrast, the *post-history* represents the transitions after and excluding time instant t, i.e., the trellis section right of S_t. The current state S_t represents, therefore, the *interface* between the pre-history of the FSM and its post-history in the following sense.

If a sequence of FSM states

$$\boldsymbol{S}_{t-} = (\ldots; S_{t-2}; S_{t-1}; S_t = s), \qquad s \in \mathcal{S}_s$$

corresponds to a valid sequence of state transitions constituting the pre-history with respect to time t and a sequence

$$\boldsymbol{S}_{t+} = (S_t = s; S_{t+1}; S_{t+2}; \ldots)$$

corresponds to a valid sequence of state transitions constituting the post-history, then we can combine the sequences, and

$$\boldsymbol{S} = (\ldots; S_{t-2}; S_{t-1}; S_t = s; S_{t+1}; S_{t+2}; \ldots)$$

represents a valid sequence of state transitions for the FSM. The only necessary and sufficient condition for combining the pre- and post-history is that the value s of the interface state S_t between the left and right trellis sections must be the same. If $\boldsymbol{S}_{t+} = (S_t = \tilde{s}; S_{t+1}; S_{t+2}; \ldots)$ with $s \neq \tilde{s}$, then combining the pre- and post-history results in an invalid sequence, i.e., in an invalid history for the FSM. In other words, such a sequence of state transitions is illegitimate, since both $s_t = \tilde{s}$ and $s_t \neq \tilde{s}$ should hold, implying a contradiction. Note that the terms "sequence of states" and "sequence of state transitions" are used synonymously, since the transitions can directly be derived from the sequence of states.

III. Introducing the Turbo Code Super-Trellis

In this section we will develop a model of an FSM

$$\mathbb{F}_{\mathrm{T}}(t) = (\mathcal{U}_{\mathrm{T}}; \mathcal{C}_{\mathrm{T}}; \mathcal{S}_{\mathrm{T}}(t); \gamma_{\mathrm{T}}(t))$$

for a **T**urbo encoder, where the subscript T refers to the state machine of the turbo encoder. The parameter t indicates that the turbo encoder FSM is time-variant, which manifests itself in a trellis structure that is different at different time instants t, exhibiting different legitimate transitions at different instants t, as we will show at a later stage. It is possible to find the turbo encoder's FSM by modeling all of its four constituent elements in Fig. 1 by their respective FSMs and combining these. For the two-component scramblers of Fig. 1 it is relatively straightforward to find the corresponding FSMs (see above). For the systematic branch of the encoder in Fig. 1, the corresponding FSM consists of a single state, where the output symbol equals the input symbol. The only device in the turbo encoder that poses difficulties is the interleaver, which introduces memory, since the complete vector of input symbols $\boldsymbol{u}^{(1)}$ must have been inserted into the interleaver, before the vector of output symbols $\boldsymbol{u}^{(2)}$ could be read out. It is therefore cumbersome to model the interleaver by an FSM, which would be complex. Hence we will choose a different way of finding the FSM of the complete turbo encoder.

The trellis associated with the turbo encoder FSM will be referred to as the turbo code's *super-trellis* in order to distinguish it clearly from the component scrambler trellises. Several proposals have already been made to find a tree representation of a turbo code [11] and its super-trellis (e.g., [6], where it is referred to as the hyper-trellis). In contrast to [6], the structure presented in this contribution differs in that the complexity of the super-trellis is not only dependent on the interleaver length K but also on the interleaver structure. We will show that for simple interleavers the complexity of the super-trellis can be drastically reduced. In these cases, an optimum turbo decoder based on this super-trellis can be implemented and its performance can be directly compared to that of the suboptimum iterative turbo decoder. Let us continue our discourse with an example.

A. Turbo Encoder Super State

Example 1: For the sake of explanation, let us consider a turbo code incorporating the scrambler of Fig. 2 and a simple two-column interleaver of total interleaving length of K bits obeying the mapping at the bottom of the following page.

Let us now introduce the concept of super-trellis in the context of turbo codes. We assume that the turbo encoder's input symbols are given by $U_1, \ldots, U_K$, where the ordering of the symbols is important and for the turbo encoder's set we have $\mathcal{U}_{\mathrm{T}} \in \{0; 1\}$. The time-domain transition index of the super-trellis is denoted by t, which can be different from the corresponding transition index of the individual trellises associated with the component codes, as will be shown later. Let us consider the positions of the first five input symbols, namely, that of $U_1, \ldots, U_5$ in both component trellises, which is illustrated in Fig. 4. Entering U_5 in the turbo encoder's FSM corresponds to $t = 5$.

In the upper component trellis in Fig. 4 the symbols $U_1^{(1)}, \ldots, U_5^{(1)}$ are in consecutive positions. The transitions

$$\boldsymbol{S}_{5-}^{(1)} = (S_0^{(1)} = 0; S_1^{(1)}; \ldots; S_5^{(1)})$$

corresponding to these input symbols constitute therefore the pre-history of the first component trellis with respect to $t = 5$. (We note that the component trellises emanate at $t = 0$ from the zero-state $S_0^{(1)} = 0$.)

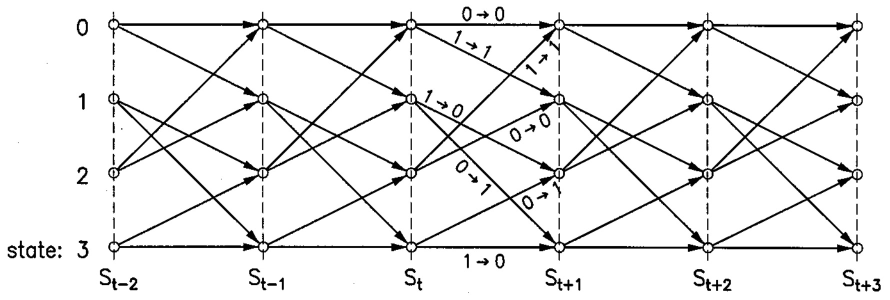
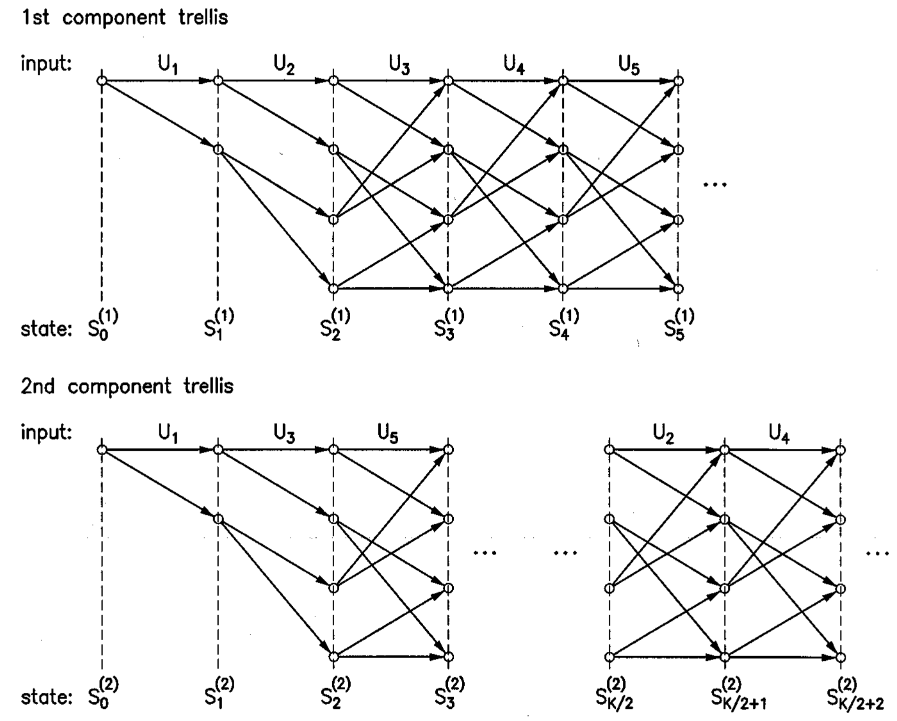


Fig. 3. Trellis of the scrambler of Fig. 2 with memory $M = 2$.



1st component trellis
input: U_1 U_2 U_3 U_4 U_5
state: $S_0^{(1)}$ $S_1^{(1)}$ $S_2^{(1)}$ $S_3^{(1)}$ $S_4^{(1)}$ $S_5^{(1)}$
2nd component trellis
input: U_1 U_3 U_5 U_2 U_4
state: $S_0^{(2)}$ $S_1^{(2)}$ $S_2^{(2)}$ $S_3^{(2)}$ $S_{K/2}^{(2)}$ $S_{K/2+1}^{(2)}$ $S_{K/2+2}^{(2)}$

Fig. 4. Trellis transitions in the two-component scrambler trellises up to time instant $t = 5$.

The post-history of the first component trellis *with respect to* (in the following abbreviated by wrt) $t = 5$ will be constructed from the input symbols $U_6^{(1)}, \ldots, U_K^{(1)}$ and from the corresponding state transitions $\boldsymbol{S}_{5+}^{(1)} = (S_5^{(1)}; S_6^{(1)}; \ldots)$. The interface between the pre- and post-history of the first component trellis wrt the time instant $t = 5$ is represented by $S_5^{(1)}$.

Due to the permutation performed by the interleaver in Fig. 1 in the second component trellis of Fig. 4, the same five bits $U_1, \ldots, U_5$ are transformed into the input symbols $U_1^{(2)}$, $U_2^{(2)}$, $U_3^{(2)}$ and $U_{K/2+1}^{(2)}$, $U_{K/2+2}^{(2)}$. Hence, the corresponding transitions are located in the second component trellis seen at the bottom of Fig. 4 in two different sections. Consequently, the pre-history wrt $t = 5$—i.e., wrt bits $U_1, \ldots, U_5$, which belong to the first five transitions of the turbo code super-trellis associated with the discrete-time instants of $t = 0, \ldots, 5$—corresponds to the transitions of the second component trellis in these two distinct trellis sections of Fig. 4. The post-history of the secund component trellis wrt $t = 5$ is constituted by the two remaining sections of the trellis, which are indicated by dots in Fig. 4.

Position	1	2	3	4	5	…	$K/2+1$	$K/2+2$	…
$\boldsymbol{U}^{(1)}$	U_1	U_2	U_3	U_4	U_5	…	$U_{K/2+1}$	$U_{K/2+2}$	…
$\boldsymbol{U}^{(2)}$	U_1	U_3	U_5	U_7	U_9	…	U_2	U_4	…

The interface between the pre- and post-history, which is associated with $t = 5$, is formed in the second component trellis by the set $(S_3^{(2)}; S_{K/2}^{(2)}; S_{K/2+2}^{(2)})$, which we will refer to as the "interface" states in the second component trellis.

These issues are further augmented below as follows. An arbitrary input sequence is given by

$$(U_1; U_2; U_3; U_4; U_5) = \left(U_1^{(2)}; U_{K/2+1}^{(2)}; U_2^{(2)}; U_{K/2+2}^{(2)}; U_3^{(2)}\right)$$

where the corresponding state transitions in both sections of the second component trellis at the bottom of Fig. 4—which constitute the pre-history related to $t = 5$—are represented by

$$\boldsymbol{S}_{5-}^{(2)} = \Big(S_0^{(2)} = 0;\ S_1^{(2)};\ S_2^{(2)};\ S_3^{(2)} = s_3^{(2)}; \\ S_{K/2}^{(2)} = s_{K/2}^{(2)};\ S_{K/2+1}^{(2)};\ S_{K/2+2}^{(2)} = s_{K/2+2}^{(2)}\Big).$$

Every valid post-history

$$\boldsymbol{S}_{5+}^{(2)} = \Big(S_3^{(2)} = s_3^{(2)};\ S_4^{(2)};\ \ldots;\ S_{K/2}^{(2)} = s_{K/2}^{(2)}; \\ S_{K/2+2}^{(2)} = s_{K/2+2}^{(2)};\ S_{K/2+3}^{(2)};\ \ldots\Big)$$

can, therefore, be combined with $\boldsymbol{S}_{5-}^{(2)}$ and forms a valid sequence of state transitions for the second component trellis, if and only if the interface states $(S_3^{(2)}; S_{K/2}^{(2)}; S_{K/2+2}^{(2)})$ have identical values $(s_3^{(2)}; s_{K/2}^{(2)}; s_{K/2+2}^{(2)})$ in both $\boldsymbol{S}_{5-}^{(2)}$ and $\boldsymbol{S}_{5+}^{(2)}$.

So far we have elaborated on the details of the pre-history related to $t = 5$ in the first and second component trellis, as well as on their interfaces to the post-history. Hence, in the turbo encoder only these interface states are important for the encoding of the forthcoming input symbols $U_6; U_7; \ldots$. The complete memory of the turbo encoder at time instant $t = 5$, which is required for further encoding operations, can be bundled in the 4-tuple $(S_5^{(1)}; S_3^{(2)}; S_{K/2}^{(2)}; S_{K/2+2}^{(2)})$, which comprises all interfaces of both component trellises. It can be seen that every valid pre-history pair of state transitions $\boldsymbol{S}_{5-}^{*} = (\boldsymbol{S}_{5-}^{(1)}; \boldsymbol{S}_{5-}^{(2)})$ can be combined with every valid post-history pair $\boldsymbol{S}_{5+}^{*} = (\boldsymbol{S}_{5+}^{(1)}; \boldsymbol{S}_{5+}^{(2)})$, in order to form a valid pair of state transition sequences for the two-component trellises, if and only if the interface states $(S_5^{(1)}; S_3^{(2)}; S_{K/2}^{(2)}; S_{K/2+2}^{(2)})$ in $\boldsymbol{S}_{5-}^{*}$ are identical to those in $\boldsymbol{S}_{5+}^{*}$, namely, $(s_5^{(1)}; s_3^{(2)}; s_{K/2}^{(2)}; s_{K/2+2}^{(2)})$.

Hence, as a super state of the example turbo encoder's FSM at time instant $t = 5$ we define the above 4-tuple, which is constituted by the states of the component scramblers of the FSMs as follows:

$$S_5^* \overset{\text{def}}{=} \left(S_5^{(1)}; S_3^{(2)}; S_{K/2}^{(2)}; S_{K/2+2}^{(2)}\right).$$

B. Turbo Encoder Super-Trellis

Let us now investigate the migration from $t = 5$ to $t = 6$, or equivalently, let us search for the super state S_6^* under the assumption that the super state S_5^* and the most recent turbo encoder input symbol U_6 are known. For the input symbols of the first component scrambler we have $U_6 = U_6^{(1)}$, such that the transition associated with U_6 in the first component trellis is directly adjacent to the transitions in the pre-history $\boldsymbol{S}_{5-}^{(1)}$ (upper trellis in Fig. 4). Since the value $s_5^{(1)}$ of $S_5^{(1)}$ is known from S_5^*, the value u_6 of U_6 dictates the transition $(s_5^{(1)}; s_6^{(1)})$. The pre-history sequence of the state transitions of the first component code simply extends from $\boldsymbol{S}_{5-}^{(1)}$ to

$$\boldsymbol{S}_{6-}^{(1)} = (S_0^{(1)} = 0; \ldots; S_5^{(1)} = s_5^{(1)}; S_6^{(1)} = s_6^{(1)}).$$

With the aid of Fig. 4 we can see that for the input symbols of the second component scrambler the relation $U_6 = U_{K/2+3}^{(2)}$ holds. In Fig. 4, the corresponding transition (index $K/2+3$ of the second component trellis) therefore emanates from $S_{K/2+2}^{(2)}$ in the right trellis section, which belongs to the pre-history. Since the value $s_{K/2+2}^{(2)}$ of $S_{K/2+2}^{(2)}$ is known from S_5^*, u_6 dictates the transition $(s_{K/2+2}^{(2)}; s_{K/2+3}^{(2)})$. Hence the pre-history of state transitions for the second component code becomes

$$\boldsymbol{S}_{6-}^{(2)} = \Big(S_0^{(2)} = 0;\ \ldots;\ S_3^{(2)} = s_3^{(2)};\ S_{K/2}^{(2)};\ \ldots; \\ S_{K/2+2}^{(2)} = s_{K/2+2}^{(2)};\ S_{K/2+3}^{(2)} = s_{K/2+3}^{(2)}\Big).$$

From this we can easily infer the interfaces of the pair $\boldsymbol{S}_{6-}^{*} = (\boldsymbol{S}_{6-}^{(1)}; \boldsymbol{S}_{6-}^{(2)})$, which is formed by the two pre-histories regarding time instant $t = 6$. Hence the super state of the turbo encoder at this time instant can be identified by exploiting the knowledge of $S_5^* = (s_5^{(1)}; s_3^{(2)}; s_{K/2}^{(2)}; s_{K/2+2}^{(2)})$ yielding

$$S_6^* \overset{\text{def}}{=} \left(S_6^{(1)}; S_3^{(2)}; S_{K/2}^{(2)}; S_{K/2+3}^{(2)}\right) = \left(s_6^{(1)}; s_3^{(2)}; s_{K/2}^{(2)}; s_{K/2+3}^{(2)}\right)$$

where both the state transitions $(s_5^{(1)}; s_6^{(1)})$ and $(s_{K/2+2}^{(2)}; s_{K/2+3}^{(2)})$ are associated with the value u_6 of the input symbol U_6.

C. Generalized Definition of the Turbo Encoder Super States

As an extension of our previous introductory elaborations in this section we will present a generalized definition of the super state transitions in the super-trellis, which also defines implicitly the super states associated with the super-trellis. For the sake of illustration the definition will be followed by the example of a simple super-trellis in Section III-D.

First we have to introduce a set of indices $\mathcal{I}_{\text{pre}}^{(2)}(t)$, which contains the number of transitions belonging to the pre-history of the second component trellis wrt time instant t. If and only if an index obeys $i \in \mathcal{I}_{\text{pre}}^{(2)}(t)$, the transition with this index is part of the pre-history of the second component trellis at time t. It follows immediately that a state $S_j^{(2)}$ with index j is an interface state in the second component trellis at time t, if and only if one of the two adjacent transitions belongs to the pre-history and the other one does not, i.e., if and only if one of the following two cases is true:

$$j \in \mathcal{I}_{\text{pre}}^{(2)}(t) \quad \wedge \quad (j+1) \notin \mathcal{I}_{\text{pre}}^{(2)}(t)$$

or

$$j \notin \mathcal{I}_{\text{pre}}^{(2)}(t) \quad \wedge \quad (j+1) \in \mathcal{I}_{\text{pre}}^{(2)}(t).$$

For an arbitrary interleaver $\boldsymbol{\pi}$, which was shown in Fig. 1, we can now define the super states of the turbo encoder FSM as follows. At time instant $t = 0$ both component trellises emanate from the zero state $S_0^{(1)} = S_0^{(2)} = 0$. Hence the following equation holds for the super state:

$$S_0^* \overset{\text{def}}{=} \left(S_0^{(1)}; S_0^{(2)}\right) = (0; 0). \tag{1}$$

We initialize the index set by including only the (nonexistent) transition index 0: $\mathcal{I}_{\text{pre}}^{(2)}(0) = \{0\}$.

Evolution from time instant t to $t+1$: the previous super state s_t^* is assumed to be known. For the first component code the relation $U_{t+1} = U_{t+1}^{(1)}$ holds, hence the input symbol $U_{t+1} = u_{t+1}$ specifies the transition $(S_t^{(1)}; S_{t+1}^{(1)}) = (s_t^{(1)}; s_{t+1}^{(1)})$ in the first component code. (In the first component trellis at time instant t the state with index t, $S_t^{(1)}$ is always the interface state contained in the super state S_t^*, and hence the value $s_t^{(1)}$ is known from the vector s_t^*.) For the second component code we have $U_{t+1} = U_j^{(2)}$ with $j = \pi_{t+1}$. The index of the associated transition in the second component trellis for $t+1$ is j, such that the corresponding transition becomes $(S_{j-1}^{(2)}; S_j^{(2)})$. This transition is not part of the pre-history wrt t and hence $j \notin \mathcal{I}_{\text{pre}}^{(2)}(t)$. Updating the set of indices means that $\mathcal{I}_{\text{pre}}^{(2)}(t+1) = \mathcal{I}_{\text{pre}}^{(2)}(t) \cup \{j\}$. In order to proceed from $(j-1)$ to j, we have to distinguish between four different cases or scenarios, as far as the second component trellis is concerned, which will be detailed below and will also be augmented in the context of an example in Section III-D and Fig. 6. In fact, the reader may find it beneficial to follow the philosophy of our forthcoming generic discussions on a case-by-case basis by referring to the specific example of Section III-D and Fig. 6.

1) In order to encounter our first scenario, the following condition must be met: $j-1 \in \mathcal{I}^{(2)}_{\text{pre}}(t)$ and $j+1 \notin \mathcal{I}^{(2)}_{\text{pre}}(t)$. The state with index $j-1$, $S^{(2)}_{j-1}$, thus is one of the interface states in the vector, representing the super state S^*_t, whereas the state with index j, $S^{(2)}_j$, is not an interface state at time instant t. This implies that the transition $(S^{(2)}_{j-1};\, S^{(2)}_j) = (s^{(2)}_{j-1};\, s^{(2)}_j)$—which is due to $U^{(2)}_j = u_{t+1}$—is directly adjacent to the state transitions in the second component trellis already belonging to the pre-history in the second component code wrt the time instant t. From a graphical point of view the transition is located directly to the right of the transitions, which have already been encountered in the trellis of Fig. 4. The new super state S^*_{t+1} of the turbo encoder FSM can be inferred from the old super state S^*_t, by substituting the old interface in the first component trellis—namely, $S^{(1)}_t = s^{(1)}_t$—by the corresponding new one—namely, by $S^{(1)}_{t+1} = s^{(1)}_{t+1}$—and also the old interface state $S^{(2)}_{j-1} = s^{(2)}_{j-1}$ by the corresponding new state $S^{(2)}_j = s^{(2)}_j$, where $(s^{(2)}_{j-1};\, s^{(2)}_j)$ is associated with u_{t+1}. We will refer to this transition from t to $t+1$ as *right-extension*, since in the second component trellis a section is extended to the right in Fig. 4. Again, these issues will be exemplified in Section III-D and Fig. 6, where this scenario is encountered for transitions $t = 0 \to 1$, $t = 2 \to 3$, $t = 3 \to 4$, $t = 4 \to 5$, and $t = 7 \to 8$.

2) The condition for encountering our second scenario is as follows: $j-1 \notin \mathcal{I}^{(2)}_{\text{pre}}(t)$ and $j+1 \in \mathcal{I}^{(2)}_{\text{pre}}(t)$. The state with index j, $S^{(2)}_j$, then constitutes a component of the vector representing the super state S^*_t ($S^{(2)}_j$ is hence an interface state in the second component trellis at time instant t), whereas the state with index $j-1$ is not an interface state. Accordingly, the transition $(S^{(2)}_{j-1};\, S^{(2)}_j)$ associated with $U^{(2)}_j$ is directly adjacent to the state transitions, which already belong to the pre-history wrt t. In order to obtain the new super state S^*_{t+1} from the vector, which represents the old super state S^*_t, one has to replace the old interface $S^{(1)}_t$ of the first component code by the new $S^{(1)}_{t+1}$ and also the old interface $S^{(2)}_j$ in the second component code by the corresponding new $S^{(2)}_{j-1}$. In analogy to scenario 1), we will refer to this case as *left-extension*.

