The Super-Trellis Structure of Turbo Codes

M. Breiling, L. Hanzo
Dept. of Electr. and Comp. Sc., Univ. of Southampton, SO17 1BJ, UK.
Tel: +44-703-593 125, Fax:+44-703-594 508
Email: lh@ecs.soton.ac.uk
http://www-mobile.ecs.soton.ac.uk

Submitted to the IEEE Tranactions on Information Theory on 14/02/1997,
published in the IEEE Transactions on Information Theory, No. 6, 2000, pp. 2212-2228

The financial support of the EPSRC, UK 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 is gratefully acknowledged. Our sincere thanks are also due to the anonymous reviewers and to Jozef Hamorsky, Lorenzo Piazzo and Andreas Knickenberg for proofreading the manuscript.

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 sub-optimum. 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.

Indexing Terms: turbo codes, iterative decoding, code trellis, MLSE decoding.

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 (BER) [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 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 feed-back 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 (SNR). Further contributions in the previous 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 been block component codes also proposed in the literature.

The outline of the paper is as follows. Section II presents a model of the turbo encoder that is used in this contribution. 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,

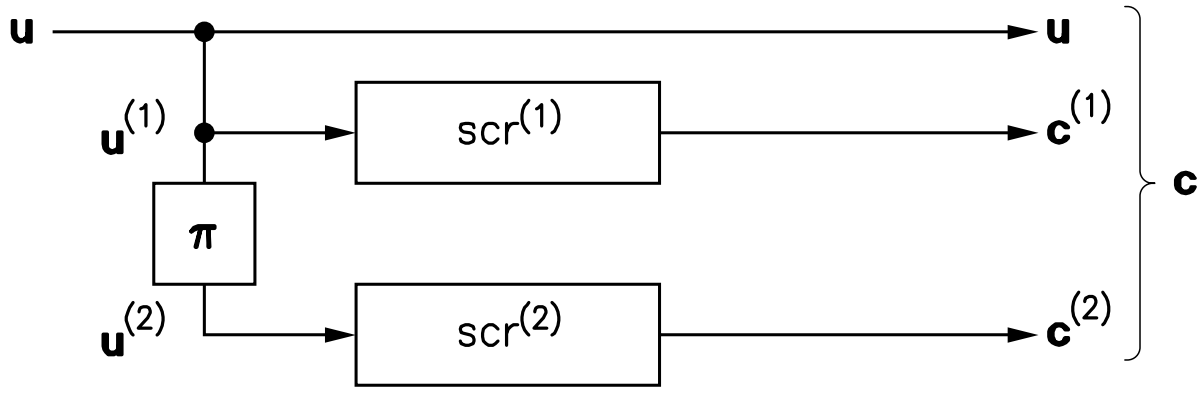

Fig. 1. System model of the Turbo encoder

Section VI discusses the results obtained, which is followed by our conclusions in Section VII.

II. System model and terminology

Figure 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 $\mathbf{c} = (\mathbf{u}; \mathbf{c}^{(1)}; \mathbf{c}^{(2)})$. The vector of K binary encoder input symbols used for generating a codeword is referred to as the *information word* $\mathbf{u} = (u_1; \ldots; u_K)$. This information word $\mathbf{u}$ 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 $\mathtt{scr}^{(1)}$ and $\mathtt{scr}^{(2)}$, which are also often referred to as component encoders, processing the information symbols in order to generate the component codewords $\mathbf{c}^{(1)}$ and $\mathbf{c}^{(2)}$ constituting the parity part of the codeword $\mathbf{c}$. Throughout the paper, we use the following terminology. Capitals 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 $\mathrm{scr}^{(1)}$ is fed with the information symbols $\mathbf{u}^{(1)} = \mathbf{u}$ obeying their original ordering. The second component scrambler $\mathrm{scr}^{(2)}$ is fed with a *permuted* version $\mathbf{u}^{(2)}$ 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 $\mathbf{u}^{(1)} = \mathbf{u}$ is achieved by permuting the vector elements generating $\mathbf{u}^{(2)}$, 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 feed-back branch as exemplified in Figure 2. This structure is also often referred to as a recursive systematic convolutional (RSC) code. Any such **s**crambler 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