3) The third potential scenario encountered by the second trellis is, when $j-1 \notin \mathcal{I}^{(2)}_{\text{pre}}(t)$ and $j+1 \notin \mathcal{I}^{(2)}_{\text{pre}}(t)$. This implies that neither the state associated with the index $j-1$, i.e., $S^{(2)}_{j-1}$, nor that with the index j, i.e., $S^{(2)}_j$ is contained in the vector corresponding to the super state S^*_t. The transition $(S^{(2)}_{j-1};\, S^{(2)}_j)$, which is due to $U^{(2)}_j$, is not adjacent to any of the transitions, which are already contained in the pre-history wrt to t. In the pre-history wrt $t+1$ this transition hence constitutes a separate section of the second component trellis, which consists of only one transition. This section possesses the two interface states $S^{(2)}_{j-1}$ and $S^{(2)}_j$. In order to obtain the super state S^*_{t+1}, one has to substitute the interface state $S^{(1)}_t$ by $S^{(1)}_{t+1}$ in the super state S^*_t and, in addition, one has to extend the super state vector by these two new interface states—namely, by $S^{(2)}_{j-1}$ and $S^{(2)}_j$—in the second component trellis. Observe, however, that neither the value $s^{(2)}_{j-1}$ of $S^{(2)}_{j-1}$ nor the value $s^{(2)}_j$ of $S^{(2)}_j$ is specified by $S^*_t = s^*_t$, which represents the interface of the pre-history wrt t in the super-trellis.

The pair $(S^{(2)}_{j-1};\, S^{(2)}_j)$ can therefore assume all legitimate values $(s^{(2)}_{j-1};\, s^{(2)}_j)$ within the vector S^*_{t+1}, which represent valid state transitions and which correspond to the input symbol $U^{(2)}_j = u_{t+1}$. The consequence is that if the super state S^*_t is known, several values are possible for the successor super state S^*_{t+1} with equal probability. Specifically, the new interface state $S^{(2)}_{j-1}$—which is contained in the associated super state vector—can assume all possible states $S^{(2)}_{j-1} \in \mathcal{S}_{\text{s}}$ and the other new interface state $S^{(2)}_j$ will be $s^{(2)}_j$, determined by the transition $(s^{(2)}_{j-1};\, s^{(2)}_j)$ due to the input symbol $U^{(2)}_j = u_{t+1}$. We will refer to this scenario as the *opening* of a new section in the second component trellis. These aspects will be revisited in Section III-D and Fig. 6, where the transition $t = 1 \to 2$ corresponds to this specific scenario.

4) The last possible scenario encountered by the second trellis is, when $j-1 \in \mathcal{I}^{(2)}_{\text{pre}}(t)$ and $j+1 \in \mathcal{I}^{(2)}_{\text{pre}}(t)$. The state associated with the index $j-1$, i.e., $S^{(2)}_{j-1}$, as well as the one with the index j, i.e., $S^{(2)}_j$, are contained in the vector corresponding to super state S^*_t. The specific state values $s^{(2)}_{j-1}$ and $s^{(2)}_j$ representing the interfaces $S^{(2)}_{j-1}$ and $S^{(2)}_j$ have already been determined by means of S^*_t representing the interface of the pre-history wrt t in the super-trellis. Although the transition $(S^{(2)}_{j-1};\, S^{(2)}_j)$—which is the transition at time instant $t+1$ in the second component trellis—is not contained in the pre-history wrt t, nonetheless this transition is determined by S^*_t due to its fixed start- and end-states constituted by $s^{(2)}_{j-1}$ and $s^{(2)}_j$, respectively. We will revisit these issues in the context of Section III-D and Fig. 6, where the transition $t = 6 \to 7$ constitutes an example of scenario 4).

If this fixed transition $(s^{(2)}_{j-1};\, s^{(2)}_j)$ is legitimate and if it is associated with the value u_{t+1} of the most recent input symbol $U^{(2)}_j$, then two interface states are connected by means of this particular transition in the second component trellis. Hence, the gap between two disjoint trellis sections of the pre-history is closed, and therefore we refer to this scenario as the *fusion* of these two sections. The new super state S^*_{t+1} evolves from the vector S^*_t by removing the two interfaces $S^{(2)}_{j-1}$ and $S^{(2)}_j$ and by substituting $S^{(1)}_t$ by $S^{(1)}_{t+1}$.

However, it may happen that for the interface states $s^{(2)}_{j-1}$ and $s^{(2)}_j$—which are determined by $S^*_t = s^*_t$—the transition $(s^{(2)}_{j-1};\, s^{(2)}_j)$ is illegitimate, since this state transition is nonexistent for the FSM of the component scramblers. In this case, assuming the value s^*_t for S^*_t constitutes a contradiction due to the illegitimate transition and this transition is therefore deemed invalid. Furthermore, it can occur that although the transition $(s^{(2)}_{j-1};\, s^{(2)}_j)$ is legitimate, it is not associated with the current specific value u_{t+1} of the input symbol for the component scrambler FSM. Hence for this reason pairing the super state s^*_t and the input symbol u_{t+1} is impossible. Therefore, in the super-trellis *no* transition emerges from the super state $S^*_t = s^*_t$, which is associated with u_{t+1}. This may appear to be a contradiction, since this super state seems to have been reached due to the sequence of input symbols $U_1, \ldots, U_t$ and since in the super-trellis a path is supposed to exist for *all* possible sequences of input symbols $U_1, \ldots, U_{t+1}$, regardless of the specific value of the input symbol U_{t+1}. However, the *fusion* of two trellis sections must always be preceded by the *opening* of a new section in the second component trellis, and—as we infer from scenario 3)—several super states are reached with equal probability from a sequence of input symbols $U_1, \ldots, U_t$. When for example the first *opening* is executed, a single sequence $U_1, \ldots, U_t$ of input symbols leads to $\|\mathcal{S}_{\text{s}}\| = 2^M$ possible super states (see scenario 3) above). Hence it is understandable that in the process of a section *fusion* in the super-trellis, a proportion of $(2^M - 1)/2^M$ of all (super state/input symbol) pairs $(s^*_t;\, u_{t+1})$ represent invalid transitions. (For any interface state value $s^{(2)}_{j-1} \in \mathcal{S}_s$, which is determined by the super state value s^*_t, and for any input symbol value $u^{(2)}_j \in \mathcal{U}_s$, there exists only *one* of 2^M possible interface state values $s^{(2)}_j \in \mathcal{S}_s$, such that the component scrambler transition $(s^{(2)}_{j-1};\, s^{(2)}_j)$ is associated with the input symbol $u^{(2)}_j$.) We will further augment this concept by means of an example in Section III-D. Note that the trellis section in the first component trellis with its interface state $S^{(1)}_t$ is in all four above scenarios extended to the right, i.e., the new interface state is $S^{(1)}_{t+1}$.

Let us now elaborate a little further. The output symbols of the turbo encoder FSM—which are associated with a super state transition at

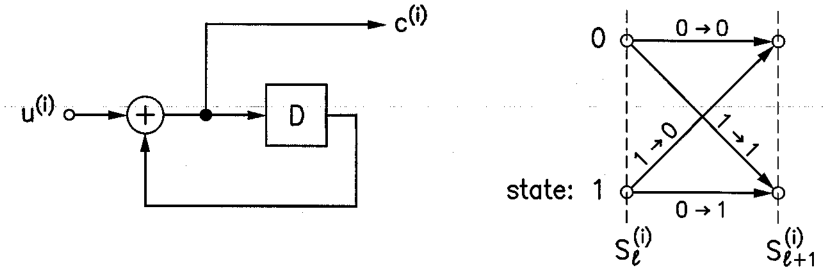


Fig. 5. Simple component scrambler and an associated trellis segments.

TABLE I
INTERLEAVING SCHEME FOR THE COMPONENT TRELLISES OF EXAMPLE 2

Position	1	2	3	4	5	6	7	8
$\mathbf{U}^{(1)}$	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8
$\mathbf{U}^{(2)}$	U_1	U_3	U_5	U_7	U_2	U_4	U_6	U_8

time instant $t+1$—consist of the systematic code symbol U_{t+1} and the output symbols emitted during the most recent state transitions in both component scrambler trellises. These are the state transitions belonging in both of the component trellises to the input symbol $U_{t+1}^{(1)} = U_{t+1}$ and $U_j^{(2)} = U_{t+1}$, $j = \pi_{t+1}$, respectively, and which therefore generate the output symbols $C_{t+1}^{(1)}$ and $C_j^{(2)}$, respectively. The output of the turbo encoder FSM $\mathbb{F}_T$ at time instant $t+1$ is therefore the vector

$$\overline{C}_{t+1} = \left(U_{t+1};\ C_{t+1}^{(1)};\ C_{\pi_{t+1}}^{(2)}\right)$$

and hence the relation $\mathcal{C}_T = [\{0;\ 1\}]^3$ holds for the output set. This constitutes a turbo code of rate $1/3$. At higher coding rates, the scrambler outputs are punctured, which also has to be taken into account in the output vector $\overline{C}$ of the turbo encoder FSM. Let us now illustrate the above concepts with the aid of another example.

D. Example of a Super-Trellis

Example 2: Let us now examine the super-trellis of a simple turbo code, incorporating the memory $M = 1$ scrambler of Fig. 5 and a 4 row×2 column rectangular interleaver resulting in $K = 8$, which is shown in Table I. The output of the turbo encoder is not punctured. The segments of the super-trellis that is developed in the forthcoming paragraphs for time instants $t = 0, \ldots, 8$, are displayed in Fig. 6. The index $j = \pi_{t+1}$ denotes in the second component trellis the transition belonging to time $t+1$, i.e., to the transition index $t+1$ in the super-trellis. Let us now consider Fig. 6, where at

$t = 0$: we initialize the super state to

$$S_0^* = (S_0^{(1)};\ S_0^{(2)}) = (0;\ 0)$$

and the index set to $\mathcal{I}_{\text{pre}}^{(2)}(0) = 0$.

$t = 0 \to 1$: We have $j = \pi_{t+1} = \pi_1 = 1$, since U_1 is at position 1 at the bottom of Table I. As $j-1 = 0 \in \mathcal{I}_{\text{pre}}^{(2)}(0)$ and $j+1 = 2 \notin \mathcal{I}_{\text{pre}}^{(2)}(0)$, this corresponds to scenario 1) described in Subsection III-C, i.e., the *right-extension* of an existing section in the second component trellis ($S_{j-1}^{(2)} = S_0^{(2)}$ is one of the interface states contained in S_0^*, while $S_j^{(2)}$ is not). Hence the new super state is $S_1^* = (S_1^{(1)};\ S_1^{(2)})$. The possible values for S_1^*—which depend on the value of U_1—are displayed in Table II.

$t = 1 \to 2$: We have $j = \pi_{t+1} = \pi_2 = 5$, since U_2 is at position 5 at the bottom of Table I. Now we find that $j-1 = 4 \notin \mathcal{I}_{\text{pre}}^{(2)}(1)$ and $j+1 = 6 \notin \mathcal{I}_{\text{pre}}^{(2)}(1)$, thus neither $S_{j-1}^{(2)} = S_4^{(2)}$ nor $S_j^{(2)} = S_5^{(2)}$ is contained in S_1^* of Fig. 6 at $t=1$. This is therefore scenario 3), representing the *opening* of a new section in the second component trellis. The new super state is $S_2^* = (S_2^{(1)};\ S_1^{(2)};\ S_4^{(2)};\ S_5^{(2)})$. For every value of S_1^* and every value of U_2, there are $2^M = 2^1 = 2$ possible values for $S_4^{(2)}$, as seen in Table II. At this stage, careful further tracing of the super-trellis evolution with the aid of Fig. 6 and Table II is helpful, in order to augment the associated operations.

$t = 2 \to 3$: This is scenario 1) again, i.e., a *right-extension*. The new superstate is $S_3^* = (S_3^{(1)};\ S_2^{(2)};\ S_4^{(2)};\ S_5^{(2)})$.

$t = 3 \to 4$: Scenario 1), *right-extension*, $S_4^* = (S_4^{(1)};\ S_2^{(2)};\ S_4^{(2)};\ S_6^{(2)})$.

$t = 4 \to 5$: Scenario 1), *right-extension*, $S_5^* = (S_5^{(1)};\ S_3^{(2)};\ S_4^{(2)};\ S_6^{(2)})$.

$t = 5 \to 6$: Scenario 1), *right-extension*, $S_6^* = (S_6^{(1)};\ S_3^{(2)};\ S_4^{(2)};\ S_7^{(2)})$.

$t = 6 \to 7$: $j = \pi_{t+1} = \pi_7 = 4$, since U_7 is at position 7 at the bottom of Table I. Since $j-1 = 3 \in \mathcal{I}_{\text{pre}}^{(2)}(6)$ and $j+1 = 5 \in \mathcal{I}_{\text{pre}}^{(2)}(6)$, i.e., both $S_3^{(2)}$ and $S_4^{(2)}$ are contained in S_6^*, this is scenario 4), corresponding to the *fusion* of two sections in the second component trellis of Fig. 6. As inferred from the trellis of Fig. 5, for $U_7 = 0$ the possible values of the state transition $(S_3^{(2)};\ S_4^{(2)})$ are $(0;\ 0)$ and $(1;\ 1)$, for $U_7 = 1$, it is $(0;\ 1)$ and $(1;\ 0)$. Depending on the value of U_7, the super state S_6^* containing other values for the pair $(S_3^{(2)};\ S_4^{(2)})$ thus has no successor super state. If there is a valid transition emerging from the super state S_6^*, which is associated with u_7, the successor super state is $S_7^* = (S_7^{(1)};\ ,S_7^{(2)})$. Otherwise, the transition is marked as "invalid" or illegitimate. Observe that a portion of $(2^M - 1)/2^M = 1/2$ of all transitions is invalid (see scenario 4) above.

$t = 7 \to 8$: Scenario 1), *right-extension* of the remaining section, $S_8^* = (S_8^{(1)};\ S_8^{(2)})$.

We see from Table II that S_8^* can only assume the values $(0;\ 0)$ and $(1;\ 1)$, since the last bit is the same in both the original and in the interleaved sequences. In other words, in our simple example at time instant $t = 8$ the memory of the interleaver in Table I has been exhausted. Hence both component scramblers have actually completed calculating the parity in the information word $\boldsymbol{u}$ and must hence be in the same state. For a random interleaver this may, however, be not the case.

If one or both component trellises are terminated by zeros, we impose the restriction of $S_K^{(1)} = 0$ and possibly $S_K^{(2)} = 0$. Only the super

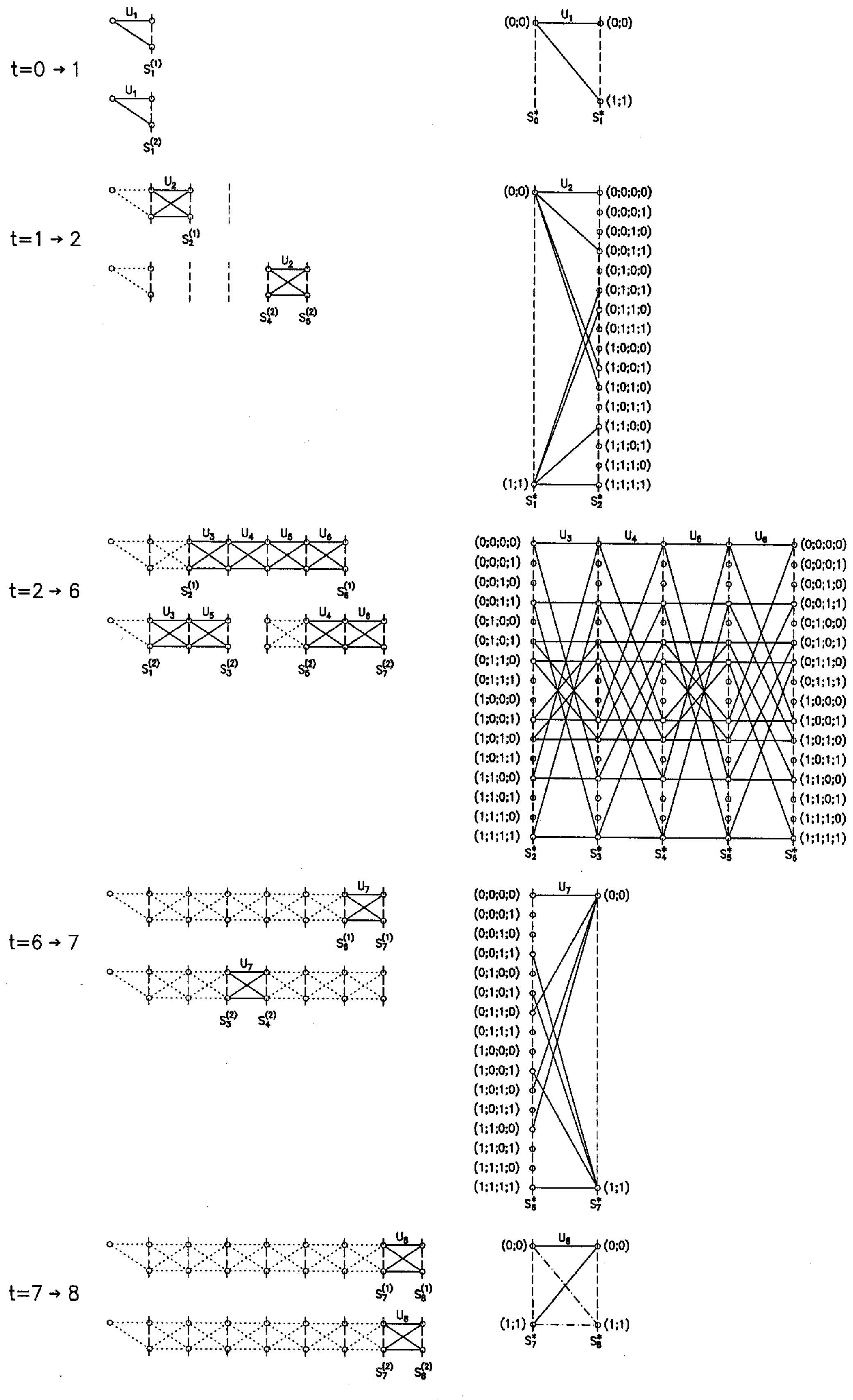


Fig. 6. Segments of the super-trellis for the turbo encoder FSM of Example 2.

TABLE II
STATE TRANSITIONS AND THEIR RESPECTIVE OUTPUT VECTORS FOR THE TURBO ENCODER SUPER-TRELLIS ASSOCIATED WITH THE COMPONENT CODES OF FIG. 5

t	S_t^*	U_{t+1}	S_{t+1}^*	$\tilde{C}_{t+1}$
0	(0;0)	0	(0;0)	(0;0;0)
		1	(1;1)	(1;1;1)
1	(0;0)	0	(0;0;0;0)	(0;0;0)
			(0;0;1;1)	(0;0;1)
		1	(1;0;0;1)	(1;1;1)
			(1;0;1;0)	(1;1;0)
	(1;1)	0	(1;1;0;0)	(0;1;0)
			(1;1;1;1)	(0;1;1)
		1	(0;1;0;1)	(1;0;1)
			(0;1;1;0)	(1;0;0)
2	(0;0;0;0)	0	(0;0;0;0)	(0;0;0)
		1	(1;1;0;0)	(1;1;1)
	(0;0;1;1)	0	(0;0;1;1)	(0;0;0)
		1	(1;1;1;1)	(1;1;1)
	(1;0;0;1)	0	(1;0;0;1)	(0;1;0)
		1	(0;1;0;1)	(1;0;1)
	(1;0;1;0)	0	(1;0;1;0)	(0;1;0)
		1	(0;1;1;0)	(1;0;1)
	(1;1;0;0)	0	(1;1;0;0)	(0;1;1)
		1	(0;0;0;0)	(1;0;0)
	(1;1;1;1)	0	(1;1;1;1)	(0;1;1)
		1	(0;0;1;1)	(1;0;0)
	(0;1;0;1)	0	(0;1;0;1)	(0;0;1)
		1	(1;0;0;1)	(1;1;0)
	(0;1;1;0)	0	(0;1;1;0)	(0;0;1)
		1	(1;0;1;0)	(1;1;0)
...				
6	(0;0;0;0)	0	(0;0)	(0;0;0)
		1	invalid	
	(1;1;0;0)	0	invalid	
		1	(0;0)	(1;0;0)
	(0;0;1;1)	0	invalid	
		1	(1;1)	(1;1;1)
	(1;1;1;1)	0	(1;1)	(0;1;1)
		1	invalid	
	(1;0;0;1)	0	(1;1)	(0;1;0)
		1	invalid	
	(0;1;0;1)	0	invalid	
		1	(1;1)	(1;1;0)
	(1;0;1;0)	0	invalid	
		1	(0;0)	(1;0;1)
	(0;1;1;0)	0	(0;0)	(0;0;1)
		1	invalid	
7	(0;0)	0	(0;0)	(0;0;0)
		1	(1;1)	(1;1;1)
	(1;1)	0	(1;1)	(0;1;1)
		1	(0;0)	(1;0;0)

states S_K^* satisfying these restrictions are valid, all others have to be discarded from the super-trellis. In the above example, if the component trellises were terminated, the only remaining legitimate value for S_8^* would be $(0;\ 0)$. The invalid transitions in the last super-trellis segment of Fig. 6 ($t = 7 \rightarrow 8$) are marked with dash–dotted lines.

From Table II and Fig. 6, we can clearly see that the turbo code super-trellis is time-variant, i.e., the structure of the trellis sections depends on the time instant t, exhibiting different legitimate transitions for different t values. More explicitly, the super-trellis is different for $t = 2 \rightarrow 3$ and $t = 3 \rightarrow 4$, while it is identical for $t = 2 \rightarrow 3$ and for $t = 4 \rightarrow 5$. For the time instants $t = 2, \ldots, 6$ we observe a periodicity in the super-trellis, manifesting itself in two different trellis sections, which are repeated alternately. This periodicity corresponds to the number of columns in the interleaver, which in our case was two. It is also easy to see that the constraint length of the super-trellis is three, while that of the component scramblers is two. Hence the memory introduced by the interleaver has increased the constraint length of the code. These issues will be augmented in more depth in Section VI.

Following the above simple example it may now be a worthwhile revisiting the generic super-trellis structure of Section III-C before proceeding further.

IV. COMPLEXITY OF THE TURBO CODE SUPER-TRELLIS

With the goal of estimating the associated complexity of the turbo code super-trellis, we will assume that the turbo encoder considered comprises two identical scramblers having a memory of M and an interleaver of length K.

A. Rectangular Interleavers

We will consider simple $\rho{\times}\chi$-rectangular interleavers, having ρ rows and χ columns. The data is written into the interleaver on a row-by-row basis and read out on a column-by-column basis. Upon using the previous definition of the time instant t (transition at time instant t in the super-trellis is due to the input symbol U_t of the turbo encoder FSM), the first component trellis is only extended to the right (scenario 1) in Section III-C) upon increasing t. In the second component trellis for every $t = 2, \ldots, \chi$ the *opening* (scenario 3) in Section III-C) of a new section will occur, respectively. As a result of this, in the second component trellis, a separate section exists for each of the χ columns of the interleaver. The section, which belongs to the first interleaver column (the leftmost section in the second component trellis) is associated with *one* interface state (at its right end), whereas the remaining $\chi - 1$ sections possess *two* interface states (left and right interface of the section, respectively). Therefore, together with the *single* interface state of the first component trellis the turbo code super state is a vector consisting of 2χ interface states of the component trellises. Each of the interface states can assume 2^M legitimate values. Hence the following statement holds for the complexity of the turbo code super-trellis.

Bound for Rectangular Interleavers: A turbo code, which is associated with a $\rho \times \chi$ rectangular interleaver, can be described by a super-trellis having a maximum of $2^{M \cdot 2\chi}$ states at any super-trellis stage.

Hence for our example turbo code incorporating the 4×2-rectangular interleaver and a scrambler of memory $M = 1$, the super-trellis can have a maximum of $2^{(1 \cdot 2 \cdot 2)} = 16$ legitimate states. However, as seen in Fig. 6, from the set of 16 possible states only eight are actually encountered. On the other hand, for a 2×4-rectangular interleaver, this upper bound is 256 super states, although it can be readily shown that this turbo code has an equivalent super-trellis representation occupying only eight of the 256 possible super states (cf. Example 2 and simply swap the first and the second component).

In order to eliminate this ambiguity in regard to the trellis complexity (one of the complexity upper bounds is associated with 16 states while the other with 256 states), we can redefine the time t in the super-trellis. In contrast to the previous situation, we have to distinguish clearly between the input symbols $U_{\mathrm{FSM},\,1}, \ldots, U_{\mathrm{FSM},\,K}$ of an *abstract* or hypothetical turbo encoder FSM and the input symbols $U_1, \ldots, U_K$ of the *real* turbo encoder. Let the input symbol $U_{\mathrm{FSM},\,t}$ of the turbo encoder FSM at the time instant t be the input symbol U_l of the turbo encoder, so that (over all time instants t) the maximum number of super states is minimized. This definition of t triggers a permutation of the input symbols U_l to $U_{\mathrm{FSM},\,t}$. From this definition it follows for a $\rho \times \chi$-rectangular interleaver, that the order of the input symbols $U_{\mathrm{FSM},\,t}$ of the turbo encoder FSM

- corresponds to the order of the input symbols of the turbo encoder for $\rho > \chi$, i.e., $U_{\mathrm{FSM},\,t} = U_t$ (there is one section in the first component trellis and χ sections in the second component trellis);
- obeys the order of symbols at the output of the rectangular interleaver in the turbo encoder for $\rho < \chi$, i.e., $U_{\mathrm{FSM},\,t} = U_{\phi_t}$, where $l = \phi_t$ is the inverse mapping of $t = \pi_l$. (There are ρ sections in the first component trellis and one section in the second component trellis, while the definition of the super states is analogous to that in Section III-C, where only the first and second component trellis have to be swapped);
- for $\rho = \chi$ the trellis complexity is minimal for both of the above mentioned orderings of the symbols.

In summary, when $\rho > \chi$, we can exchange the two decoders, as in Example 2, and the maximum number of super-trellis states can be formulated as $\leq 2^{2 \cdot M \cdot \min(\rho,\,\chi)}$.

B. Uniform Interleaver

Instead of considering a specific random interleaver, we will now derive an upper bound for the super-trellis complexity, averaged over all possible interleavers, a scenario, which is referred to as the so-called uniform interleaver [6] of length K.

Without loss of generality, we define the time t such that the transition in the super-trellis at time instant t is associated with the input symbol U_t. This implies that the order of the input symbols of the abstract or hypothetical turbo encoder FSM is the same as for the real turbo encoder, i.e., $U_{\mathrm{FSM},\,t} = U_t$. Hence, only one section exists in the first component trellis, which is extended to the right upon increasing t.

The specific input symbol of the second component scrambler—which belongs to the input symbol U_t and hence is input to the second scrambler simultaneously with U_t entering the first one—is $U^{(2)}_{\pi_t}$. Therefore, the corresponding transition in the second component trellis has the index π_t. Let us consider a state $S^{(2)}_l$ with arbitrary index $l = 1, \ldots, (K-1)$ in the second component trellis at time instant t. This state $S^{(2)}_l$ forms the interface at the right of a section in the second component trellis, which belongs to the pre-history wrt t, if and only if both of the following two conditions are satisfied.

1) A section is situated directly at the left of $S^{(2)}_l$. This implies that a transition, which already belongs to the pre-history wrt t is directly adjacent to the left of this state in the second component trellis. The input symbol $U^{(2)}_l$, which belongs to this transition must have been therefore already an input symbol of the turbo encoder FSM, hence the relation $\exists\, m \in \{1 \ldots t\}$, $l = \pi_m$ must hold. For an arbitrary l and when averaged over all possible interleavers $\boldsymbol{\pi}$, this condition is met with a probability of $p_1(t) = t/K$.

2) No section exists directly to the right of $S^{(2)}_l$, or correspondingly, that the transition with index $l+1$, which is directly adjacent at the right, does not belong to the pre-history of the super-trellis wrt t. Hence, the relation $l + 1 \neq \pi_m$, $\forall\, m \in \{1, \ldots, t\}$ must hold. Under the assumption that condition 1) is fulfilled, condition 2) will hold with a probability of $p_2(t) = (K-t)/(K-1)$.

For every state $S^{(2)}_l$, $l = 1, \ldots, K-1$ the probability that it represents the right interface to a section in the second component trellis at time instant t is, therefore, $p_{12}(t) = p_1(t) \cdot p_2(t) = [t \cdot (K-t)]/[K \cdot (K-1)]$. Condition 2) is cancelled for state $S^{(2)}_K$, since the end of the trellis is located directly to the right of it. If the second component trellis is terminated, then we have $S^{(2)}_K = 0$, and hence $S^{(2)}_K$ does not appear as an interface state.

An arbitrary state therefore constitutes the right-hand side (RHS) interface of a section in the second component trellis with a probability of

$$p_3(t) = (K-1)/K \cdot p_{12}(t) + 1/K \cdot p_1(t) = t \cdot (K+1-t)/K^2$$

at time instant t. Hence there are on average

$$n_r(t) = K \cdot p_3(t) = t \cdot (K+1-t)/K$$

RHS interfaces in the second component trellis. This number is maximized for $t_m = (K+1)/2$, for which there are $n_r(t_m) = (K+1)^2/(4K)$ RHS interfaces on average in the second component trellis. Similarly, one can deduct that there are $n_l(t) = n_r(t)$ left-hand side (LHS) interfaces on average in the second component trellis, when the edge effects at the start and end of the trellis are ignored.

Along with the single interface state $S^{(1)}_t$ in the first component trellis we obtain for the maximum total number of interfaces in both trellises under the assumption of a uniform interleaver, as defined above

$$n_{\max} = 1 + \frac{(K+1)^2}{2K} \tag{2}$$

and, therefore, the following bound accrues.

Bound for the Uniform Interleaver: The following upper bound holds for the number of super states in the minimal super-trellis of a turbo code, averaged over all possible interleavers of length K

$$\begin{aligned} E\left[\|\mathcal{S}_T(t)\|\right] &\leq 2^{M \cdot n_{\max}} \\ &= 2^{M + M \cdot (K+1)^2/(2K)}, \qquad \text{for } t = 1, \ldots, K. \end{aligned}$$

From (2) for large values of K, i.e., for large interleavers, we have $n_{\max} \approx K/2$, for which a maximum of $2^{M \cdot K/2}$ super states is expected in the super-trellis. For $M \leq 2$ this means that the trellis complexity of a turbo code with a random interleaver is on average only marginally lower than that of a random code [12] having 2^K codewords, which can be described by means of a trellis having a maximum of 2^K states.

V. Optimum Decoding of Turbo Codes

Having illuminated the super-trellis structure of turbo codes, we can now invoke this super-trellis, in order to optimally decode simple turbo codes. Let us commence by defining the task a decoder should carry out. Specifically, the goal of a decoder may be that of finding the codeword $\boldsymbol{c}_i$ with the highest probability of having been transmitted, upon reception of the sequence $\boldsymbol{y}$, which is formulated as identifying the index i

$$i = \arg\max_l \left(P(\boldsymbol{c}_l|\boldsymbol{y})\right) \tag{3}$$

where $\arg\max$ returns the index of the maximum and $P(\boldsymbol{c}_l|\boldsymbol{y})$ is the conditional probability that $\boldsymbol{c}_l$ is the transmitted codeword when $\boldsymbol{y}$ is received. These decoders are usually referred to as maximum *a posteriori* sequence estimation (MAPSE) schemes, since they estimate a complete codeword or the associated information word. By contrast, maximum *a posteriori* symbol-by-symbol estimation (MAPSSE) schemes [4] attempt to find the most probable values of all symbols contained in the information word or codeword.

It is straightforward to show that for memoryless additive white Gaussian noise (AWGN) channels, the decision rule of (3) for MAPSE decoding can be simplified to the following maximum-likelihood sequence estimation (MLSE) type decoding using Bayes' rule, if all codewords $\boldsymbol{c}_l$ appear with the same probability

$$i = \arg\min_l \left(\|\boldsymbol{c}_l - \boldsymbol{y}\|^2\right).$$

For MLSE decoding of a codeword transmitted over a memoryless AWGN channel, our decoding goal is equivalent to finding the valid codeword $\boldsymbol{c}_l$ that is closest to the received sequence $\boldsymbol{y}$ in terms of the Euclidean distance. We can thus introduce a Euclidian metric

$$\boldsymbol{M}_{\boldsymbol{c}_l} = \|\boldsymbol{c}_l - \boldsymbol{y}\|^2 \tag{4}$$

for each codeword, and having calculated the whole set of metrics, we opt for the codeword having the lowest metric, yielding

$$i = \arg\min_l \left(\boldsymbol{M}_{\boldsymbol{c}_l}\right). \tag{5}$$

When the code used can be described by a trellis, i.e., when the code symbols are generated by an FSM, the task of finding the minimum metric can be accomplished by using a dynamic programming method, such as the Viterbi algorithm. This algorithm reduces the number of metrics to be calculated by introducing metrics associated with paths in the trellis, which can be recursively updated, if the path is extended by one transition. The maximum number of paths/metrics, that has to be kept in memory, is $2 \cdot \|\mathcal{S}\|$, if $\mathcal{S}$ is the set of states in the trellis and each state transition is associated with a binary-input symbol. This can be achieved without ever discarding the optimum path in the sense of (5).

A. Comparison with Iterative Decoding

Since we have found a super-trellis representation for turbo codes, we can now invoke the appropriately modified Viterbi algorithm in order to MLSE decode the turbo code. In the Appendix, the optimality of this decoding approach is proven. Similarly, the algorithm [4] proposed by Bahl *et al.* could be used in order to MAPSSE decode the turbo code along its super-trellis, obtaining exact *a posteriori* probabilities for the information and code symbols. We have however implemented an MLSE decoder, since several other attempts have already been undertaken in order to MLSE decode turbo codes [13]–[15] and since the complexity of an MLSE decoder is considerably lower than that of an MAPSSE decoder.

In contrast to the aforementioned proposals for MLSE decoding of turbo codes, when the super-trellis is used, the super-trellis decoding imposes no restrictions on the information word length K of the code, only on the super-trellis complexity and therefore on the structure of the interleaver. Here we opted for rectangular interleavers having a low number of columns and compared the decoding BER results to those of an iterative "turbo" decoder.

All turbo codes in the following examples contain two identical component scramblers. The first component scrambler was terminated at the end of the information word, whereas the second component trellis was left open at its right end. Puncturing was applied, in order to obtain a turbo code of rate $1/2$. The code symbols were transmitted over an AWGN channel using Binary Phase Shift Keying (BPSK). E_b/N_0 represents the received energy per transmitted information bit E_b divided by the one-sided noise power spectral density N_0. The conventional, iterative "turbo" decoder benchmarker used the MAPSSE algorithm [4] for decoding the component codes.

First, we consider a turbo code with component scramblers of memory $M = 1$ and a two-column interleaver, as used above for Example 2. We examine a turbo code with a 499×2 rectangular interleaver, i.e., the information word length is $K = 998$. Fig. 7 shows the BER results of our simulations. We have used a Viterbi decoder for the "MLSE decoded" super-trellis as well as the "iterative" turbo decoder and plotted the BER results after the first and 16th iterations. We observe that for this example the code performance is quite low (also displayed is the BER for uncoded transmission with the same symbol energy $E_s = \frac{1}{2}E_b$ as for the coded transmission with rate $1/2$). There is virtually no difference between the MLSE decoder and the iterative one. Furthermore, we note that there is no BER improvement for more than one iteration.

All turbo codes used in the following simulations incorporate component scramblers of memory $M = 2$, which was chosen in order maintain a low complexity.

From Fig. 8 we can see that the BER differences between the noniterative MLSE and iterative decoding become clearer for three-column interleavers. We portrayed the simulation results for a 33×3 (i.e., $K = 99$) and a 333×3 ($K = 999$) interleaver, where the curves correspond to noniterative MLSE decoding and iterative decoding. The first and 16th iterations are shown. Here, there is a gain in the iterative algorithm, when performing more than one iteration. However, the iterative decoder cannot attain the BER performance of the noniterative MLSE decoder. The remaining SNR gap is about 0.5 dB at a BER of 10^{-3}. For lower BERs the associated curves seem to converge. The turbo codes have an identical number of 4096 super states in conjunction with both interleavers, which explains, why their coding performance is fairly similar. Since both interleavers have an identical number of columns, which determines the number of super states, their respective number of rows hardly affects the coding performance, although at low BERs the curve for the 33×3 interleaver diverges from that of the 333×3 interleaver. This is due to an edge effect at the end of the super-trellis (or at the end of the two component trellises), which shall not be discussed in more detail here and which causes an error floor. Fig. 9 shows the performance of the iterative algorithm for the turbo code with the 333×3 interleaver. Specifically, the BER is shown after one, two, four, eight, and 16 iterations, as is the BER curve for the noniterative MLSE decoder. The performance improvements of the iterative decoder appear to saturate after a few iterations, exhibiting a performance gap with respect to the noniterative MLSE decoder. In Fig. 10 we see that a clear gap also exists between the word error rate (WER) curves obtained for the noniterative MLSE and the iterative decoder after 16 iterations using the 33×3 interleaver.

Fig. 11 portrays the BER results for a turbo code incorporating a 39×5 rectangular interleaver, yielding $K = 195$. Observe that a gap of about 0.25 dB remains between the performance of the iterative and the noniterative MLSE decoder. Compared to Fig. 9, we note that the gap has narrowed upon increasing the number of columns in the interleaver from 3 to 5. Note, however, that only a low number of

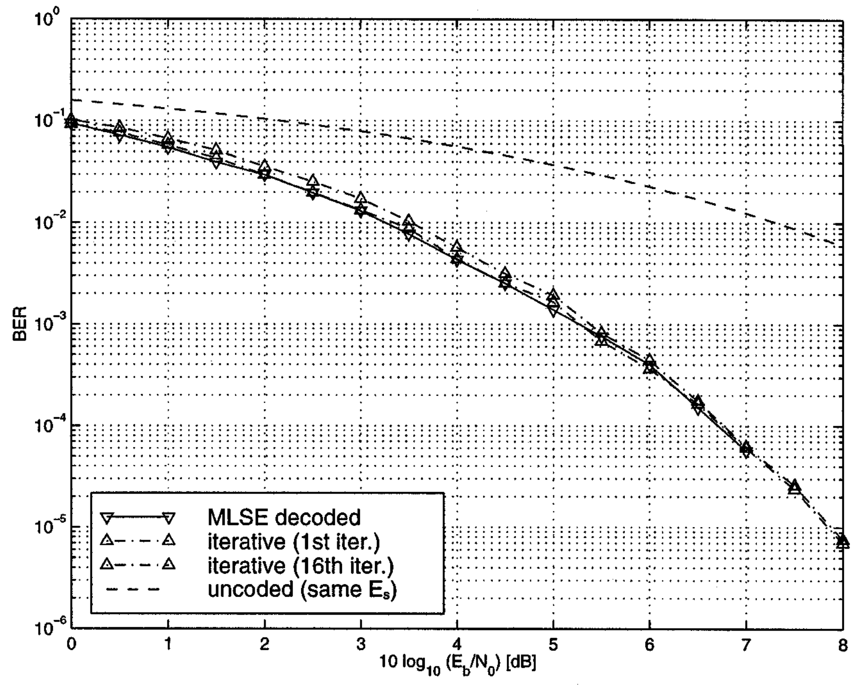


Fig. 7. Turbo coded BER performance for $M = 1,499 \times 2$ rectangular interleaver: comparison between optimum noniterative and iterative decoding.

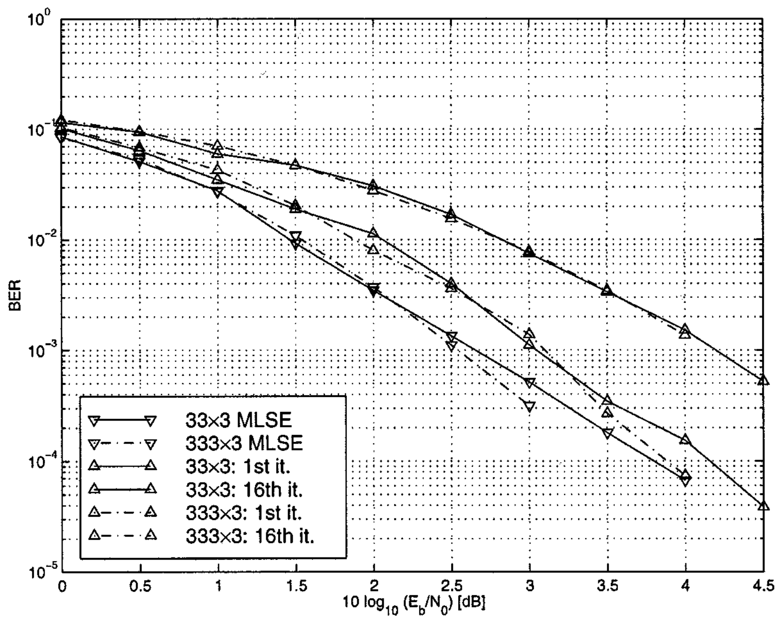


Fig. 8. Turbo coded BER performance for $M = 2, 33 \times 3$ and 333×3 rectangular interleavers: comparison between optimum noniterative and iterative decoding.

word errors were registered for each point in the BER curve of the optimum decoder. For example, only 1457 codewords were transmitted at an SNR of $10 \log_{10}(E_b/N_0) = 3$ (dB), of which 21 were in error after decoding. The reason for this low number of transmitted codewords is the complexity of the super-trellis for the turbo code, which possesses 2^{20} super states. For the iterative decoder the associated complexity is considerably lower. The component scramblers have four states each. The iterative decoder exhibits a complexity per iteration, which is about four times that of MLSE-decoding one of the component codes (one forward and one backward recursion for both component codes [4], [1]). The optimum noniterative decoder associated with the large super-trellis has therefore a complexity, which corresponds to about $2^{20}/(4 \cdot 4) = 2^{16}$ iterations in the conventional "turbo" decoder for the 39×5-interleaver.

An intermediate result is that the process of iterative decoding is suboptimum. In practice, the interleavers, which have been investigated so

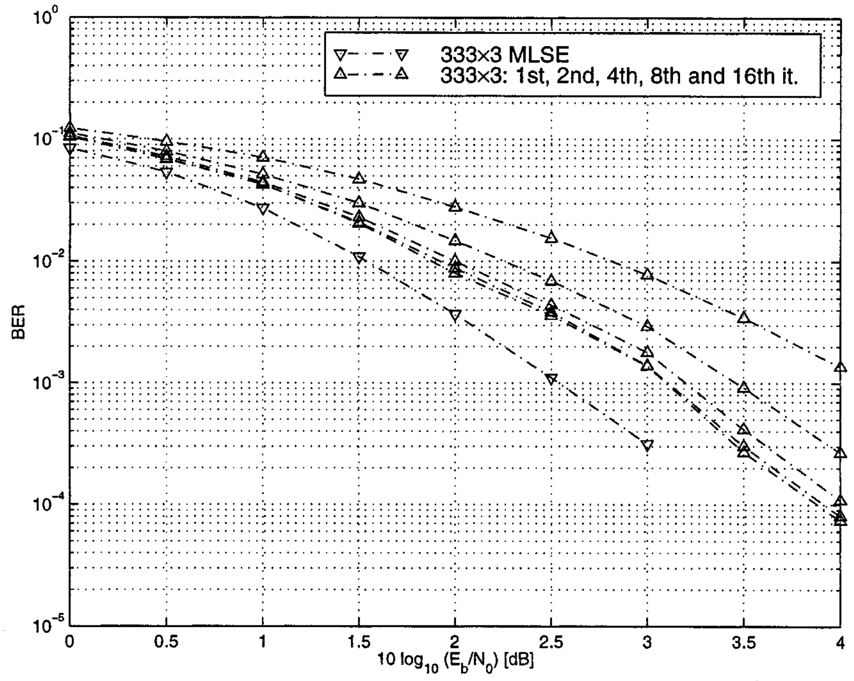


Fig. 9. Turbo coded BER performance for $M = 2, 333 \times 3$ rectangular interleavers: convergence behavior of the iterative algorithm compared to optimum noniterative decoding.

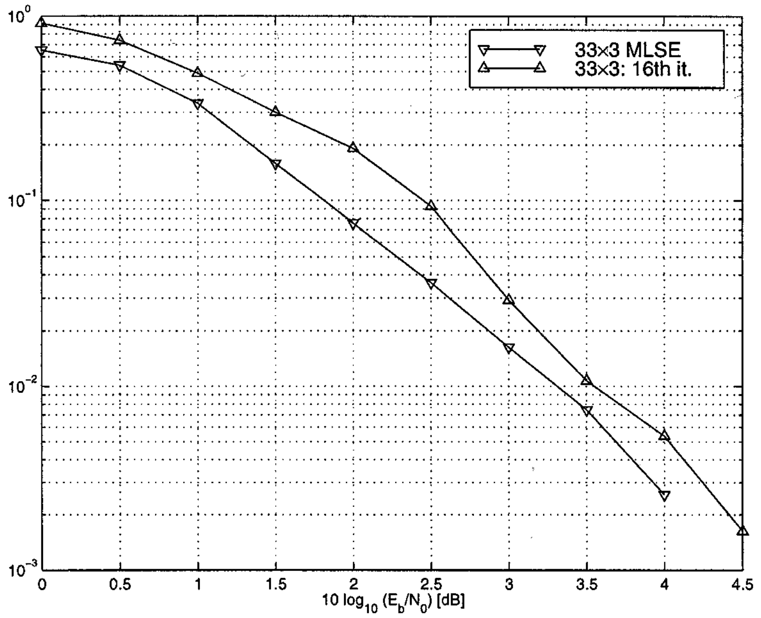


Fig. 10. Turbo coded word error rate (WER) performance for $M = 2, 33 \times 3$ rectangular interleaver: comparison between optimum noniterative and iterative decoding.

far, are not employed in practical turbo codes, since their performance is relatively low in comparison to other interleavers. For short information word lengths, such as $K \leq 200$, one normally employs rectangular interleavers [16], where the number of rows approximately equals the number of columns. For high information word lengths K one normally employs random interleavers, in order to achieve a good coding performance. The super-trellises, which belong to these schemes, are fairly complex, as it was detailed in Section IV, and hence they cannot be used for noniterative MLSE decoding in practical codecs.

B. Comparison with Conventional Convolutional Codes

In this subsection we do not wish to assess the performance of the iterative decoding algorithm in comparison to the optimum one. Instead,

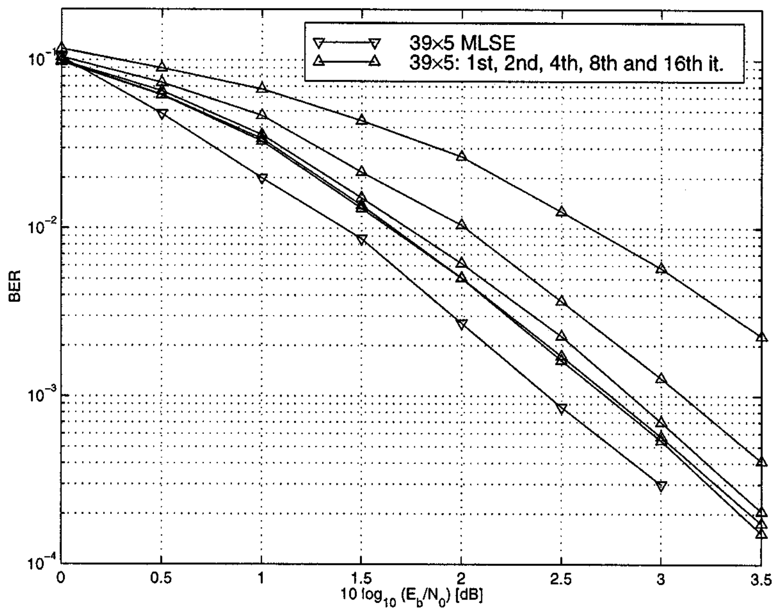


Fig. 11. Turbo coded BER performance for $M = 2$, 39×5 rectangular interleaver: comparison between optimum noniterative and iterative decoding.

we compare the performance of a turbo code to that of conventional convolutional codes, whose trellis exhibits the same complexity as that of the turbo code super-trellis.

First of all we consider a turbo code with component scramblers having a memory of $M = 1$ and a two-column interleaver, as in Example 2, but in this case with a 499×2 rectangular interleaver. As we have noted above in Example 2, the super-trellis is associated with a maximum of 16 super states, of which only eight super states are actually encountered in the super-trellis. Therefore, in Fig. 12 we compared the performance of the turbo code ("TC") having a 499×2 rectangular interleaver and invoking noniterative MLSE decoding of the super-trellis with that of a convolutional code ("C") having a memory of $M = 3$ (eight states, octal generator polynomials of 15_o and 17_o) and $M = 4$ (16 states, generator polynomials of 23_o and 35_o), respectively, which are also MLSE (Viterbi) decoded. We observe that both convolutional codes are more powerful, than the turbo code of the same decoder complexity, if the turbo code is optimally decoded.

Furthermore, in Fig. 13 we compared a turbo code having component scramblers of memory $M = 2$ and a 333×3 rectangular interleaver, which has a maximum of 4096 super states, with a convolutional code of the same trellis complexity. Specifically, the convolutional code's memory was $M = 12$ and the octal generator polynomials were 10533_o and 17661_o, which are optimal according to [17]. Similarly to the previous case, for medium and high SNRs, the performance of this convolutional code is significantly better than that of the optimally decoded turbo code of the same trellis complexity.

In order to summarize, for both cases we found that a conventional convolutional code is far more powerful than a noniterative MLSE-decoded turbo code of the same trellis complexity.

VI. Discussion of the Results

As mentioned earlier, when inspecting the super-trellis of Fig. 6 and Table II, the trellis periodicity with period 2—which is based on the periodicity of the 4×2 rectangular interleavers employed—is apparent. In other words, we stated before that the structure of the trellis segments depends on the time instant t, exhibiting different legitimate transitions for different t values. More explicitly, the trellis was different for $t = 2 \rightarrow 3$ and $t = 3 \rightarrow 4$, while it was identical for $t = 2 \rightarrow 3$ and for $t = 4 \rightarrow 5$. We found that $\rho \times \chi$ rectangular interleavers exhibit a periodicity in the super-trellis (if edge effects are ignored), which has a period of $\min(\rho, \chi)$. By rearranging the interface states of the super state vector one can eliminate the time-variant nature of the trellis and hence a time-invariant super-trellis can be generated, which is identical at all values of t. In the context of Example 2, we could relabel the super state vectors as follows:

$$\tilde{S}_2^* = \left(S_2^{(1)};\ S_5^{(2)};\ S_4^{(2)};\ S_1^{(2)}\right)$$
$$\tilde{S}_3^* = \left(S_3^{(1)};\ S_2^{(2)};\ S_4^{(2)};\ S_5^{(2)}\right)$$
$$\tilde{S}_4^* = \left(S_4^{(1)};\ S_6^{(2)};\ S_4^{(2)};\ S_2^{(2)}\right)$$
$$\tilde{S}_5^* = \left(S_5^{(1)};\ S_3^{(2)};\ S_4^{(2)};\ S_6^{(2)}\right)$$
$$\tilde{S}_6^* = \left(S_6^{(1)};\ S_7^{(2)};\ S_4^{(2)};\ S_3^{(2)}\right)$$

in order to ensure that the two interface states, that have replaced two previous interface states for generating super state $\tilde{S}_{t+1}^*$ from $\tilde{S}_t^*$ (see Subsection III-C, scenario 1)), are always placed as the first two vector elements, whereas the remaining interface states are the last two elements of the super state vector. Table III shows the corresponding super state transitions for these relabeled super states. Note that relabeling of the super states does not affect the output vector $\overline{C}_t$ associated with a (relabeled) super state transition, which thus remains the same. Fig. 14 shows a part of the super-trellis for Example 2, which has been rendered time-invariant by the above relabeling of the super states. It is readily inferred that the constraint length for this example was 2 for the component scramblers, which increased to 3 for the super-trellis. The minimum free Hamming distance for this rate $1/3$ code is $\delta_{\text{free}} = 5$, if we take into consideration only paths in the time-invariant part of the super-trellis (i.e., ignoring edge effects at the super-trellis start and end). For comparison, the best conventional convolutional code with eight states (constraint length 4) and rate $1/3$ has $\delta_{\text{free}} = 10$ [17], which explains the convolutional code's superior BER performance.

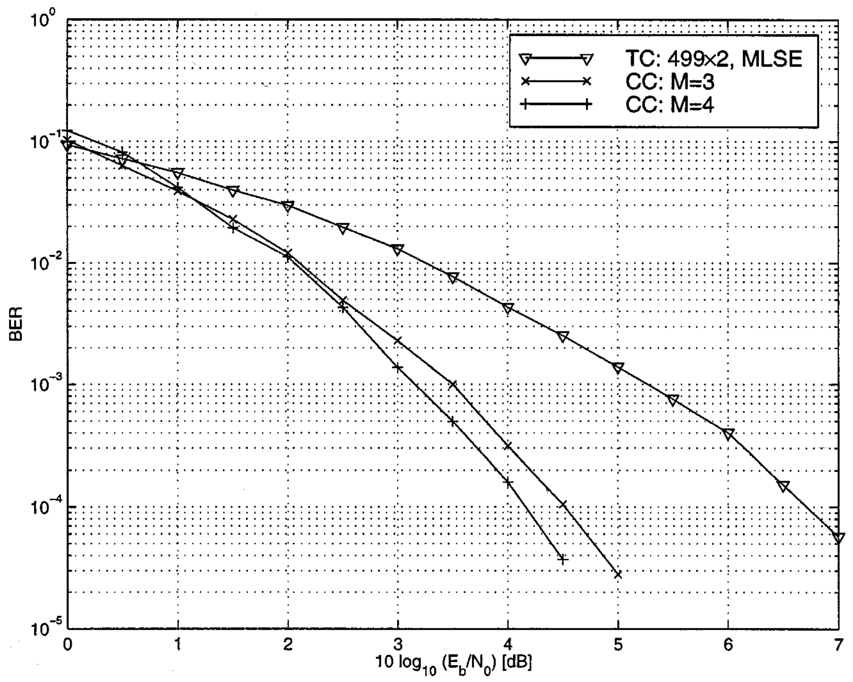


Fig. 12. Turbo coded BER performance with a maximum of 16 super states, of which eight are reached, compared to convolutional codes with 16 and eight states.

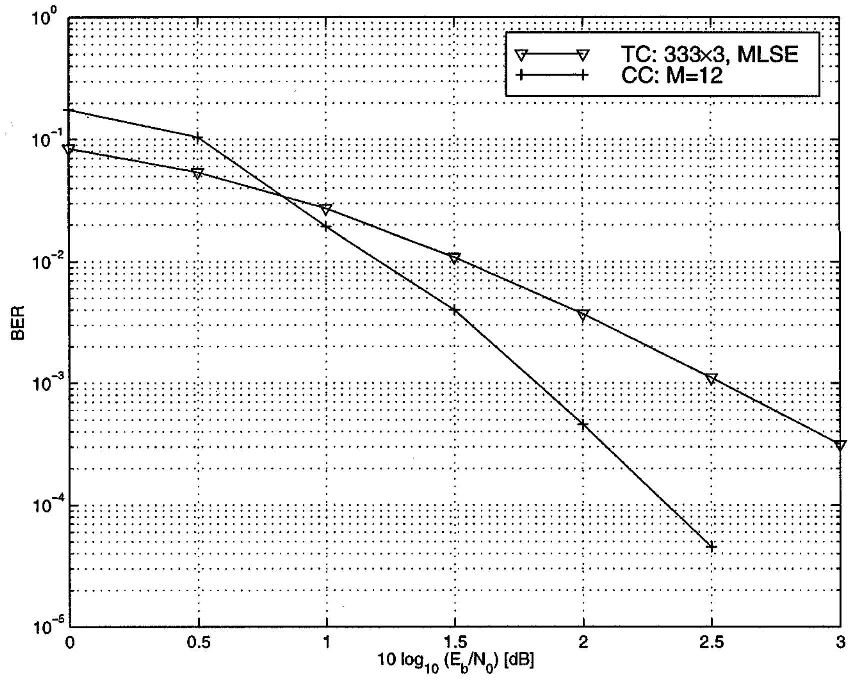


Fig. 13. Turbo coded BER performance with 4096 super states compared to convolutional code with 4096 states.

In general, the time-invariant super-trellis of a turbo code having a $\rho \times \chi$ rectangular interleaver, which has been obtained by relabeling the super states, is reminiscent of the trellis of a conventional block-based or zero-terminated convolutional code. In general, the constraint length of the turbo code super-trellis is $1 + m * M$ for component scramblers having a memory of M, where $m = \min(\rho, \chi)$, which is typically higher than the constraint length of a conventional convolutional code, if m is sufficiently high. The turbo code super-trellis exhibits $\leq 2^{M \cdot 2m}$ super states. Unfortunately—as underlined by our simulation results—its BER performance is inferior to that of a good conventional convolutional code having the same number of states.

TABLE III
State Transitions and Associated Output After Rearranging the Interface States in the Super-State Vector for Example 2

$\tilde{S}_t^*$	U_{t+1}	$\tilde{S}_{t+1}^*$	$\tilde{C}_{t+1}$
(0;0;0;0)	0	(0;0;0;0)	(0;0;0)
	1	(1;1;0;0)	(1;1;1)
(0;0;1;1)	0	(0;1;1;0)	(0;0;1)
	1	(1;0;1;0)	(1;1;0)
(0;1;0;1)	0	(0;1;0;1)	(0;0;1)
	1	(1;0;0;1)	(1;1;0)
(0;1;1;0)	0	(0;0;1;1)	(0;0;0)
	1	(1;1;1;1)	(1;1;1)
(1;0;0;1)	0	(1;1;0;0)	(0;1;1)
	1	(0;0;0;0)	(1;0;0)
(1;0;1;0)	0	(1;0;1;0)	(0;1;0)
	1	(0;1;1;0)	(1;0;1)
(1;1;0;0)	0	(1;0;0;1)	(0;1;0)
	1	(0;1;0;1)	(1;0;1)
(1;1;1;1)	0	(1;1;1;1)	(0;1;1)
	1	(0;0;1;1)	(1;0;0)

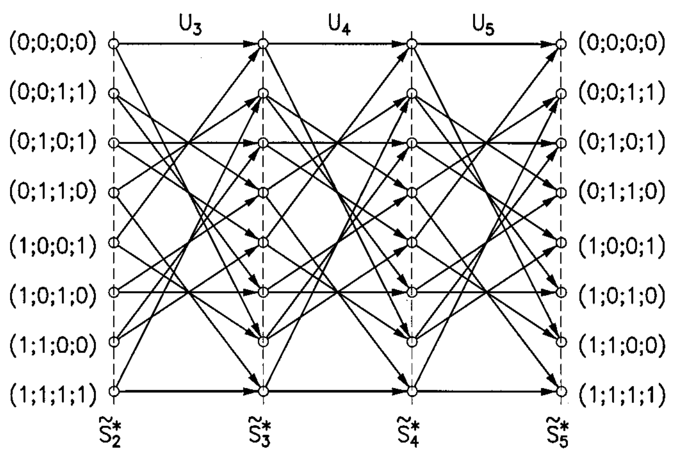


Fig. 14. Three segments of the new time-invariant super-trellis of Example 2 obtained by relabeling the super states.

Although turbo codes, which exhibit a complex super-trellis, cannot be noniterative MLSE—decoded for complexity reason, they can be subjected to iterative decoding. Indeed, the more complex the super-trellis, the closer the performance of the iterative decoding seems to be to that of the noniterative MLSE decoding. The impact of the interleaver is that it results in a high number of super states of the turbo code. Hence the statistical dependencies between the *a priori* probabilities input to one of the component decoders, which stem from the extrinsic probabilities of the other component decoder, are reduced. This is also the reason that throughout our simulations the iterative algorithm was unable to reach the performance of the noniterative MLSE decoding, while according to [6] this is possible on average across the range of all feasible interleavers, although nearly all of these interleavers result in a relatively high super-trellis complexity. Another example of convolutional codes having a high constraint length, which is decodable in an iterative fashion, is represented by the so-called low-density parity-check codes [18]. Conventional turbo codes have a high number of super states in their super-trellis and hence they are superior to conventional convolutional codes having a low number of states. The advantage of turbo codes is that they are amenable to low-complexity iterative decoding, although their super-trellis structure is not optimal in terms of maximizing the minimum Hamming distance $\delta_{\min}$. The low minimum Hamming distance of turbo codes is also reflected by the so-called *error-floor* of the BER curve [6], [8]. In general, the code performance is only determined at medium to high SNRs by $\delta_{\min}$ of the code. At low SNR, where turbo codes show a high power efficiency, the convenient shaping of the distance profile of turbo codes employing a random interleaver, which has been described in [7], is more important than achieving a high $\delta_{\min}$. The complex super-trellis of the turbo code does not endeavor to optimize $\delta_{\min}$, but instead it is responsible for the attractive shaping of the distance profile. Specifically, although there exist low-weight paths in the super-trellis, nonetheless, in comparison to other types of channel codes, only a low number of these paths exists.

If we consider, for example, a conventional nonrecursive convolutional code with a free distance of δ_{free}, a single "1" in the information sequence at the input of the encoder results in a nonzero weight path, which is associated with a low output weight δ_1 with $\delta_{\text{free}} \leq \delta_1 \leq \lceil L/R \rceil$, where L is the constraint length, R is the rate of the code, and $\lceil \bullet \rceil$ denotes the upwards rounding (ceiling) function. Accordingly, one can give $l \cdot \delta_1$ as the upper bound of the associated codeword weight for every information word of length K of a block-based or zero-terminated convolutional code, which contains l binary "1" symbols. In case of a small l value there are at least $\binom{K}{l}$ codewords of a block-based or zero-terminated conventional convolutional code, which produce a low output weight of $w \leq l \cdot \delta_1$. For the class of recursive convolutional codes this statement holds as well, since for every recursive convolutional code, there is a nonrecursive code, that has the same set of codewords and only differs in the mapping from information bits to codewords. The same statement can be derived for turbo codes with a *rectangular* interleaver, which is due to the fact, that a time-invariant super-trellis exists in this case, which exhibits a different trellis structure at different instants t. In a turbo code incorporating a *random* interleaver, however, there are much fewer codewords of a low weight [8], [6]. In contrast to rectangular interleavers, in conjunction with the random interleavers the number of low-weight paths does not increase with K, which is the reason for the interleaver gain [6], [8].

VII. Conclusion

We have shown that turbo codes can be described by means of a super-trellis, if the trellises of the component scramblers and the interleaver structure are known. For rectangular interleavers one can model this super-trellis as time-invariant, so that it resembles the trellis of a conventional convolutional code. We have also argued that a "good" conventional convolutional code of the same trellis complexity can be more powerful than a turbo code. On the basis of simulations, we have found that iterative decoding is suboptimum, at least for the investigated simple rectangular interleavers having a low number of columns. We have presented an upper bound for the super-trellis complexity of turbo codes based on rectangular interleavers and an upper bound for the super-trellis complexity averaged over all possible interleavers. The second bound gives rise to the supposition that the complexity of a turbo code having a random interleaver is of the same magnitude as that of a random code.

APPENDIX
PROOF OF ALGORITHMIC OPTIMALITY

In order to show that using a Viterbi algorithm along the super-trellis is optimum in the sense of noniterative MLSE decoding of turbo codes, we have to demonstrate that when paths are discarded in the super-trellis (in the following referred to as *super paths*) during the course of the Viterbi algorithm, the specific super path associated with the optimum codeword $\boldsymbol{c}_{\rm opt}$ in the sense of (5) (for BPSK transmission over an AWGN channel) is never discarded.

Let $\mathcal{M}_t$ be the set of codewords associated with the nondiscarded super paths at decoding stage t (corresponds to time instant t in the encoder), which implies at stage $t = 0$ that $\mathcal{M}_0 = \{$all valid codewords $\boldsymbol{c}_i\}$. Let us now invoke the method of induction below.

We Claim That: $\boldsymbol{c}_{\rm opt} \in \mathcal{M}_t \forall t$

Proof:

Commence Induction From: $\boldsymbol{c}_{\rm opt} \in \mathcal{M}_0$.

Inductive Assumption: Let us assume that for an arbitrary t the following is true: $\boldsymbol{c}_{\rm opt} \in \mathcal{M}_t$.

Induction Conclusion: When proceeding from the decoding stage t to the following stage $t+1$ by extending the surviving super paths and then discarding all but one path merging into any super state, the super path associated with the optimum codeword $\boldsymbol{c}_{\rm opt}$ is not discarded, i.e., $\boldsymbol{c}_{\rm opt} \in \mathcal{M}_{t+1}$, as shown below.

From the mapping $\gamma_{\rm s}$ of the component scrambler FSM $\mathbb{F}_{\rm s}$ we see that the output symbol $c_k^{(i)}$ of the ith component scrambler at any transition index k is a function f_1 of the encoder state $s_{k-1}^{(i)}$ before this transition and the current input symbol $u_k^{(i)}$, which can be expressed as

$$c_k^{(i)} = f_1\left(s_{k-1}^{(i)};\, u_k^{(i)}\right). \tag{6}$$

Similarly to the output symbol $c_k^{(i)}$, the new encoder state $s_k^{(i)}$ is also a function of the previous state $s_{k-1}^{(i)}$ and the input symbol $u_k^{(i)}$, as shown below

$$s_k^{(i)} = g_1\left(s_{k-1}^{(i)};\, u_k^{(i)}\right). \tag{7}$$

A section $(c_{k+1}^{(i)};\, \ldots;\, c_l^{(i)})$ of a component codeword is therefore only dependent on the encoder state $s_k^{(i)}$ at encoding stage k and the input symbols $(u_{k+1}^{(i)};\, \ldots;\, u_l^{(i)})$

$$\left(c_{k+1}^{(i)};\, \ldots;\, c_l^{(i)}\right) = \boldsymbol{f}_2\left(s_k^{(i)};\, u_{k+1}^{(i)};\, \ldots;\, u_l^{(i)}\right). \tag{8}$$

We now consider decoding stage $t+1$ of the Viterbi algorithm for the super-trellis. For the sake of simplicity, we omit the time index $t+1$ in the variables. Let J be the total number of sections in the two component trellises belonging to the pre-history of the super-trellis as regards to the current time $t+1$ entailing the sections in the component trellises, which have already been explored by the decoder at decoding stage $t+1$. Let N be the number of trellis sections belonging to the post-history as regards to $t+1$, which entails the so-far unexplored sections. Every turbo codeword $\boldsymbol{c}$ can be split into subvectors of two types, namely, into J subvectors $\boldsymbol{c}_{{\rm pre},\,j}$, $j = 1, \ldots, J$ of code symbols, which are associated with the J component trellis sections belonging to the pre-history as regards to $t+1$, and in

$$\boldsymbol{c}_{{\rm post},\,n} = \left(c_{k_n+1}^{(i_n)};\, \ldots;\, c_{l_n}^{(i_n)}\right), \qquad n = 1, \ldots, N$$

which are associated with the N sections belonging to the post-history.

Let $\boldsymbol{c}_{\rm pre}$ be a vector that contains all code symbols belonging to the pre-history

$$\boldsymbol{c}_{\rm pre} = (\boldsymbol{c}_{{\rm pre},\,1};\, \ldots;\, \boldsymbol{c}_{{\rm pre},\,J})$$

and let $\boldsymbol{c}_{\rm post}$ represent all the code symbols belonging to the post-history at decoding stage $t+1$

$$\boldsymbol{c}_{\rm post} = (\boldsymbol{c}_{{\rm post},\,1};\, \ldots;\, \boldsymbol{c}_{{\rm post},\,N}).$$

Any codeword $\boldsymbol{c}$ is hence constituted by these two vectors. Explicitly, $\boldsymbol{c}$ is a specific permutation Π of a concatenation of these two vectors, where the actual nature of the permutation is irrelevant in this context, leading to the following formulation:

$$\boldsymbol{c} = \Pi\,(\boldsymbol{c}_{\rm pre};\, \boldsymbol{c}_{\rm post}). \tag{9}$$

According to (8), for the post-history $\boldsymbol{c}_{\rm post}$ of any codeword $\boldsymbol{c}$, its N constituent parts $\boldsymbol{c}_{{\rm post},\,n}$ depend only on the encoder state $s_{k_n}^{(i_n)}$ at the beginning of the specific section n, $n = 1, \ldots, N$ in the corresponding component trellis and on the component scrambler input $(u_{k_n+1}^{(i_n)};\, \ldots;\, u_{l_n}^{(i_n)})$ associated with the transitions inside this section. Hence we obtain the following dependence:

$$\boldsymbol{c}_{\rm post} = \boldsymbol{f}_3\left(s_{k_1}^{(i_1)};\, s_{k_2}^{(i_2)};\, \ldots;\, s_{k_N}^{(i_N)};\, \boldsymbol{u}_{\rm post}\right) \tag{10}$$

where

$$\boldsymbol{u}_{\rm post} = \left(u_{k_1+1}^{(i_1)};\, \ldots;\, u_{l_1}^{(i_1)};\, u_{k_2+1}^{(i_2)};\, \ldots;\, u_{l_N}^{(i_N)}\right)$$

represents the information symbols associated with the post-history of this codeword $\boldsymbol{c}$, and where $\boldsymbol{f}_3$ expresses a functional dependence, which does not have to be further specified here. Furthermore, from (7), we obtain a second dependence for the states at the end of the sections

$$\left(s_{l_1}^{(i_1)};\, s_{l_2}^{(i_2)};\, \ldots;\, s_{l_N}^{(i_N)}\right) = \boldsymbol{g}_2\left(s_{k_1}^{(i_1)};\, s_{k_2}^{(i_2)};\, \ldots;\, s_{k_N}^{(i_N)};\, \boldsymbol{u}_{\rm post}\right). \tag{11}$$

Again, $\boldsymbol{g}_2$ represents a functional dependence, whose actual nature is irrelevant in this context.

The encoder states $s_{l_j}^{(i_j)}$, $j = 1, \ldots, J$ and $s_{k_n}^{(i_n)}$, $n = 1, \ldots, N$ constitute the super state s^*, that the super path associated with the codeword $\boldsymbol{c}$ is merging into at time instant $t+1$, as discussed in Section III-C. The codeword symbols belonging to the post-history can therefore be expressed by

$$\boldsymbol{c}_{\rm post} = \boldsymbol{f}_4\,(s^*;\, \boldsymbol{u}_{\rm post}) \tag{12}$$

and this vector $\boldsymbol{c}_{\rm post}$ exists, if and only if $(s^*;\, \boldsymbol{u}_{\rm post})$ is a valid pair according to the condition of (11).

When two super paths merge into a super state s^* at decoding stage $t+1$, then the interface states in all the component trellis sections constituting the pre-history as regards to $t+1$ are identical for the two super paths, which is clear from the definition of the super states in Section III-C.

Suppose that the optimum codeword $\boldsymbol{c}_{\rm opt}$ is associated with the super path $\overset{>}{p}_{\rm opt}$ (the $>$ identifies it as a path) merging into the super state $s_{\rm opt}^*$ at decoding stage $t+1$. Since we have imposed that $\boldsymbol{c}_{\rm opt} \in \mathcal{M}_t$, this super path has not been discarded in earlier decoding stages and is present in the Viterbi decoder at the current stage $t+1$ before the discarding process commences.

Let $\boldsymbol{c}_{\overset{>}{p}_{\rm opt}}$ denote the code symbols belonging to the super path $\overset{>}{p}_{\rm opt}$ (i.e., the output of the turbo encoder FSM associated with this super path of length $t+1$). Thus $\boldsymbol{c}_{\overset{>}{p}_{\rm opt}}$ is identical to $\boldsymbol{c}_{{\rm opt},\,{\rm pre}}$ (pre-history part of $\boldsymbol{c}_{\rm opt}$), and $\boldsymbol{c}_{{\rm opt},\,{\rm post}}$ represents the remaining code symbols of $\boldsymbol{c}_{\rm opt}$. We denote the information symbols associated with the post-history of $\boldsymbol{c}_{\rm opt}$ by $\boldsymbol{u}_{{\rm opt},\,{\rm post}}$. Then we see from (12) that

$$\boldsymbol{c}_{{\rm opt},\,{\rm post}} = \boldsymbol{f}_4\left(s_{\rm opt}^*;\, \boldsymbol{u}_{{\rm opt},\,{\rm post}}\right) \tag{13}$$

and from (9) we have

$$\boldsymbol{c}_{\rm opt} = \Pi\left(\boldsymbol{c}_{\overset{>}{p}_{\rm opt}};\, \boldsymbol{c}_{\rm opt,\,post}\right). \tag{14}$$

Let $\boldsymbol{c}_{\overset{>}{p}_2}$ represent the code symbols associated with another super path $\overset{>}{p}_2$ merging into $s^*_{\rm opt}$ at time instant $t+1$. Then for *every* valid $\overset{>}{p}_2$, we can construct a valid codeword

$$\boldsymbol{c}_2 = \Pi\left(\boldsymbol{c}_{\overset{>}{p}_2};\, \boldsymbol{c}_{\rm opt,\,post}\right) \tag{15}$$

where $\boldsymbol{c}_{\rm opt,\,post}$ is given in (13).

If in the Viterbi algorithm, $\boldsymbol{M}_{\overset{>}{p}_{\rm opt}}$ is the Euclidian metric (cf. (4)) in the pre-history part (i.e., the metric of the code symbols $\boldsymbol{c}_{\overset{>}{p}_{\rm opt}}$ associated with super path $\overset{>}{p}_{\rm opt}$) and $\boldsymbol{M}_{\rm opt,\,post}$ is the metric associated with the code symbols $\boldsymbol{c}_{\rm opt,\,post}$ in the post-history part of the codeword $\boldsymbol{c}_{\rm opt}$, then the Euclidian metric of $\boldsymbol{c}_{\rm opt}$ is given by

$$\boldsymbol{M}_{\boldsymbol{c}_{\rm opt}} = \boldsymbol{M}_{\overset{>}{p}_{\rm opt}} + \boldsymbol{M}_{\rm opt,\,post} \tag{16}$$

which is the minimum of all codeword metrics $\boldsymbol{M}_{\boldsymbol{c}_i}$.

For the metric $\boldsymbol{M}_{\boldsymbol{c}_2}$ of the codeword $\boldsymbol{c}_2$ this means that

$$\boldsymbol{M}_{\boldsymbol{c}_{\rm opt}} = \boldsymbol{M}_{\overset{>}{p}_{\rm opt}} + \boldsymbol{M}_{\rm opt,\,post} \le \boldsymbol{M}_{\boldsymbol{c}_2} = \boldsymbol{M}_{\overset{>}{p}_2} + \boldsymbol{M}_{\rm opt,\,post} \tag{17}$$

and, consequently, for *all* valid $\overset{>}{p}_2$ merging into $s^*_{\rm opt}$

$$\boldsymbol{M}_{\overset{>}{p}_{\rm opt}} \le \boldsymbol{M}_{\overset{>}{p}_2}. \tag{18}$$

In the discarding process of decoding stage $t+1$, any super path $\overset{>}{p}_2$ leading to the super state $s^*_{\rm opt}$ is discarded because of its higher metric $\boldsymbol{M}_{\overset{>}{p}_2}$ and the optimum super path $\overset{>}{p}_{\rm opt}$ leading to this super state is retained, i.e., $\boldsymbol{c}_{\rm opt} \in M_{t+1}$. □

Acknowledgment

The authors wish to thank the anonymous reviewers as well as J. Hamorsky, L. Piazzo, and A. Knickenberg for proofreading the manuscript.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *Int. Conf. Communications*, 1993, pp. 1064–1070.
[2] P. Robertson, "Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes," in *Global Conf. Commications*, 1994, pp. 1298–1303.
[3] R. Gallager, "Low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. IT-8, pp. 21–28, Jan. 1962.
[4] L. Bahl *et al.*, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284–287, Mar. 1974.
[5] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 429–437, Mar. 1996.
[6] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 409–428, Mar. 1996.
[7] G. Battail, "On random-like codes," in *Canadian Workshop on Information Theory*, 1995.
[8] L. Perez, J. Seghers, and D. Costello, "A distance spectrum interpretation of turbo codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1698–1709, Nov. 1996.
[9] A. Khandani, "Design of turbo-code interleaver using hungarian method," *Electron. Lett.*, pp. 63–65, 1998.
[10] K. Andrews, C. Heegard, and D. Kozen, "Interleaver design methods for turbo codes," in *IEEE Int. Symp. Information Theory*, Cambridge, MA, 1998, p. 420.
[11] A. Ambroze, G. Wade, and M. Tomlinson, "Turbo code tree and code performance," *Electron. Lett.*, pp. 353–354, 1998.
[12] C. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, pp. 379–427, 1948.
[13] J. Sadowsky, "A maximum-likelihood decoding algorithm for turbo codes," in *IEEE Communications Theory Workshop*, Tucson, AZ, 1997.
[14] P. Höher, "New iterative ("turbo") decoding algorithms," in *Int. Symp. Turbo Codes*, Brest, France, 1997, pp. 63–70.
[15] C. Berrou, "Some clinical aspects of turbo codes," in *Int. Symp. Turbo Codes*, Brest, France, 1997, pp. 26–31.
[16] P. Jung and M. Naßhan, "Dependence of the error performance of turbo-codes on the interleaver structure in short frame transmission systems," *Electron. Lett.*, pp. 287–288, 1994.
[17] J. Proakis, *Digital Communications*, 2nd ed. New York: McGraw-Hill, 1989.
[18] A. Jimenez and K. Zigangirov, "Time-varying periodical convolutional codes with low-density parity-check matrix," *IEEE Trans. Inform. Theory*, vol. 45, pp. 2181–2191, Sept. 1999.

The Weight Hierarchies of Some Product Codes

Jeng Yune Park

***Abstract*—We determine the weight hierarchies of the product of an n-tuple space and an arbitrary code, the product of an m-dimensional even-weight code and the $[24,\,12,\,8]$ extended Golay code, and the product of an m-dimensional even-weight code and the $[8,\,4,\,4]$ extended Hamming code. The conjecture $d_r = d_r^*$ is proven for all three cases.**

***Index Terms*—Generalized Hamming weight, product codes, weight hierarchy.**

I. Introduction

Throughout this correspondence, all codes considered will be binary linear codes. We write $D \leqslant C$ when D is a subcode of C.

The *support* $\chi(D)$ *of a subset* $D \subseteq \mathbb{F}_2^n$ is the set of positions where at least one vector in D is nonzero. We remark that if D is a linear code then $\chi(D) = \chi(\text{basis of } D)$. The rth *generalized Hamming weight* (GHW) of an $[n,\,k,\,d]$ code C is defined as

$$d_r = d_r(C) = \min\{|\chi(D)|\,|D \leqslant C \text{ and } \dim(D) = r\}.$$

The *weight hierarchy* of C is the set of GHWs $\{d_0,\, d_1,\, d_2,\, \ldots,\, d_k\}$.

An $[n,\,k]$ code C is said to satisfy the *chain condition* provided that, for each $r (1 \leqslant r \leqslant k)$, there exists an r-dimensional subcode D_r of C such that

$$d_r(C) = |\chi(D_r)| \text{ and } D_1 < D_2 < D_3 < \cdots < D_k = C.$$

Manuscript received September 28, 1999; revised March 16, 2000.

The author is with the Department of Mathematics, University of Wisconsin–Madison, Madison, WI 53706 USA (e-mail: park@math.wisc.edu).

Communicated by I. F. Blake, Associate Editor for Coding Theory.

Publisher Item Identifier S 0018-9448(00)07008-5.

Wei and Yang [2] introduced the notation d_r^* for a product code $C_1 \otimes C_2$, which is defined in terms of the GHWs of C_1 and C_2. (See Section II-B.) They conjectured that $d_r = d_r^*$ if both C_1 and C_2 satisfy the chain condition. In this correspondence, we prove the conjecture for two classes of product codes: the product of an m-dimensional even-weight code and the $[24, 12, 8]$ extended Golay code, and the product of an m-dimensional code and the $[8, 4, 4]$ extended Hamming code. At the same time, we determine the weight hierarchies of these two product codes. We also show that $d_r(\mathbb{F}_2^n \otimes C) = d_r^*$ for an arbitrary code C.

II. Preliminaries

A. Product Codes

Definitions 1: The *product code* $C_1 \otimes C_2$ of an $[n_1, k_1]$ code C_1 and an $[n_2, k_2]$ code C_2 is defined by

$$C_1 \otimes C_2 = \left\{ \boldsymbol{c} = (c_{ij})_{\substack{1\leqslant i\leqslant n_1\\ 1\leqslant j\leqslant n_2}} \,\middle|\, \begin{array}{l} (c_{ij})_{1\leqslant i\leqslant n_1} \in C_1 \,\forall j \\ (c_{ij})_{1\leqslant j\leqslant n_2} \in C_2 \,\forall i \end{array} \right\}$$

that is, the set of $n_1 \times n_2$ arrays whose columns belong to C_1 and rows to C_2. The *product* $\boldsymbol{x} \otimes \boldsymbol{y}$ of two vectors $\boldsymbol{x} = (x_1, x_2, \ldots, x_{n_1})$ and $\boldsymbol{y} = (y_1, y_2, \ldots, y_{n_2})$ is the $n_1 \times n_2$ matrix $\boldsymbol{x}^T \boldsymbol{y}$ whose (i, j)-entry is $x_i y_j$.

The pair (i, j) is in $\chi(C_1 \otimes C_2)$ if there exists a codeword $\boldsymbol{c} \in C_1 \otimes C_2$ with $c_{ij} \neq 0$. The following lemma is well known.

Lemma 2: The product $C_1 \otimes C_2$ of an $[n_1, k_1, d^1]$ code C_1 and an $[n_2, k_2, d^2]$ code C_2 has parameters $[n_1 n_2, k_1 k_2, d^1 d^2]$. Furthermore, if $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{k_1}\}$ and $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{k_2}\}$ are bases for C_1 and C_2, respectively, $\{\boldsymbol{x}_i \otimes \boldsymbol{y}_j | 1 \leqslant i \leqslant k_1, 1 \leqslant j \leqslant k_2\}$ is a basis for $C_1 \otimes C_2$.

B. $d_r^*(C_1 \otimes C_2)$

Throughout this section, let C_1, C_2 be $[n_1, k_1]$, $[n_2, k_2]$ codes with the weight hierarchies $\{d_1^1, d_2^1, \ldots, d_{k_1}^1\}$ and $\{d_1^2, d_2^2, \ldots, d_{k_2}^2\}$, respectively.

Helleseth and Kløve [4] refined the definition of d_r^* using the concept of partition. In the following, we use the definition and notations established in [4].

Definitions 3: A *partition* π of a positive integer r is a sequence $(t_1, t_2, \ldots, t_s)$ of integers such that $t_1 \geqslant t_2 \geqslant \cdots \geqslant t_s \geqslant 1$ and $\sum_{i=1}^s t_i = r$. We may think of a partition $\pi = (t_1, t_2, \ldots, t_s)$ as an $s \times t_1$ matrix of squares and blanks, where row i contains t_i squares, all to the left. A (u, v)*-partition* of r is a partition $(t_1, t_2, \ldots, t_s)$ such that $s \leqslant u$ and $t_1 \leqslant v$. Let $\mathcal{P}(u, v, r)$ denote the set of all (u, v)-partitions of r.

Definition 4: For a partition $\pi = (t_1, t_2, \ldots, t_s)$ in $\mathcal{P}(k_1, k_2, r)$, define ∇_π by

$$\nabla_\pi = \nabla_\pi(C_1, C_2) = \sum_{i=1}^{s} (d_i^1 - d_{i-1}^1)\, d_{t_i}^2$$

and $d_r^*(C_1 \otimes C_2)$ by

$$d_r^*(C_1 \otimes C_2) = \min\{\nabla_\pi(C_1, C_2) | \pi \in \mathcal{P}(k_1, k_2, r)\}. \qquad (1)$$

We note that the chain condition was not assumed in the definition of d_r^*. A result similar to the following lemma is stated in [2] with the assumption of the chain condition for both C_1 and C_2. In the following, we do not assume the chain condition.

Lemma 5: For all r such that $1 \leqslant r \leqslant k_1 k_2$

$$d_r^*(C_1 \otimes C_2) = d_r^*(C_2 \otimes C_1).$$

Proof: The result follows directly from the observation

$$\begin{aligned}\nabla_\pi(C_1, C_2) &= \sum_{i=1}^{s} (d_i^1 - d_{i-1}^1)\, d_{t_i}^2 \\ &= \sum_{i=1}^{s} \sum_{j=1}^{t_i} (d_i^1 - d_{i-1}^1)(d_j^2 - d_{j-1}^2)\end{aligned}$$

made by Helleseth and Kløve [4] □.

Lemma 6 [2]: If C_1 and C_2 satisfy the chain condition, then for all r with $1 \leqslant r \leqslant k_1 k_2$

$$d_r^*(C_1 \otimes C_2) \geqslant d_r(C_1 \otimes C_2).$$

Remark 7: Wei and Yang [2] conjectured that the equality in Lemma 6 holds for all r if both C_1 and C_2 satisfy the chain condition. They established the conjecture in a few cases: for the product of two even-weight codes, for the product of a dual Hamming code and an even-weight code, and for the product of a first-order Reed–Muller code and an even-weight code. Barbero and Tena [3] proved the conjecture for $r \leqslant 4$ without assuming the chain condition. In Theorem 8 of the following section, we do not assume the chain condition for C either.

Helleseth and Kløve [4] used the following method to calculate ∇_π directly from the weight hierarchies of component codes. We will use this method in the next section.

Let $\pi = (t_1, t_2, \ldots, t_s) \in \mathcal{P}(k_1, k_2, r)$. From the expression of ∇_π (in the Proof of Lemma 5) we have

$$\nabla_\pi = \sum_{i=1}^{s} \sum_{j=1}^{t_i} \Delta_{ij} \qquad (2)$$

where $\Delta_{ij} = (d_i^1 - d_{i-1}^1)(d_j^2 - d_{j-1}^2)$. Consider the $k_1 \times k_2$ matrix

$$\Delta = \begin{pmatrix} \Delta_{11} & \Delta_{12} & \ldots & \Delta_{1k_2} \\ \Delta_{21} & \Delta_{22} & \ldots & \Delta_{2k_2} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta_{k_1 1} & \Delta_{k_1 2} & \ldots & \Delta_{k_1 k_2} \end{pmatrix}.$$

Then, by (2), we get ∇_π by taking the sum of the elements of the matrix covered by π.

III. Weight Hierarchy $C \otimes \mathbb{F}_2^n$

Here, C represents an arbitrary binary linear code.

Theorem 8: Let C be an $[m, k]$ code. Then, for all r, $0 \leqslant r \leqslant kn$

$$d_r(C \otimes \mathbb{F}_2^n) = d_r^*(C \otimes \mathbb{F}_2^n).$$

Proof: Let D be an r-dimensional subcode of $C \otimes \mathbb{F}_2^n$. We may think of a codeword of $C \otimes \mathbb{F}_2^n$ as a $1 \times n$ row vector $[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$ where $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ are $m \times 1$ column vectors of C. We now can consider an $r \times n$ generator "matrix" of D, whose entries are $m \times 1$

column vectors of C. After elementary row operations and reordering of column positions if necessary, we have a generator matrix

$$\mathcal{D} = \begin{bmatrix} B_1 & & & & \\ & B_2 & & * & \\ & & B_3 & & * \\ & O & & \ddots & \\ & & & & B_s \end{bmatrix}_{r\times n}$$

where

$$B_i = \begin{bmatrix} \boldsymbol{c}_{i1} \\ \boldsymbol{c}_{i2} \\ \vdots \\ \boldsymbol{c}_{it_i} \end{bmatrix}, \quad \begin{array}{l} \boldsymbol{c}_{ij} \in C \\ \boldsymbol{c}_{i1}, \boldsymbol{c}_{i2}, \ldots, \boldsymbol{c}_{it_i} \text{ linearly independent} \\ (t_1, t_2, \ldots t_s) \in \mathcal{P}(n, k, r). \end{array} \tag{3}$$

Consider the matrix $\mathcal{D}'$ obtained from $\mathcal{D}$ by changing $*$ to 0

$$\mathcal{D}' = \begin{bmatrix} B_1 & & & & \\ & B_2 & & 0 & \\ & & B_3 & & 0 \\ & 0 & & \ddots & \\ & & & & B_s \end{bmatrix}_{r\times n} \quad \text{with (3).} \tag{4}$$

The rows of $\mathcal{D}'$ are linearly independent, so it generates an r-dimensional subcode D' of $C \otimes \mathbb{F}_2^n$, such that $|\chi(D')| \leqslant |\chi(D)|$.

On the other hand, for any partition $(t_1, t_2, \ldots t_s) \in \mathcal{P}(n, k, r)$, we can find t_i linearly independent vectors of C for all $i = 1, 2, \ldots, s$. Thus there is an r-dimensional subcode, whose generator matrix is of the form (4). Therefore, in order to find $d_r(C \otimes \mathbb{F}_2^n)$, we only need to consider the subcodes D' that have a generator matrix of the form (4).

For a fixed partition $\pi = (t_1, t_2, \ldots, t_s)$ of r

$$|\chi(D')| = \sum_{i=1}^{s} |\chi(B_i)| = \sum_{i=1}^{s} \left| \bigcup_{j=1}^{t_i} \chi(\boldsymbol{c}_{ij}) \right|$$

is minimal when $|\bigcup_{j=1}^{t_i} \chi(\boldsymbol{c}_{ij})|$ are minimal for all $i = 1, 2, \ldots, s$. This happens when $|\bigcup_{j=1}^{t_i} \chi(\boldsymbol{c}_{ij})| = d_{t_i}(C)$. Hence

$$\begin{aligned} d_r(C \otimes \mathbb{F}_2^n) &= \min\left\{ \sum_{i=1}^{s} d_{t_i}(C) | \pi = (t_1, \ldots, t_s) \in \mathcal{P}(n, k, r) \right\} \\ &= \min\{\nabla_\pi(C, \mathbb{F}_2^n) | \pi \in \mathcal{P}(n, k, r)\} \\ &= d_r^*(C \otimes \mathbb{F}_2^n). \qquad \square \end{aligned}$$

Let G denote the $[24, 12, 8]$ extended Golay code. The weight hierarchy of G is [1]

$$\{8, 12, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24\} \tag{5}$$

As a corollary to the previous theorem, we determine the weight hierarchy of $\mathbb{F}_2^n \otimes G$ by finding $d_r^*(\mathbb{F}_2^n \otimes G)$.

Corollary 9: For $1 \leqslant r \leqslant 12n$, $d_r(\mathbb{F}_2^n \otimes G) = 24(a-1) + d_b(G)$ where a, b are positive integers such that $r = 12(a-1) + b$, $0 < b \leqslant 12$.

Proof: The $[n, n, 1]$ code $\mathbb{F}_2^n$ has weight hierarchy $\{d_i^1 = i | 0 \leqslant i \leqslant n\}$ and so $d_i^1 - d_{i-1}^1 = 1$ for all $i = 1, 2, \ldots, n$. The Golay code G has weight hierarchy as in (5). Hence we have

$$\Delta = \begin{pmatrix} 8 & 4 & 2 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 8 & 4 & 2 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}_{n\times 12}.$$

For a fixed r with $1 \leqslant r \leqslant 12n$, we can find unique positive integers a and b such that $r = 12(a-1) + b$, $0 < b \leqslant 12$. Let

$$\pi_0 = (\underbrace{12, 12, \ldots, 12}_{a-1}, b) \in \mathcal{P}(n, 12, r).$$

We will show that $d_r^* = \nabla_{\pi_0}$ for all r. We claim that: If $\pi \neq \pi_0$, then we can find a partition $\pi' \in \mathcal{P}(n, 12, r)$, obtained from π by moving the squares, such that $\nabla_{\pi'} \leqslant \nabla_\pi$ and π' is closer to π_0 than π is.

Consider $\pi = (t_1, t_2, \ldots, t_s)$ and let l be such that $t_i = 12$ for each $i < l$, but $t_l < 12$. Note that $s > l$ (and so $t_l \neq 0$) under our assumption $\pi \neq \pi_0$. If $t_l + 1 \neq 2, 3, 6$, define π' by

$$\pi' = (t_1, \ldots, t_{l-1}, t_l + 1, t_{l+1}, \ldots, t_{s-1}, t_s - 1)$$

that is, one square is moved from row s to row l. Since $\Delta_{l, t_l+1} = 1 \leqslant \Delta_{s, t_s}$, we have $\nabla_{\pi'} \leqslant \nabla_\pi$.

If $t_l + 1 = 2, 3$, or 6, then $t_l \leqslant 5$ and so $t_s \leqslant 5$. Hence, we have $t_l + t_s \leqslant 10 \leqslant 12 = t_{l-1}$ and

$$\pi' = (t_1, \ldots, t_{l-1}, t_l + t_s, t_{l+1}, \ldots, t_{s-1})$$

is an $(n, 12)$-partition of r, obtained from π by removing row s and adding t_s squares to row l. The difference $\nabla_\pi - \nabla_{\pi'}$ is positive because

$$\begin{aligned} \nabla_\pi - \nabla_{\pi'} &= d_{t_s}^2 - (d_{t_l+t_s}^2 - d_{t_l}^2) \\ &\geqslant 8 - (d_{2t_l}^2 - d_{t_l}^2) \\ &\geqslant 2 \end{aligned}$$

for $t_l = 1, 3, 5$. So $\nabla_\pi \geqslant \nabla_{\pi'}$.

We continue the procedure until we reach $l = s$, i.e., until we obtain π_0. Hence $d_r^* = \nabla_{\pi_0}$ and we get $d_r^*(\mathbb{F}_2^n \otimes G) = 24(a-1) + d_b(G)$ where $r = 12(a-1) + b$, $0 < b \leqslant 12$. Therefore, by Lemma 5 and Theorem 8, we have

$$d_r(\mathbb{F}_2^n \otimes G) = 24(a-1) + d_b(G). \qquad \square$$

IV. Weight Hierarchy of $E_m \otimes G$

The $[24, 12, 8]$ extended Golay code G satisfies the chain condition [2]. The $[m+1, m, 2]$ even-weight code E_m also satisfies the chain condition. The product of the two codes $E_m \otimes G$ is a $[24(m+1), 12m, 16]$ code. In this section, we first calculate $d_r^*(E_m \otimes G)$ and then show that these upper bounds are equal to the GHWs $d_r(E_m \otimes G)$.

A. $d_r^*(E_m \otimes G)$

Let $\{d_i^1 | 1 \leqslant i \leqslant m\}$ and $\{d_j^2 | 1 \leqslant j \leqslant 12\}$ be the weight hierarchies of E_m and G, respectively. The matrix Δ for the product code $E_m \otimes G$ is the following $m \times 12$ matrix:

$$\begin{aligned} \Delta &= \Delta(E_m \otimes G) \\ &= \begin{pmatrix} 16 & 8 & 4 & 2 & 2 & 4 & 2 & 2 & 2 & 2 & 2 & 2 \\ 8 & 4 & 2 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 8 & 4 & 2 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}_{m\times 12}. \end{aligned}$$

Notation 1: For r such that $1 \leqslant r \leqslant 12m$, let s and b be the unique positive integers determined by the following:

$$r = 12(s-1) + b, \qquad 0 < b \leqslant 12.$$

Note that this s is the minimum possible number of rows that an $(m, 12)$-partition of r can have.

Lemma 10: Let r and s be as in Notation 1. Then

$$d_r^*(E_m \otimes G) = d_r^*(E_s \otimes G).$$

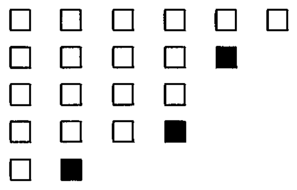


Fig. 1. The corner positions (■) in partition $(6, 4, 4, 3, 1)$

TABLE I
COMPARISON OF ∇_{t_l} with $\nabla_{\pi\backslash\pi'}$

t_l	∇_{t_l}	maximum possible $\nabla_{\pi\backslash\pi'}$
1	8	8
2	12	$6 = 4 + 2$
3	14	$8 = 2 + 2 + 4$
4	15	$10 = 2 + 2 + 2 + 4$
5	16	$12 = 2 + 2 + 2 + 2 + 4$
6	18	$12 = 2 + 2 + 2 + 2 + 2 + 2$
7	19	$13 = 2 + 2 + 2 + 2 + 2 + 2 + 1$
8	20	$14 = 2 + 2 + 2 + 2 + 2 + 2 + 1 + 1$
9	21	$15 = 2 + 2 + 2 + 2 + 2 + 2 + 1 + 1 + 1$
10	22	$16 = 2 + 2 + 2 + 2 + 2 + 2 + 1 + 1 + 1 + 1$
11	23	$17 = 2 + 2 + 2 + 2 + 2 + 2 + 1 + 1 + 1 + 1 + 1$
12	24	$18 = 2 + 2 + 2 + 2 + 2 + 2 + 1 + 1 + 1 + 1 + 1 + 1$

Proof: We view $E_s \otimes G$ as a subcode of $E_m \otimes G$ by viewing E_s as a subcode of E_m with the latest $m - s$ coordinates equal to 0. Then, by (1) and Lemma 2

$$d_r^*(E_m \otimes G) \leqslant d_r^*(E_s \otimes G).$$

Now we prove that there exists a partition $\pi' \in (s, 12, r)$ such that $\nabla_{\pi'}(E_m, G) = d_r^*(E_m \otimes G)$.

Let $\pi \in (m, 12, r)$ be such that

$$\nabla_\pi = \nabla_\pi(E_m, G) = d_r^*(E_m \otimes G).$$

If $\pi \in \mathcal{P}(s, 12, r)$ then take $\pi' = \pi$. Now suppose that

$$\pi = (t_1, t_2, \ldots, t_l) \notin \mathcal{P}(s, 12, r).$$

Then $l > s$ and there exists at least one "corner" among rows $1, \ldots, l - 1$. The "corners" are illustrated in Fig. 1.

Move the rightmost square in the last row of π to one of the corners of π. The result is a partition of r, which has $t_l - 1$ squares in row l. Since $s < l$, we can continue this procedure until all the squares in the row l are removed. The final partition $\pi' = (t_1', \ldots, t_{l-1}')$ has $l - 1$ rows, i.e., $\pi' \in \mathcal{P}(l - 1, 12, r)$. Next, we show that $\nabla_{\pi'} \leqslant \nabla_\pi$.

Let ∇_{t_l} denote the sum of the elements of Δ covered by the last row of π, and $\nabla_{\pi'\backslash\pi}$ denote the sum of the elements of Δ covered by the positions which are in π' but not in π. Then

$$\nabla_\pi = \nabla_{\pi'} + (\nabla_{t_l} - \nabla_{\pi\backslash\pi'}).$$

By Table I, $\nabla_{t_l} \geqslant \nabla_{\pi\backslash\pi'}$ for all cases. Hence $\nabla_\pi \geqslant \nabla_{\pi'}$.

If π' has more than s rows (i.e., $l - 1 > s$) then rename π' as π and repeat the procedure. We repeat this $l - s$ times to get a partition π' with s rows. In each step, ∇ does not increase, so $\nabla_\pi(E_m, G) \geqslant \nabla_{\pi'}(E_m, G)$. Also, since $\pi' \in \mathcal{P}(s, 12, r)$, $\nabla_{\pi'}(E_m, G) = \nabla_{\pi'}(E_s, G)$. Hence, we have

$$d_r^*(E_m \otimes G) = \nabla_\pi(E_m, G) \geqslant \nabla_{\pi'}(E_s, G) \geqslant d_r^*(E_s \otimes G). \qquad \square$$

Remark 11: The above Lemma implies that

$$\cdots = d_r^*(E_{m+1} \otimes G) = d_r^*(E_m \otimes G) = d_r^*(E_{m-1} \otimes G) = \cdots = d_r^*(E_s \otimes G).$$

Let d_r^* denote this common value. In order to find d_r^*, we only need to consider partitions in $\mathcal{P}(s, 12, r)$.

TABLE II
$d_r^*(E_m \otimes G)$

$s\backslash b$	1	2	3	4	5	6	7	8	9	10	11	12
1	16	24	28	30	32	36	38	40	42	44	46	48
2	56	**57**	**59**	**60**	**62**	**63**	**65**	**66**	**68**	**69**	71	72
3	80	**83**	**84**	**86**	**87**	**88**	**90**	**91**	**92**	94	95	96
4	104	108	**109**	**110**	112	**113**	**114**	**115**	117	118	119	120
5	128	132	134	135	136	**137**	**138**	140	141	142	143	144
6	152	156	158	159	160	**161**	163	164	165	166	167	168
$\vdots$							$24s + d_b(G)$					

TABLE III
$d_t(G) + d_{|b-t|}(G)$

$t\backslash b$	1	2	3	4	5	6	7	8	9	10	11	12
1	8	12	16	18	19	20	22	23	24	25	26	27
2	11	12	16	20	22	23	24	26	27	28	29	30
3	12	13	14	18	22	24	25	26	28	29	30	31
4	12	13	14	15	19	23	25	26	27	29	30	31
5	12	13	14	15	16	20	24	26	27	28	30	31
6	13	14	15	16	17	18	22	26	28	29	30	32
7	13	14	15	16	17	18	19	23	27	29	30	31
8	12	14	15	16	17	18	19	20	24	28	30	31
9	12	13	15	16	17	18	19	20	21	25	29	31
10	12	13	14	16	17	18	19	20	21	22	26	30
11	11	13	14	15	17	18	19	20	21	22	23	27
12	8	12	14	15	16	18	19	20	21	22	23	24
$d_b(G)$	8	12	14	15	16	18	19	20	21	22	23	24

Lemma 12: Let r, s, b be as in Notation 1 and let k, a be the unique integers satisfying $r = sk + a$, $0 < a \leqslant s$. Then

$$d_r^* = d_r^*(E_m \otimes G) = \begin{cases} 2\,d_r(G), & \text{if } 1 \leqslant r \leqslant 12 \\ \min\{24s + d_b(G), \\ \quad (s+1)d_k(G) + a + 1\}, & \text{if } 13 \leqslant r \leqslant 72 \\ 24s + d_b(G) & \text{if } r \geqslant 73. \end{cases}$$

Proof: We first define two special partitions π_0 and π_1. The partition π_0 is defined by

$$\pi_0 = (\underbrace{12, 12, \ldots, 12}_{s-1}, b) \in \mathcal{P}(s, 12, r).$$

Then

$$\nabla_{\pi_0} = \begin{cases} 2\,d_r(G), & \text{if } s = 1 \\ 24s + d_b(G), & \text{otherwise.} \end{cases}$$

The partition π_1 is defined by

$$\pi_1 = (\underbrace{k+1, k+1, \ldots, k+1}_{a}, \underbrace{k, \ldots, k}_{s-a}) \in \mathcal{P}(s, 12, r)$$

i.e., its conjugate partition $\overline{\pi}_1$ satisfies

$$\overline{\pi}_1 = (\underbrace{s, s, \ldots, s}_{k}, a) \in \mathcal{P}(k + 1, s, r).$$

Hence

$$\nabla_{\pi_1} = \begin{cases} 2\,d_r(G), & \text{if } s = 1 \\ (s+1)d_k(G) + a + 1, & \text{otherwise.} \end{cases}$$

Recall

$$d_r^* = d_r^*(E_s \otimes G) = \min\{\nabla_\pi(E_s, G) | \pi \in \mathcal{P}(s, 12, r)\}.$$

Case $s = 1 (1 \leqslant r \leqslant 12)$: Since there is only one partition in $\mathcal{P}(1, 12, r)$, $\pi_0 = \pi_1$ and $d_r^* = \nabla_{\pi_0} = \nabla_{\pi_1} = 2\,d_r(G)$.

TABLE IV
$\max\{d_t(G)+d^*_{r-t},\, 3\,d_t(G)\}$ $[s=1,\, 1\leqslant r\leqslant 12]$

$t\backslash b$	1	2	3	4	5	6	7	8	9	10	11	12
1	$\boxed{24}$	$\boxed{24}$	$\boxed{32}$	$\boxed{36}$	$\boxed{38}$	$\boxed{40}$	$\boxed{44}$	46	48	50	$\boxed{52}$	$\boxed{54}$
2		36	36	$\boxed{36}$	40	42	$\boxed{44}$	48	50	52	54	56
3			42	42	42	42	$\boxed{44}$	46	50	52	54	56
4				45	45	45	45	$\boxed{45}$	$\boxed{47}$	51	53	55
5					48	48	48	48	48	$\boxed{48}$	$\boxed{52}$	$\boxed{54}$
6						54	54	54	54	54	54	54
7							57	57	57	57	57	57
8								60	60	60	60	60
9									63	63	63	63
10										66	66	66
11											69	69
12												72
d^*_r	16	24	28	30	32	36	38	40	42	44	46	48

TABLE V
$\max\{d_t(G)+d^*_{r-t},\, 3\,d_t(G)\}$ $[s=2,\, 13\leqslant r\leqslant 24]$

$t\backslash b$	1	2	3	4	5	6	7	8	9	10	11	12
1	$\boxed{56}$											
2	58	60										
3	58	60	63									
4	57	59	61	63								
5	$\boxed{56}$	58	60	62	64							
6	$\boxed{56}$	58	60	62	64	66						
7	57	$\boxed{57}$	$\boxed{59}$	61	63	65	67					
8	60	60	60	$\boxed{60}$	$\boxed{62}$	64	66	68				
9	63	63	63	63	63	$\boxed{63}$	$\boxed{65}$	67	69			
10	66	66	66	66	66	66	66	$\boxed{66}$	$\boxed{68}$	70		
11	69	69	69	69	69	69	69	69	69	$\boxed{69}$	$\boxed{71}$	
12	72	72	72	72	72	72	72	72	72	72	72	$\boxed{72}$
d^*_r	56	57	59	60	62	63	65	66	68	69	71	72

TABLE VI
$\max\{d_t(G)+d^*_{r-t},\, 4\,d_t(G)\}$ $[s=3,\, 25\leqslant r\leqslant 36]$

$t\backslash b$	1	2	3	4	5	6	7	8	9	10	11	12
1	$\boxed{80}$											
2	83	84										
3	83	85	86									
4	83	84	86	87								
5	82	84	85	87	88							
6	83	84	86	87	89	90						
7	82	84	85	87	88	90	91					
8	82	$\boxed{83}$	85	$\boxed{86}$	88	89	91	92				
9	84	84	$\boxed{84}$	$\boxed{86}$	$\boxed{87}$	89	$\boxed{90}$	92	93			
10	88	88	88	88	88	$\boxed{88}$	$\boxed{90}$	$\boxed{91}$	93	$\boxed{94}$		
11	92	92	92	92	92	92	92	92	$\boxed{92}$	$\boxed{94}$	$\boxed{95}$	
12	96	96	96	96	96	96	96	96	96	96	96	$\boxed{96}$
d^*_r	80	83	84	86	87	88	90	91	92	94	95	96

Case $2\leqslant s\leqslant 6 (13\leqslant r\leqslant 72)$: For each r, there are at most 11 partitions in $\mathcal{P}(s,\, 12,\, r)$. Careful examination reveals that the minimum ∇_π occurs when the partition π is either π_0 or π_1. Hence

$$\begin{aligned} d^*_r &= \min\{\nabla_{\pi_0}, \nabla_{\pi_1}\} \\ &= \min\{24s+d_b(G),\, (s+1)d_k(G)+a+1\}. \end{aligned}$$

Case $s\geqslant 7 (r\geqslant 73)$: Let $\pi' = (t_1,\, t_2,\, \ldots,\, t_s)\in \mathcal{P}(s,\, 12,\, r)$ be different from $\pi_0 = (12,\, \ldots,\, 12,\, b)$. Then $t_s > b$ and hence

$$\nabla_{t_s} \geqslant \nabla_{b+1} \geqslant \nabla_b + \epsilon$$

where $\epsilon\in\{1,\, 2,\, 4\}$. Hence $\nabla_{\pi'} - \nabla_{\pi_0} \geqslant \epsilon - 1 \geqslant 0$. So

$$d^*_r = \nabla_{\pi_0} = 24s + d_b(G). \qquad \square$$

TABLE VII
$\max\{d_t(G) + d^*_{r-t},\, 5\, d_t(G)\}$ $[s = 4,\, 37 \leqslant r \leqslant 48]$

$t\backslash b$	1	2	3	4	5	6	7	8	9	10	11	12
1	$\boxed{104}$											
2	107	$\boxed{108}$										
3	108	109	110									
4	107	109	110	111								
5	107	$\boxed{108}$	110	111	$\boxed{112}$							
6	108	109	110	112	$\boxed{112}$	114						
7	107	109	110	111	113	114	115					
8	107	$\boxed{108}$	110	111	$\boxed{112}$	114	115	116				
9	107	$\boxed{108}$	$\boxed{109}$	111	$\boxed{112}$	$\boxed{113}$	115	116	$\boxed{117}$			
10	110	110	110	$\boxed{110}$	$\boxed{112}$	$\boxed{113}$	$\boxed{114}$	116	$\boxed{117}$	$\boxed{118}$		
11	115	115	115	115	115	115	115	$\boxed{115}$	$\boxed{117}$	$\boxed{118}$	$\boxed{119}$	
12	120	120	120	120	120	120	120	120	120	120	120	$\boxed{120}$
d^*_r	104	108	109	110	112	113	114	115	117	118	119	120

TABLE VIII
$\max\{d_t(G) + d^*_{r-t},\, 6\, d_t(G)\}$ $[s = 5,\, 49 \leqslant r\cdot \leqslant 60]$

$t\backslash b$	1	2	3	4	5	6	7	8	9	10	11	12
1	$\boxed{128}$											
2	131	$\boxed{132}$										
3	132	133	$\boxed{134}$									
4	132	133	$\boxed{134}$	$\boxed{135}$								
5	131	133	$\boxed{134}$	$\boxed{135}$	$\boxed{136}$							
6	132	133	135	136	137	138						
7	132	133	$\boxed{134}$	136	137	138	139					
8	132	133	$\boxed{134}$	$\boxed{135}$	137	138	139	$\boxed{140}$				
9	131	133	$\boxed{134}$	$\boxed{135}$	$\boxed{136}$	138	139	$\boxed{140}$	$\boxed{141}$			
10	132	$\boxed{132}$	$\boxed{134}$	$\boxed{135}$	$\boxed{136}$	$\boxed{137}$	139	$\boxed{140}$	$\boxed{141}$	$\boxed{142}$		
11	138	138	138	138	138	138	$\boxed{138}$	$\boxed{140}$	$\boxed{141}$	$\boxed{142}$	$\boxed{143}$	
12	144	144	144	144	144	144	144	144	144	144	144	$\boxed{144}$
d^*_r	128	132	134	135	136	137	138	140	141	142	143	144

In Table II, the number on the $(s,\, b)$-position is d^*_r where $r = 12(s-1) + b,\, 0 < b \leqslant 12$. Except for the positions with bold numbers, $d^*_r = \nabla_{\pi_0}$. For the positions with bold numbers, $d^*_r = \nabla_{\pi_1} < \nabla_{\pi_0}$ with $\nabla_{\pi_0} - \nabla_{\pi_1} \in \{1,\, 2,\, 3\}$.

B. GHW of $E_m \otimes G$

Lemma 13: For all $m \geqslant 1$ and $1 \leqslant r \leqslant 12m$,

$$d_r(E_m \otimes G) \geqslant d^*_r.$$

Proof: Induction on m. Let D be an r-dimensional subcode of $E_m \otimes G$. We will show that $|\chi(D)| \geqslant d^*_r$. Let s and b be the unique positive numbers such that

$$r = 12(s-1) + b,\ 0 < b \leqslant 12.$$

Suppose $m = 1$. Then $1 \leqslant r \leqslant 12$ and $s = 1$. The elements of $E_1 \otimes G$ are 2×24 matrices with identical rows, and these rows are Golay codewords. A basis of D consists of r linearly independent matrices of this type. Thus we have

$$|\chi(D)| \geqslant 2\, d_r(G) = d^*_r, \qquad \text{for } 1 \leqslant r \leqslant 12$$

by Lemma 12.

Now assume $m > 1$ and $1 \leqslant r \leqslant 12m$. Let R_i denote the subspace of G consisting of ith rows of all the elements in D. Without loss of generality

$$0 < \dim R_1 \leqslant \dim R_i, \qquad 1 \leqslant i \leqslant m+1.$$

Write $t = \dim R_1$. Then by the monotonicity of GHW

$$d_t(G) < d_{\dim R_i}(G) \qquad \forall\, i > 1.$$

Also, by the definition of GHW

$$|\chi(R_i)| \geqslant d_{\dim R_i}(G) \qquad \forall\, i.$$

So we have

$$|\chi(D)| = \sum_{i=1}^{m+1} |\chi(R_i)| \geqslant \sum_{i=1}^{m+1} d_{\dim R_i}(G) > (m+1)\, d_t(G).$$

Since $m \geqslant s \geqslant 1$ and $m \geqslant 2$, we have

$$|\chi(D)| \geqslant \begin{cases} 3\, d_t(G), & \text{if } s = 1 \\ (s+1)\, d_t(G), & \text{if } s \geqslant 2. \end{cases}$$

Consider r basis elements for D. Using the elementary "row" operations on these r $(m+1) \times 24$ matrices as "rows," we get a basis with the following properties:

i) the first t elements have linearly independent first rows, which are Golay codewords;

ii) the remaining $r - t$ basis elements have all zero first rows and, viewed as elements of $E_{m-1} \otimes G$, they generate an $(r-t)$-di-

TABLE IX
$\max\{d_t(G)+d^*_{r-t},\, 7\,d_t(G)\}$ $[s=6,\, 61 \leqslant r \leqslant 72]$

$t\backslash b$	1	2	3	4	5	6	7	8	9	10	11	12
1	$\boxed{152}$											
2	155	$\boxed{156}$										
3	156	157	$\boxed{158}$									
4	156	157	$\boxed{158}$	$\boxed{159}$								
5	156	157	$\boxed{158}$	$\boxed{159}$	$\boxed{160}$							
6	156	158	159	160	161	162						
7	156	157	159	160	161	162	$\boxed{163}$					
8	156	157	$\boxed{158}$	160	161	162	$\boxed{163}$	$\boxed{164}$				
9	156	157	$\boxed{158}$	$\boxed{159}$	161	162	$\boxed{163}$	$\boxed{164}$	$\boxed{165}$			
10	156	157	$\boxed{158}$	$\boxed{159}$	$\boxed{160}$	162	$\boxed{163}$	$\boxed{164}$	$\boxed{165}$	$\boxed{166}$		
11	161	161	161	161	161	$\boxed{161}$	$\boxed{163}$	$\boxed{164}$	$\boxed{165}$	$\boxed{166}$	$\boxed{167}$	
12	168	168	168	168	168	168	168	168	168	168	168	$\boxed{168}$
d^*_r	152	156	158	159	160	161	163	164	165	166	167	168

TABLE X
$\max\{d_t(G)+d^*_{r-t},\, 8\,d_t(G)\}$ $[s=7,\, 73 \leqslant r \leqslant 84]$

$t\backslash b$	1	2	3	4	5	6	7	8	9	10	11	12
1	$\boxed{176}$											
2	179	$\boxed{180}$										
3	180	181	$\boxed{182}$									
4	180	181	$\boxed{182}$	$\boxed{183}$								
5	180	181	$\boxed{182}$	$\boxed{183}$	$\boxed{184}$							
6	181	182	183	184	185	$\boxed{186}$						
7	180	182	183	184	185	$\boxed{186}$	$\boxed{187}$					
8	180	181	183	184	185	$\boxed{186}$	$\boxed{187}$	$\boxed{188}$				
9	180	181	$\boxed{182}$	184	185	$\boxed{186}$	$\boxed{187}$	$\boxed{188}$	$\boxed{189}$			
10	180	181	$\boxed{182}$	$\boxed{183}$	185	$\boxed{186}$	$\boxed{187}$	$\boxed{188}$	$\boxed{189}$	$\boxed{190}$		
11	184	184	184	184	$\boxed{184}$	$\boxed{186}$	$\boxed{187}$	$\boxed{188}$	$\boxed{189}$	$\boxed{190}$	$\boxed{191}$	
12	192	192	192	192	192	192	192	192	192	192	192	$\boxed{192}$
d^*_r	176	180	182	183	184	186	187	188	189	190	191	192

mensional subcode of $E_{m-1}\otimes G$. Hence, we have another lower bound for $|\chi(D)|$

$$\begin{aligned}|\chi(D)| &= |\chi(R_1)| + \sum_{i=2}^{m+1} |\chi(R_i)| \\ &\geqslant d_t(G) + d_{r-t}(E_{m-1}\otimes G) \\ &\geqslant d_t(G) + d^*_{r-t} \quad \text{by induction hypothesis.} \qquad (7)\end{aligned}$$

We show that one of the lower bounds (6) and (7) is bigger than d^*_r.

Case $s=1 (1\leqslant r\leqslant 12)$: In this case, $t\leqslant r=b$. In Table IV, the number in $(t,\, b)$ position is $\max\{3\,d_t(G),\, d_t(G)+d^*_{r-t}\}$, which is a lower bound for $|\chi(D)|$. For each b (and so for each r), the smallest number is boxed for comparison with d^*_r. The last row of the table is $d^*_r s$. By the table, we see that $d^*_r \leqslant |\chi(D)|$ in this case.

Case $2\leqslant s\leqslant 7 (13\leqslant r\leqslant 84)$: If $b>t$, then

$$d^*_{r-t} = 24s + d_{b-t}(G) - \epsilon$$

where $\epsilon = 0,\, 1,\, 2$, or 3. Hence

$$\begin{aligned}|\chi(D)| &\geqslant d_t(G) + d^*_{r-t} \quad \text{by (7)} \\ &= d_t(G) + 24s + d_{b-t}(G) - \epsilon \\ &= 24s + [d_t(G) + d_{b-t}(G)] - \epsilon \\ &\geqslant 24s + [d_b(G) + 4] - \epsilon \quad \text{by Table III} \\ &> 24s + d_b(G) = \nabla_{\pi_0} \\ &\geqslant d^*_r.\end{aligned}$$

If $b\leqslant t$, Tables IV–X show that

$$\begin{aligned}|\chi(D)| &\geqslant \max\{(s+1)d_t(G),\, d^*_{r-t} + d_t(G)\} \\ &\geqslant d^*_r.\end{aligned}$$

Case $s\geqslant 8 (r\geqslant 85)$: If $b>t$, then

$$d^*_{r-t} = 24s + d_{b-t}(G).$$

Hence

$$\begin{aligned}|\chi(D)| &\geqslant d_t(G) + 24s + d_{b-t}(G) \\ &= 24s + [d_t(G) + d_{b-t}(G)] \\ &> 24s + d_b(G) \quad \text{by Table III} \\ &= d^*_r.\end{aligned}$$

If $b\leqslant t$, then

$$d^*_{r-t} = 24(s-1) + d_{12+b-t}(G).$$

Hence

$$\begin{aligned}|\chi(D)| &\geqslant d_t(G) + 24(s-1) + d_{12+b-t}(G) \\ &\geqslant 24s + [d_t(G) + d_{12+b-t}(G) - 24] \\ &\geqslant 24s + d_b(G) \quad \text{by Table III} \\ &= d^*_r.\end{aligned}$$

Therefore, for all $m \geqslant 1$ and $1 \leqslant r \leqslant 12m$

$$d_r(E_m \otimes G) \geqslant d_r^*.$$

Theorem 14: For all $m \geqslant 1$ and $1 \leqslant r \leqslant 12m$

$$d_r(E_m \otimes G) = d_r^* = \begin{cases} 2\,d_r(G), & \text{if } 1 \leqslant r \leqslant 12 \\ \min\{24s + d_b(G), \\ \quad (s+1)d_k(G) + a + 1\}, & \text{if } 13 \leqslant r \leqslant 72, \\ 24s + d_b(G), & \text{if } r \geqslant 73. \end{cases}$$

Proof: Let s denote the integer determined uniquely by

$$r = 12(s-1) + b \quad \text{and} \quad 0 < b \leqslant 12.$$

Let D be an r-dimensional subcode of $E_s \otimes G$ such that $|\chi(D)| = d_r(E_s \otimes G)$.

By appending $(m - s)$ zero rows to each element of D, we get an r-dimensional subcode D' of $E_m \otimes G$ with the same support as D. Thus we have

$$d_r(E_m \otimes G) \leqslant |\chi(D')| = d_r(E_s \otimes G).$$

Also, we saw $d_r(E_s \otimes G) \leqslant d_r^*$ in the previous section. Combine above with the previous lemma to get

$$d_r^* \leqslant d_r(E_m \otimes G) \leqslant d_r(E_s \otimes G) \leqslant d_r^*.$$

Therefore,

$$d_r^* = d_r(E_m \otimes G), \qquad \text{for all } m. \qquad \square$$

V. Weight Hierarchy of $E_m \otimes H$

The $[8, 4, 4]$ extended Hamming code H satisfies the chain condition and has the weight hierarchy $\{4, 6, 7, 8\}$ [1]. The product of an even-weight code and the Hamming code $E_m \otimes H$ is an $[8(m+1), 4m, 8]$ code. Using the same method as in the $E_m \otimes G$ case, we can determine the weight hierarchy of $E_m \otimes H$.

Theorem 15: For all $m \geqslant 1$ and $1 \leqslant r \leqslant 4m$

$$d_r(E_m \otimes H) = d_r^* = \begin{cases} 2\,d_b(H), & \text{if } 1 \leqslant r \leqslant 4 \\ 21, & \text{if } r = 6 \\ 8s + d_b(H), & \text{otherwise} \end{cases}$$

where s, b are unique integers satisfying $r = 4(s-1) + b$, $0 < b \leqslant 4$.

References

[1] V. K. Wei, "Generalized hamming weights for linear codes," *IEEE Trans. Inform. Theory*, vol. 37, pp. 1412–1418, Sept. 1991.

[2] V. K. Wei and K. Yang, "On the generalized hamming weights of product codes," *IEEE Trans. Inform. Theory*, vol. 39, pp. 1709–1713, Sept. 1993.

[3] A. I. Barbero and J. G. Tena, "Weight hierarchy of a product code," *IEEE Trans. Inform. Theory*, vol. 41, pp. 1475–1479, Sept. 1995.

[4] T. Helleseth and T. Kløve, "The weight hierarchies of some product codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1029–1034, May 1996.

[5] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes vol. I, II.* Amsterdam, The Netherlands: North-Holland, 1977.

A Linear Programming Estimate of the Weight Distribution of BCH$(255, k)$

Massimiliano Sala and Aldo Tamponi

***Abstract*—To estimate the weight distribution of a binary linear code, a linear programming approach has been proposed, which leads to the best known values for BCH$(255, k)$. Following that approach, we propose three methods which give tighter estimates. At the same time, we find the distance of the dual of BCH$[255, 199, 15]$.**

***Index Terms*—BCH codes, dual distance, linear programming, undetected error, weight distribution.**

I. Introduction

It is known that the weight distribution of a binary linear code gives its P_{ue} (Probability of Undetected Error) in the binary symmetric channel [1, p. 397].

Extensive work has been done on estimating the weight distribution of Bose–Chaudhuri–Hocquenghem (BCH) codes, in particular binary primitive codes BCH$(2^m - 1, k)$. The weight distribution of BCH$(127, k)$ codes is known for all k [2]; the weight distribution is also known for BCH$(255, k)$ codes, for $k \leq 63$ and $k \geq 207$ [3]. Many estimates have been proposed, some recently (see, for example, [4]), but for small codes (those that allow effective computation) the best estimates are obtained via the linear programming approach proposed by Kasami, Fujwara, and Lin [5]. Our work is an improvement to [5], which leads, in some cases, to significant tightening of the estimates.

II. Presentation of the Methods

Let H be a binary linear code, A_i be the number of its words of weight i and B_j the number of words (of weight j) of its dual. Then each A_i can be represented as a linear function of the B_j, via the MacWilliams identities [6], and the B_j have to satisfy the Pless identities, which are linear functions.

Choosing two codes, H_1 (called *subcode*) and H_2 (called *supercode*), such that $H_1 \subset H \subset H_2$, we define $A_i(H_1)$ $(A_i(H_2))$ as the number of words of weight i of H_1 (H_2) and $B_j(H_1)$ $(B_j(H_2))$ as the number of words of weight j of the dual of H_1 (H_2). Then

$$\forall\, j \in \{0, \ldots, n\}, \qquad B_j(H_2) \leq B_j \leq B_j(H_1) \qquad (1)$$

$$\forall\, i \in \{0, \ldots, n\}, \qquad A_i(H_1) \leq A_i \leq A_i(H_2). \qquad (2)$$

Method A (Kasami, Fujiwara, and Lin [5])

One solves a linear programming problem for each i, thinking of the B_j as independent variables, of the A_i as the function to be maximized (or minimized) and of the Pless identities on the B_j as constraints. In particular for BCH$(2^m - 1, k)$ one can simplify the problem by using known properties of such codes. First of all, one can consider the extended codes EBCH$(2^m, k)$, because in this case $A_h = 0$ and $B_h = 0$

Manuscript received October 19, 1999; revised March 24, 2000.

M. Sala is with the Department of Mathemathics, University of Milan, Milan, Italy, and also with the Department of Mathematics, University of Pisa, 56100 Pisa, Italy (e-mail: sala@posso.dm.unipi.it; sala@dm.unipi.it).

A. Tamponi is with the R&D Department, Ansaldo Segnalamento Ferroviario, 16129 Genoa, Italy (e-mail: Tamponi.Aldo@asf.ansaldo.it).

Communicated by A. Barg, Associate Editor in Coding Theory.

Publisher Item Identifier S 0018-9448(00)07012-7.

for h odd [9]. In addition, $B_{n-j} = B_j$ and $A_{n-i} = A_i$ [3], where $n = 2^m$ is the length of the extended code. In this way, the computation is simplified by reducing the number of variables (B_j) and functions (A_i). From the estimated A_i of the extended code, one can come back to the weight distribution of the nonextended one via two simple formulas which hold for binary primitive BCH codes [7, Theorem 8.15, p. 246]. Once these estimates are found, one can compute the P_{ue} for any desired bit-error rate [1, p. 397].

Method B

For any H one can always find a supercode and a subcode, at least just taking as H_1 and H_2, respectively, the smallest code constituted by the null word and the largest code constituted by the set of all possible words. If one knows the values $B_j(H_1)$, $B_j(H_2)$, then one can consider the inequalities (1) as natural constraints on the variables B_j, reducing the simplex containing the possible solutions. Knowing $A_i(H_1)$, $A_i(H_2)$, one can use (2) to obtain more constraints on the B_j, as we can replace the A_i with their expressions as functions of the B_j via the MacWilliams identities. So, as in Method A, one solves a linear programming problem for each i, thinking of the B_j as independent variables and of the A_i as the function to be maximized (or minimized), but this time the constraints are the Pless identities on B_j, (1), (2). As for Method A, one can now compute the P_{ue}.

Method C

Method B works also by using appropriate estimates for H_1 and/or H_2. So we can propose an iterative form of Method B as follows.

Suppose that there exist two other codes, G_1 and G_2, such that $G_1 \subset H_1 \subset H \subset H_2 \subset G_2$ and whose weight distributions are known. First, one applies Method B to estimate both the weight distribution of H_1 (via the inclusion relations $G_1 \subset H_1 \subset G_2$) and the weight distribution of H_2 (via the inclusion relations $G_1 \subset H_2 \subset G_2$). Second, one applies Method B to estimate the weight distribution of H (via the standard inclusion relations $H_1 \subset H \subset H_2$), using for $A_i(H_1)$, $A_i(H_2)$, $B_j(H_1)$, $B_j(H_2)$ the values found in the previous step. Finally, one computes the P_{ue}.

Method D

One can slightly modify Method B. First, one fixes the value of the bit-error rate for which the estimate of the P_{ue} is desired. Second, one solves a linear programming problem, using the same variables and the same constraints of Method B, but changing the functions to be maximized (minimized): instead of the A_i, one can directly maximize (minimize) the P_{ue}, seen as a linear function of the B_j. In this way one does not find the weight distribution of H, but one obtains still tighter estimates of the P_{ue} for the chosen bit-error rate.

III. Remarks

Actual computation has shown remarkable improvements. To measure how accurate our (upper) estimates are, we have used two parameters: $\max P_{ue}$ and the relative error on the sum (RES) of the A_i (the latter is not applicable to Method D). The parameter $\max P_{ue}$ is the maximum value of the P_{ue} in correspondence to all bit-error rates multiple of 0.01. If we call α_i an estimate of A_i, then we can define

$$\text{RES} = \left(\sum \alpha_i - \sum A_i\right) \Big/ \sum A_i \ .$$

In fact, we know the value of $\sum A_i$, which is just the total number of words in the code (i.e., 2^k if the code is an EBCH(2^m, k)). The parameter $\max P_{ue}$ is more strongly influenced by the A_i terms with lower i, while the RES is more strongly influenced by the A_i terms with higher i.

Another fact should be mentioned. Surprisingly enough, adding only the constraints $0 = A_i(H_1) \leq A_i$ gives sometimes interesting improvements, in particular it prevents the lower estimates of the A_i terms to be negative, as happens, for example, when applying Method A to BCH[255, 199, 15] to search the minima of the A_i.

IV. The Distance of the Dual of the BCH[255, 199, 15] Code is 64

A crucial point is the Hamming distance of the dual code (d'): the larger it is, the fewer variables (B_j) we have to use and the more accurate will be the result. For binary primitive BCH codes, the Hamming distance of the dual of the extended code is equal to the Hamming distance of the dual of the nonextended one [10, p. 176].

For EBCH[256, 199, 16] (hence for BCH[255, 199, 15]) we have found the exact value of d', which is a new result, as of the 1996 paper [9]. According to [8] and [9], $d' \geq 58$ and, according to [3] (with the obvious inclusion BCH[255, 199, 15] $\subset$ BCH[255, 207, 13]), it is also known that $d' \leq 64$. But, according to [10, p. 181], d' is a multiple of 4. So the possible values for d' are only 60 and 64. Denoting RM(r, m) the Reed–Muller code of order r and length 2^m, we find [6, p. 385]

$$\text{RM}(4,\, 8) \subset \text{EBCH}[256,\, 199,\, 16] \tag{3}$$

$$\text{RM}(3,\, 8) \subset \text{EBCH}[256,\, 139,\, 32] \subset \text{EBCH}[256,\, 191,\, 18] \tag{4}$$

where RM$(4, 8)$ has dimension 163 and RM$(3, 8)$ has dimension 93. The inclusion (3) implies (passing to dual codes) that the dual code of EBCH[256, 199, 16] is included in RM$(3, 8)$, as RM$(3, 8)$ is the dual of RM$(4, 8)$ [7, p. 317]. But RM$(3, 8)$ has no codeword of weight 60 [10], so for EBCH[256, 199, 16] $A_{60} = 0$, proving that d' is not 60. As a consequence, d' is 64.

V. Some Sample Cases

We present here an application of our methods to the following codes: BCH[255, 199, 15], BCH[255, 191, 17], and BCH[255, 139, 31]. We use (3) and (4), since the weight distributions of RM$(3, 8)$ and RM$(4, 8)$ are known [10].

For **EBCH[256, 199, 16]**, we take as supercode EBCH[256, 207, 14], whose weight distribution is known [3], and as subcode RM$(4, 8)$. So to BCH[255, 199, 15] we apply Method B.

For **EBCH[256, 191, 18]**, we choose RM$(3, 8)$ as H_1 and EBCH[256, 199, 16] as H_2. But the weight distribution of EBCH[256, 199, 16] is unknown, so we apply Method C, using the upper estimates we derived previously for the $A_i(H_2)$, and as lower estimates of the $B_j(H_2)$ we take the weight distribution of the dual of EBCH[256, 207, 14].

For **EBCH[256, 139, 32]**, we take RM$(3, 8)$ as H_1 and EBCH[256, 191, 18] as H_2. As before, the weight distribution of H_2 is not known, so we apply Method C, using the upper estimates we derived previously for the $A_i(H_2)$, and as lower estimates of the $B_j(H_2)$ we put the weight distribution of the dual of EBCH[256, 207, 14] (as a matter of fact, we are iterating twice).

For d' of BCH[255, 191, 17] and BCH[255, 139, 31], exact values are still unknown, so we use the best known lower estimates, respectively, 42 and 26 [8], [9]. For each code we do five computations, focusing on the maximization of the functions, summarized in Tables I and II.

- "RES 1," the RES obtained by Method A;
- "RES 2," the RES obtained by Method B or C;
- "P_{ue} 1," the $\max P_{ue}$ obtained by Method A;
- "P_{ue} 2," the $\max P_{ue}$ obtained by Method B or C;
- "P_{ue} 3," the $\max P_{ue}$ obtained by Method D.

TABLE I
COMPARISON BETWEEN UPPER ESTIMATES OF A_i

k	d'	RES 1	RES 2	H_1	H_2
199	64	.408e-23	.169e-23	163	207
191	42	.408e-17	.899e-18	93	199*
139	26	.582e-3	.541e-3	93	191*

TABLE II
COMPARISON BETWEEN UPPER ESTIMATES OF $\max P_{ue}$

k	d'	P_{ue} 1	P_{ue} 2	P_{ue} 3	H_1	H_2
199	64	.289e-14	.769e-15	.747e-15	163	207
191	42	.226e-15	.395e-16	.390e-16	93	199*
139	26	.489e-24	.121e-26	.114e-26	93	191*

TABLE III
DIFFERENT UPPER ESTIMATES OF $\max P_{ue}$ FOR BCH$[255, 139, 31]$

$\max P_{ue}$ computed via Method A	.489e-24
$\max P_{ue}$ computed via Method C	.121e-26
$\max P_{ue}$ computed via Method D	.114e-26
$\max P_{ue}$ computed via mixed approach, $s = 44$	.122e-26
$\max P_{ue}$ computed via mixed approach, $s = 38$	.254e-25

In Table I the first column contains the dimension of H, the second one contains the distance of the dual code of H (or a lower estimate), the following two columns contain the RES calculated as above, the last two columns[1] contain, respectively, the dimensions of H_1 and H_2. Table II is similar to Table I except for the middle columns, which contain $\max P_{ue}$ calculated as above.

If the computation of RES 2 (hence of P_{ue} 2) is quite costly (as for the BCH$[255, 139, 31]$) and the researcher is only interested in the P_{ue}, we suggest a mixed approach between Method A and Method B (or Method C), in three steps. First, one calculates all α_i by Method A. Second, one calculates only some α_i by Method B (or Method C), more precisely, the α_i with lower i (say $i \le s$). Finally, one calculates the estimate of the P_{ue} taking for $i \le s$ the α_i obtained by the second step and, for the remaining values, the α_i obtained by the first one.

A few results of the mixed approach for the $\max P_{ue}$ of BCH$[255, 139, 31]$ are shown in Table III.

ACKNOWLEDGMENT

The authors wish to thank T. Berger, P. Charpin, T. Fujiwara, and C. Traverso for their helpful suggestions.

REFERENCES

[1] E. R. Berlekamp, *Algebraic Coding Theory*. New York: McGraw-Hill, 1968.
[2] Y. Desaki, T. Fujiwara, and T. Kasami, "The weight distributions of extended binary primitive BCH codes of length 128," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1364–1371, July 1997.
[3] T. Fujiwara and T. Kasami, "The weight distributions of $(256, k)$ extended binary primitive BCH codes with $k \le 63$ and $k \ge 207$," IEICE, Tech. Rep. IT97-46 (1997-09), pp. 29–33.
[4] O. Keren and S. Litsyn, "More on the distance distribution of BCH codes," *IEEE Trans. Inform. Theory*, vol. 45, pp. 251–255, Jan. 1999.
[5] T. Kasami, T. Fujiwara, and S. Lin, "An approximation to the weight distribution of binary linear codes," *IEEE Trans. Inform. Theory*, vol. IT-31, pp. 769–780, Nov. 1985.
[6] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North Holland, 1977.
[7] W. W. Peterson and E. J. Weldon Jr., *Error Correcting Codes*. Cambridge, MA: MIT Press, 1972.
[8] T. Schaub, "A Linear Complexity Appoach to Cyclic Codes," Dissertation ETH no. 8730 in Technical Sciences, Swiss Federal Inst. Technol., Zürich, Switzerland, 1988.
[9] D. Augot and F. Levy-dit-Vehel, "Bounds on the minimum distance of the duals of BCH codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1257–1260, July 1996.
[10] T. Kasami, N. Tokura, and S. Azumi, "On the weight enumeration of weights less than 2.5 d of Reed–Muller codes," *Inform. Contr.*, vol. 30, pp. 380–395, Apr. 1976.
[11] F. Levy-dit-Vehel, "Bounds on the minimum distance of duals of extended BCH codes over Fp," *Applicable Algebra in Eng. Commun. Comput. AAECC*, vol. 6, pp. 175–190, 1995.

[1]"*" stands for "use of estimated values of weight distributions."

Separation of Random Number Generation and Resolvability

Karthik Visweswariah, *Member, IEEE*,
Sanjeev R. Kulkarni, *Senior Member, IEEE*, and
Sergio Verdú, *Fellow, IEEE*

***Abstract*—We consider the problem of determining when a given source can be used to approximate the output due to any input to a given channel. We provide achievability and converse results for a general source and channel. For the special case of a full-rank discrete memoryless channel we give a stronger converse result than we can give for a general channel.**

***Index Terms*—Approximation theory, channel output statistics, random number generation, resolvability, source–channel separation.**

I. INTRODUCTION

The classical separation theorem essentially states that source and channel coding can be done separately without losing optimality. In [5], a source–channel separation theorem was shown for sources and channels more general than those in the classical separation theorem. Here we investigate an analogous separation theorem in the case of resolvability and random number generation. Random number generation involves finding a deterministic transformation to generate a sequence of equiprobable bits from a given source of randomness. Resolvability is a property of a channel which gives the amount of randomness required to simulate, at the output of the channel, any distribution that can be achieved by a random input to the channel. We consider the two problems together, i.e., when can a given source of randomness be used to simulate the output of a given channel due to any input, to arbitrary accuracy. In Section II, we give some notation and background to state

Manuscript received May 5, 1999; revised April 6, 2000. This work was supported in part by the National Science Foundation under Grants NYI Award IRI-9457645 and NCR 9523805. The material in this correspondence was presented in part at the Allerton Conference on Communication, Control and Computing, Monticello, IL, September 1998.

K. Visweswariah is with IBM T. J. Watson Research Center, Yorktown Heights NY 10598 USA (e-mail: kv1@watson.ibm.com).

S. R. Kulkarni and S. Verdú are with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: kulkarni @ee.princeton.edu: verdu@ee.princeton.edu).

Communicated by I. Csiszár, Associate Editor for Shannon Theory.
Publisher Item Identifier S 0018-9448(00)07014-0.

the problem precisely. Sections III and IV give achievability and converse results for a source to approximate the output of a given channel due to any input. We have not been able to show results fully analogous to [5] and there is a gap between the achievability and converse results that we have not been able to bridge. Section V deals with the special case when the channel is discrete-memoryless and of full rank. For this special case we will be able to show a converse which is stronger than the converse in Section IV for a general channel.

II. Preliminaries and Problem Formulation

In this section we give, some basic definitions, a precise statement of the problem to be considered, and results which are already known for the problem of separation of resolvability and random number generation.

The following definition is as in [3] and we repeat it here for the sake of completeness.

Definition 1: A channel $\boldsymbol{W}$ with input alphabet A and output alphabet B is a sequence of conditional distributions

$$\boldsymbol{W} = \{W^n(y^n|x^n): (x^n, y^n) \in A^n \times B^n\}_{n=1}^{\infty}.$$

Definition 2: A source of randomness $\boldsymbol{X}$ is a sequence of finite-dimensional distributions $\{P_{X^n}\}_{n=1}^{\infty}$ with X^n taking values in A^n.

Note that there are no consistency requirements on the sequence of finite-dimensional distributions, this is the difference between Definition 2 and the standard definition of a random process. Throughout we assume that the source and channel alphabets are finite.

We now give some notation that is used throughout this correspondence. Let the output distribution, when the input is distributed according to Q^n, be denoted by Q^nW^n. Thus

$$Q^nW^n(y^n) = \sum_{x^n \in A^n} W^n(y^n|x^n)Q^n(x^n).$$

Also let

$$i_{X^nW^n}(a^n, b^n) \triangleq \log \frac{W^n(b^n|a^n)}{P_{Y^n}(b^n)}$$

where P_{Y^n} is the output distribution when P_{X^n} is input to the channel W^n, and

$$h_{Z^n}(z^n) \triangleq \log \frac{1}{P_{Z^n}(z^n)}.$$

We will use l_1 distance to measure the difference between two distributions on the same alphabet. We will denote the distance between P and Q by $d(P, Q)$. We note that

$$d(P, Q) = 2 \sup_{E \subseteq A} |P(E) - Q(E)|$$

where A is the alphabet on which the two distributions are defined. If X and Y are two random variables on the same alphabet we sometimes write $d(X, Y)$ for the l_1 distance between the distributions of X and Y.

We now define precisely what we mean by a source $\boldsymbol{Z}$ being an approximating source for a channel $\boldsymbol{W}$.

Definition 3: For any $\epsilon > 0$, the source $\boldsymbol{Z}$ with alphabet F is called an ϵ-approximating source for the channel $\boldsymbol{W}$ if for any arbitrary input source $\tilde{\boldsymbol{X}}$, there exists a sequence of deterministic mappings $\{\phi_n: F^n \longmapsto A^n\}$ such that for sufficiently large n

$$d(Y^n, \tilde{Y}^n) < \epsilon$$

where Y^n and $\tilde{Y}^n$ are the outputs of the channel due to $\phi(Z^n)$ and $\tilde{X}^n$, respectively.

Using this definition of an ϵ-approximating source we can now define $\boldsymbol{Z}$ to be an *approximating source* for $\boldsymbol{W}$ if it is ϵ-approximating for $\boldsymbol{W}$ for all $\epsilon > 0$.

The resolvability of a channel is the minimum number of random bits required per input sample to approximate arbitrarily well the output of the channel due to any input process (see [3] for a formal definition of resolvability). The problem of resolvability of a channel was first considered by Han and Verdú ([3]) where it was shown that the resolvability of a channel is $\sup_{\boldsymbol{X}} \overline{I}(\boldsymbol{X}, \boldsymbol{Y})$ (For a definition of $\overline{I}(\boldsymbol{X}, \boldsymbol{Y})$ see [3]). The problem we consider here is that of finding necessary and sufficient conditions for a given source $\boldsymbol{Z}$ to be an approximating source for a channel $\boldsymbol{W}$. To use a given source to approximate the output of a channel due to another source we could use the source to generate random bits at the best possible rate and then use a deterministic transformation of the random bits at the input of the channel. The problem of random bit generation was considered by Vembu and Verdú [6], where fundamental limits on the rate at which random bits can be generated from a given source were given. Using the results of [3] and [6] and the two-step process outlined above we can easily show the following sufficient condition.

Theorem 1: If $\underline{H}(\boldsymbol{Z}) > S$ then $\boldsymbol{Z}$ is an approximating source for a channel $\boldsymbol{W}$, where S is the resolvability of the channel $\boldsymbol{W}$.

The converse below can also be derived using the results in [3]. A source $\boldsymbol{Z}$ can be approximated using $\overline{H}(\boldsymbol{Z})$ random bits per sample [3, Theorem 3]. This also means any process derived from $\boldsymbol{Z}$ by a deterministic transformation can be approximated using $\overline{H}(\boldsymbol{Z})$ random bits per sample. Thus if a source $\boldsymbol{Z}$ is an approximating source for a channel then we can approximate any output of the channel with $\overline{H}(\boldsymbol{Z})$ random bits per sample and so the resolvability S of the channel must be smaller than $\overline{H}(\boldsymbol{Z})$. Thus we have the following theorem.

Theorem 2: If $\overline{H}(\boldsymbol{Z}) < S$ then $\boldsymbol{Z}$ is not an approximating source for a channel $\boldsymbol{W}$.

In the next two sections we will give an achievable and converse results in the spirit of [5].

III. Achievability

In this section we give a sufficient condition for a source $\boldsymbol{Z}$ to be an approximating source for a channel $\boldsymbol{W}$ which implies the sufficient condition given by Theorem 1. The sufficient condition is analogous to a sufficient condition derived in [5] for the source–channel separation problem. We will first define the notion of a source strictly dominating a channel.

Definition 4: A source $\boldsymbol{Z}$ is said to strictly dominate a channel $\boldsymbol{W}$ if for every channel input process $\boldsymbol{X}$ there exists a $\delta > 0$ such that

$$\lim_{n\to\infty} \inf_{c_n \in r} \left\{ P\left[\frac{1}{n} h_{Z^n}(Z^n) \le c_n + \delta\right] + P\left[\frac{1}{n} i_{X^nW^n}(X^n; Y^n) \ge c_n\right]\right\} = 0.$$

We note that $\underline{H}(\boldsymbol{Z}) > S$ implies that $\boldsymbol{Z}$ strictly dominates $\boldsymbol{W}$ since if we take $c_n = (\underline{H}(\boldsymbol{Z}) + S)/2$ the required condition is satisfied. The converse, however, is not true.

Theorem 3: If a source $\boldsymbol{Z}$ strictly dominates a channel $\boldsymbol{W}$ then $\boldsymbol{Z}$ is an approximating source for the channel $\boldsymbol{W}$.

Proof: If the source $\boldsymbol{Z}$ strictly dominates the channel $\boldsymbol{W}$, then for each input process $\boldsymbol{X}$ there exists a sequence $\{c_n\}_{n=1}^{\infty}$ and a $\delta > 0$ such that

$$P\left[\frac{1}{n}\, h_{Z^n}(Z^n) \le c_n + \delta\right] \le \tau_n \tag{1}$$

and

$$P\left[\frac{1}{n}\, i_{X^n W^n}(X^n;\, Y^n) \ge c_n\right] \le \tau_n \tag{2}$$

where $\tau_n \to 0$ as $n \to \infty$. Equation (1) implies that we can approximate any distribution with type less than $\exp\, n\left(c_n + \frac{2}{3}\,\delta\right)$ using the distribution Z^n. What we mean by type here is the following: A distribution is of type k if all probabilities that it assigns are integral multiples of $1/k$. To show this we use the procedure in the Aggregation Lemma [6].

Define

$$S^{(n)} = \{z^n \in F^n \colon P_{Z^n}(z^n) \le \exp^{-n(c_n+\delta)}\}.$$

Consider any M-type distribution P_M on $\{1,\, 2,\, \ldots,\, M\}$. We will place elements of $S^{(n)}$ in bin $B_n(i)$ until we have

$$P_{Z^n}(B_n(i)) > P_M(i) - \exp^{-n(c_n+\delta)}.$$

We stop either when we complete this process for all $i = 1,\, 2,\, \ldots,\, M$ or when we run out of sequences in the set $S^{(n)}$. All remaining sequences in F^n are placed in $B_n(1)$. At the end of this process we have

$$\sum_{i=1}^{M} |P_{Z^n}(B_n(i)) - P_M(i)| \le \max\left(2M \exp^{-n(c_n+\delta)},\, 2\tau_n + M \exp^{-n(c_n+\delta)}\right).$$

For any $M \le \exp\, n\left(c_n + \frac{2}{3}\,\delta\right)$ the right-hand side of the last equation goes to zero, since $\tau_n \to 0$.

We will now state and use a lemma the proof of which readily follows from the argument used in [3, Proof of Theorem 4].

Lemma 1: If

$$P\left[\frac{1}{n}\, i_{X^n W^n}(X^n;\, Y^n) \ge c_n\right] \le \tau_n$$

where $\tau_n \to 0$ as $n \to \infty$ then for any $\gamma > 0$ there exists a process $\tilde{\boldsymbol{X}}$ (producing output $\tilde{\boldsymbol{Y}}$) such that

$$\lim_{n\to\infty} d(Y^n,\, \tilde{Y}^n) = 0$$

and $\tilde{X}^n$ is an M-type distribution with $M \le \exp\, n(c_n + \gamma)$

Equation (2) and Lemma 1 imply that there exists a process $\tilde{X}^n$ with type smaller than $\exp\, n\left(c_n + \frac{2}{3}\,\delta\right)$ which approximates the output due to $\boldsymbol{X}$. We can approximate arbitrarily closely any distribution with type less than or equal to $\exp\, n\left(c_n + \frac{2}{3}\delta\right)$ (and hence $\tilde{X}^n$) using Z^n. This along with the fact that $d(PW,\, QW) \le d(P,\, Q)$ for any channel W and distributions $P,\, Q$ imply that $\boldsymbol{Z}$ can be used to approximate the output of the channel $\boldsymbol{W}$ when the input process is $\boldsymbol{X}$. Since we can find a sequence $\{c_n\}_{n=1}^{\infty}$ and a $\delta > 0$ which satisfy (1) and (2) for any input process $\boldsymbol{X}$, the source $\boldsymbol{Z}$ can approximate the output due to any process $\boldsymbol{X}$. □

IV. Converse

In this section we will give a converse result stating a condition under which the source will not be an approximating source for a channel in a certain sense. We would like to have a result completely analogous to the necessary condition in [5] for the source–channel separation problem, which would have been: If $\boldsymbol{Z}$ is an approximating source for a channel $\boldsymbol{W}$ then the source dominates the channel. The notion of a source dominating a channel would be defined (analogous to [5]) as follows.

Definition 5: A source $\boldsymbol{Z}$ dominates the channel $\boldsymbol{W}$ if for every process $\boldsymbol{X}$, for every $\delta > 0$, and for every sequence of nonnegative numbers $\{c_n\}_{n=1}^{\infty}$

$$\lim_{n\to\infty} P\left[\frac{1}{n}\, h_{Z^n}(Z^n) \le c_n - \delta\right] P\left[\frac{1}{n}\, i_{X^n W^n}(X^n;\, Y^n) \ge c_n\right] = 0.$$

It can be verified that if a source strictly dominates a channel then it dominates the channel. Also note that $\overline{H}(\boldsymbol{Z}) < S$ implies that the source does not dominate the channel.

We have not been able to show this statement that we set out to prove but we give a weaker statement that neither implies nor is implied by Theorem 2. The result is stated in contrapositive form and is weaker than the statement that we would like to make in two ways. First it involves the notion of a source being a strong approximating source for a channel and secondly, the necessary condition that we show for a source to be strongly approximating for a channel is weaker than the notion of a source dominating a channel.

We now give the stronger definition of an approximating source, following which we can state a necessary condition for a given source to be a strong approximating source for a given channel.

Definition 6: For any $\epsilon > 0$, the source $\boldsymbol{Z}$ with alphabet F is called a strong ϵ-approximating source for the channel $\boldsymbol{W}$ if for any arbitrary input source $\tilde{\boldsymbol{X}}$, there exists a sequence of deterministic mappings $\{\phi_n \colon F^n \longmapsto A^n\}$ such that $P_{\phi_n(Z^n)} \ll P_{\tilde{X}^n}$ and such that for sufficiently large n

$$d(Y^n,\, \tilde{Y}^n) < \epsilon$$

where Y^n and $\tilde{Y}^n$ are the outputs of the channel due to $\phi(Z^n)$ and $\tilde{X}^n$, respectively.

We call $\boldsymbol{Z}$ a strong approximating source for $\boldsymbol{W}$ if it is a strong ϵ-approximating source for $\boldsymbol{W}$ for all $\epsilon > 0$.

Note that the only difference between Definitions 3 and 6 is that the latter places an extra constraint that the deterministic transformation can only map to those sequences which have nonzero probability under the source whose output we are trying to approximate.

The following theorem states precisely a necessary condition for the source to be a strong approximating source for a channel. We note that in (4) below if we did not have β_n going to 1 but just being strictly bigger than 0 then this condition would be the negation of a source dominating a channel.

Theorem 4: Suppose that for some process $\boldsymbol{X}$, for some $\alpha,\, \delta > 0$, and for some sequence of nonnegative numbers $\{c_n\}_{n=1}^{\infty}$

$$P\left[\frac{1}{n}\, h_{Z^n}(Z^n) \le c_n - \delta\right] > \alpha \tag{3}$$

and

$$P\left[\frac{1}{n}\, i_{X^n W^n}(X^n;\, Y^n) \ge c_n\right] > \beta_n \tag{4}$$

for all $n \in I$ where I is an infinite set of integers and $\beta_n \to 1$ as $n \to \infty$. Then the source $\boldsymbol{Z}$ is not a strong approximating source for the channel $\boldsymbol{W}$.

Proof: Equation (4) along with Feinstein's lemma [2] implies that there exists a code of length n with $\exp\, n\left(c_n - \frac{\delta}{10}\right)$ codewords and with maximal probability of error less than ϵ_n for $n \in I$ where $\epsilon_n \to 0$ as $n \to \infty$. Consider a good code with $M = \exp\, n\left(c_n - \frac{\delta}{10}\right)$ codewords. Let the codewords be $\{b_i\}_{i=1}^{M}$ and their corresponding decoding sets be $\{D_i\}_{i=1}^M$. Consider now the process $\tilde{X}^n$ which places mass $\frac{1}{M}$ on each of the M codewords, $\{b_i\}_{i=1}^M$. We will show that the output due to this process cannot be approximated well with our source.

Consider the set

$$S^n = \left\{z^n \colon \tfrac{1}{n} h_{Z^n}(z^n) \le c_n - \delta\right\}.$$

By (3), $P_{Z^n}(S^n) > \alpha$ for all $n \in I$. Consider any deterministic mapping ϕ_n (from F^n to A^n) and the N codewords to which the sequences in S^n get mapped. Assume the codewords are numbered so that these codewords are $\{b_i\}_{i=1}^N$. Since $|S^n| \le \exp n(c_n - \delta)$, we have $N \le \exp n(c_n - \delta)$.

Let

$$B = \bigcup_{i=1}^N D_i$$

and $\tilde{Y}^n$ be the output due to $\tilde{X}^n$.

$$\begin{aligned} P_{\tilde{Y}^n}(B) &= \frac{1}{M}\sum_{i=1}^N P(B|b_i) + \frac{1}{M}\sum_{i=N+1}^M P(B|b_i) \\ &\le \frac{N}{M} + \frac{\epsilon_n}{M}(M-N) \\ &\le 2\epsilon_n \end{aligned}$$

where the last inequality holds for sufficiently large $n \in I$. Let $\boldsymbol{X}$ be $\phi(\boldsymbol{Z})$ and let $\boldsymbol{Y}$ be the output process with $\boldsymbol{X}$ as the input to the channel

$$\begin{aligned} P_{Y^n}(B) &= \sum_{i=1}^M P(B|b_i)P_{X^n}(b_i) \\ &\ge \sum_{i=1}^N P(D_i|b_i)P_{X^n}(b_i) \\ &\ge (1-\epsilon_n)P_{Z^n}(S^n) \\ &\ge (1-\epsilon_n)\alpha. \end{aligned}$$

Thus we have

$$P_{\tilde{Y}^n}(B) - P_{Y^n}(B) \ge (1-\epsilon_n)\alpha - 2\epsilon_n$$

for all sufficiently large $n \in I$. We note that the right-hand side of the equation above does not go to zero as n increases and so $\boldsymbol{Z}$ cannot be an approximating source in the strong sense for the channel $\boldsymbol{W}$. $\square$

V. A Special Case

We will now look at the special case of a full-rank, discrete, memoryless channel (FRDMC). A channel $\boldsymbol{W}$ is of full rank if the transition vectors $\{W(.|a)\}_{a\in A}$ are linearly independent. For the FRDMC we will be able to show a result analogous to [5], that we would liked to show for general channels.

Theorem 5: For a FRDMC, if the source $\boldsymbol{Z}$ is an approximating source for the channel $\boldsymbol{W}$ then the source $\boldsymbol{Z}$ dominates the channel $\boldsymbol{W}$.

Proof: We will show that if the source $\boldsymbol{Z}$ does not dominate the FRDMC $\boldsymbol{W}$ then $\boldsymbol{Z}$ is not an approximating source for the channel $\boldsymbol{W}$.

At the outset we mention why we need the assumption of full rank. We show that if source $\boldsymbol{Z}$ does not dominate the channel $\boldsymbol{W}$, the source cannot approximate the output due to a particular independent and identically distributed (i.i.d.) input X. The reason we need the full rank assumption is that to approximate the output of an i.i.d. input to an FRDMC it is necessary to place mass on the typical sequences of X. This is not true if the channel is not full-rank.

If a source $\boldsymbol{Z}$ does not dominate $\boldsymbol{W}$ then there is a process $\overline{\boldsymbol{X}}$ such that for some $\alpha,\, \delta > 0$ and for some sequence of nonnegative numbers $\{c_n\}_{n=1}^\infty$

$$P\left[\frac{1}{n} h_{Z^n}(Z^n) \le c_n - \delta\right] > \alpha \tag{5}$$

and

$$P\left[\frac{1}{n} i_{\overline{X}^n W^n}\left(\overline{X}^n; \overline{Y}^n\right) \ge c_n\right] > \alpha \tag{6}$$

for all $n \in I$ where I is an infinite set of integers. From [4, Corollary 2] we have that for an FRDMC $S = C$. But for a DMC, $C = \sup_X I(X;Y)$. Since $S = \sup_{\boldsymbol{X}} \overline{I}(\boldsymbol{X};\boldsymbol{Y})$ we have for a FRDMC

$$\sup_{\boldsymbol{X}} \overline{I}(\boldsymbol{X};\boldsymbol{Y}) = \sup_X I(X;Y).$$

Thus for an FRDMC if there is a process $\overline{\boldsymbol{X}}$ such that (5) and (6) are satisfied then there exists an i.i.d. process $\boldsymbol{X}$ such that

$$c_n \le I(X;Y) + \frac{\delta}{10} \tag{7}$$

for sufficiently large $n \in I$. We will show that the source $\boldsymbol{Z}$ cannot approximate the output due to process $\boldsymbol{X}$.

Define the set of sequences in A^n that are not γ-typical ($\gamma > 0$) by

$$D_{X^n}(\gamma) = \left\{x^n \colon \left|\frac{1}{n}N(a|x^n) - P_X(a)\right| > \gamma \text{ for some } a \in A\right\}$$

where $N(a|x^n)$ denotes the number of times that a occurs in the sequence x^n.

Also define the set of sequences jointly γ-typical with x^n by

$$T_W^n(x^n,\gamma) = \left\{y^n \colon \left|\frac{1}{n}N(a,b|x^n,y^n) - \frac{1}{n}N(a|x^n)W(b|a)\right| \le \gamma \text{ for all } (a,b) \in A\times B\right\}$$

where $N(a,b|x^n,y^n)$ denotes the number of times that (a,b) occurs in (x^n,y^n).

Let W denote the matrix whose columns are the transition vectors $\{W(.|a)\}_{a\in A}$. Let $\boldsymbol{r}$ be an $|A|$-dimensional vector and let $\boldsymbol{s} = W\boldsymbol{r}$. Since W is of full rank if $|r_i| > \gamma$ for some $i \in \{1,2,\ldots,|A|\}$ then $|s_j| > k\gamma$ for some $j \in \{1,2,\ldots,|B|\}$ and some $k > 0$ independent of $\boldsymbol{r}$. Thus we have that if $x^n \in D_X^n(\gamma)$ then $W\boldsymbol{N}(x^n)/n$ differs from $W\boldsymbol{P}_X$ by $k\gamma$ in at least one component, where $\boldsymbol{N}(x^n)$, $\boldsymbol{P}_X$ are column vectors consisting of $N(a|x^n)$, $P_X(a)$, respectively, for $a \in A$. Now if $y^n \in T_W^n(x^n,\gamma)$ then

$$\left|\frac{1}{n}N(b|y^n) - \sum_{a\in A}\frac{1}{n}N(a|x^n)W(b|a)\right| \le |A|\gamma$$

for all $b \in B$. Thus we have that if

$$x^n \in D_X^n\left(\frac{\gamma}{k}\right) \quad \text{and} \quad y^n \in T_W^n\left(x^n, \frac{\gamma}{3|A|}\right)$$

then $\boldsymbol{N}(y^n)/n$ differs from $\boldsymbol{P}_Y$ in some component by at least $\frac{2\gamma}{3}$. Thus $y^n \in D_Y^n\left(\frac{2\gamma}{3}\right)$. We have shown that if $x^n \in D_X^n\left(\frac{\gamma}{k}\right)$ then

$$T_W^n\left(x^n, \frac{\gamma}{3|A|}\right) \subseteq D_Y^n\left(\frac{2\gamma}{3}\right).$$

But

$$P_{Y^n}\left(D_Y^n\left(\frac{2\gamma}{3}\right)\right) \to 0, \qquad \text{as } n \to \infty$$

and

$$W^n\left(T_W^n\left(x^n, \frac{\gamma}{3|A|}\right)|x^n\right) \to 1, \qquad \text{as } n \to \infty$$

at a rate independent of x^n (from [1, Lemma 2.12]). So if the input $\phi^n(Z^n)$ approximates the output due to X^n then we must have

$$P_{Z^n}\left(\phi^n(Z^n) \in D_X^n\left(\frac{\gamma}{k}\right)\right) \to 0, \qquad \text{as } n \to \infty$$

since if this were not so

$$P_{\phi^n(Z^n)}W^n\left(D_Y^n\left(\frac{2\gamma}{3}\right)\right) - P_{Y^n}\left(D_Y^n\left(\frac{2\gamma}{3}\right)\right) > \beta$$

infinitely often, for some $\beta > 0$. This would imply that the input $\phi^n(Z^n)$ does not approximate the output due to X^n.

Now by a slight modification of [1, Lemma 2.13]

$$\left|\frac{1}{n}\log|T_W^n(x^n, \gamma/3|A|)| - H(Y|X)\right| \le \epsilon(\gamma)$$

for every $x^n \in D_{X^n}^c\left(\frac{\gamma}{k}\right)$ where $\epsilon(\gamma)$ is continuous in γ, independent of x^n and $\epsilon(\gamma) \to 0$ as $\gamma \to 0$. Define

$$Q^n = \left\{z^n: \frac{1}{n}h_{Z^n}(z^n) \le c_n - \delta\right\}.$$

Clearly, $|Q^n| \le \exp(n(c_n - \delta))$. Also define R^n as the image of Q^n under the mapping ϕ^n and

$$S^n = \bigcup_{x^n \in R^n \cap D_{X^n}^c\left(\frac{\gamma}{k}\right)} T_W^n\left(x^n, \frac{\gamma}{3|A|}\right).$$

Then we have

$$|S^n| \le \exp\, n(c_n - \delta)\, \exp\, n(H(Y|X) + \epsilon(\gamma))$$

and $P_{\phi^n(Z^n)}W^n(S^n) \ge \frac{\alpha}{2}$ for sufficiently large $n \in I$. This is because

$$W^n\left(T_W^n\left(x^n, \frac{\gamma}{3|A|}\right)|x^n\right) \to 1$$

as $n \to \infty$ at a rate independent of x^n

$$P_{Z^n}\left[\phi^n(Z^n) \in D_{X^n}^c\left(\frac{\gamma}{k}\right)\right] \to 1$$

as $n \to \infty$ and

$$P_{Z^n}[\phi^n(Z^n) \in R^n] > \alpha$$

for all $n \in I$. Using (7) we can upper-bound $|S^n|$ by

$$|S^n| \le \exp\, n\left(H(Y) - \frac{9\delta}{10} + \epsilon(\gamma)\right) \le \exp\, n\left(H(Y) - \frac{\delta}{2}\right)$$

for sufficiently large $n \in I$, where the second inequality holds if we choose $\gamma > 0$ sufficiently small. We have $P_{Y^n}(S^n) \to 0$ as $n \to \infty$ (by the strong source coding theorem for i.i.d. sources). For any deterministic mapping ϕ^n we can find a set $S^n \subseteq B^n$ such that $P_{\phi^n(Z^n)}W^n(S^n) \ge \frac{\alpha}{2}$ for sufficiently large $n \in I$ and $P_{Y^n}(S^n) \to 0$ as $n \to \infty$. Hence, $\boldsymbol{Z}$ cannot be an approximating source for the FRDMC $\boldsymbol{W}$. □

We note that Theorem 5 implies that if $\underline{H}(\boldsymbol{Z}) < S$ then the source $\boldsymbol{Z}$ is not an approximating source for a FRDMC channel with resolvability S.

References

[1] I. Csiszár and Körner, *Information Theory: Coding Theorems for Discrete Memoryless Systems*. New York: Academic, 1981.

[2] A. Feinstein, "A new basic theorem in information theory," *IRE Trans. Inform. Theory*, vol. IT-4, pp. 2–22, Jan. 1954.

[3] T. S. Han and S. Verdú, "Approximation theory of output statistics," *IEEE Trans. Inform. Theory*, vol. 39, pp. 752–772, May 1993.

[4] ——, "Spectrum invariancy under output approximation for full rank discrete memoryless channels" (in Russian), *Probl. Pered. Inform.*, no. 2, pp. 101–118, 1993.

[5] S. Vembu, S. Verdú, and Y. Steinberg, "The source-channel separation theorem revisited," *IEEE Trans. Inform. Theory*, vol. 41, pp. 44–54, Jan. 1995.

[6] S. Vembu and S. Verdú, "Generating random bits from an arbitrary source: Fundamental limits," *IEEE Trans. Inform. Theory*, vol. 41, pp. 1322–1332, Sept. 1995.