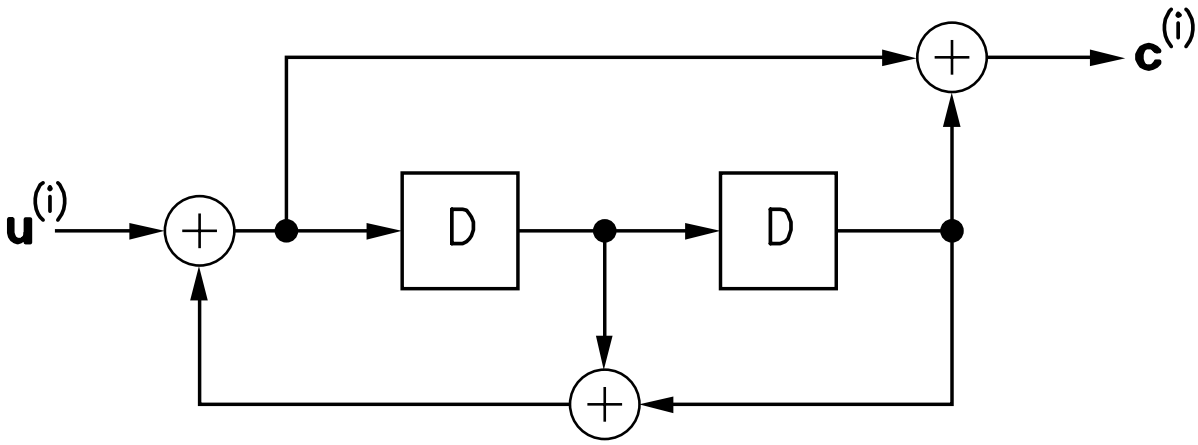

Fig. 2. Example of a component scrambler or component encoder with feed-back and feed-forward polynomials expressed in octal form as 7 and 5, respectively

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 $\gamma_\mathrm{s}: \ (\mathcal{S}_\mathrm{s} \times \mathcal{U}_\mathrm{s}) \rightarrow (\mathcal{S}_\mathrm{s} \times \mathcal{C}_\mathrm{s})$ represents the state transitions, where for every pair (s, u) with $s \in \mathcal{S}_\mathrm{s}$ (predecessor state) and $u \in \mathcal{U}_\mathrm{s}$ (scrambler input) a pair (s', c) with $s' \in \mathcal{S}_\mathrm{s}$ (successor state) and $c \in \mathcal{C}_\mathrm{s}$ (scrambler output) is specified. A FSM can be graphically characterised by its state transition diagram or, if we also incorporate the elapse of time, by its trellis. Figure 3 displays the trellis of the scrambler of Figure 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 a 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 Figure 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

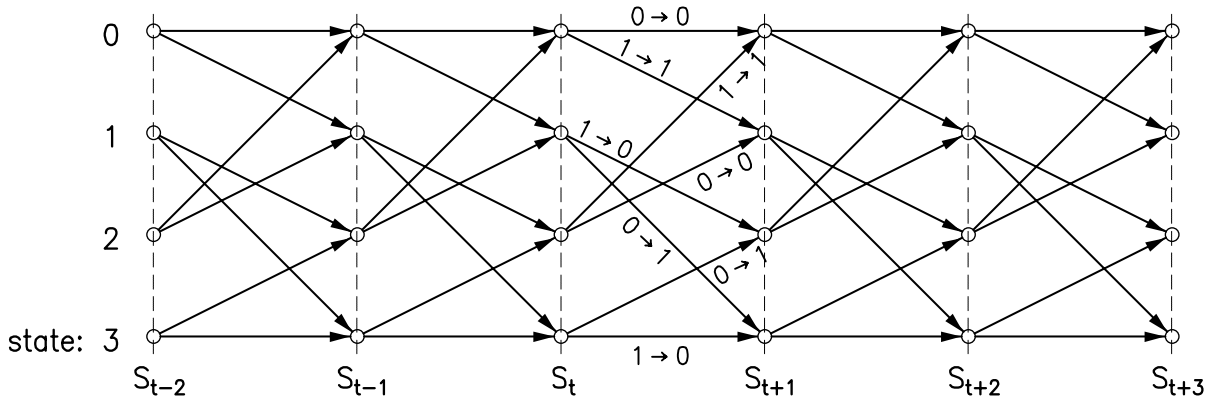

Fig. 3. Trellis of the scrambler of Figure 2 with memory M=2

FSM and its post-history in the following sense:

If a sequence of FSM states $\mathbf{S}_{t-} = (\ldots; S_{t-2}; S_{t-1}; S_t = s),\ s \in \mathcal{S}_\mathrm{s}$ corresponds to a valid sequence of state transitions constituting the pre-history with respect to time t and a sequence $\mathbf{S}_{t+} = (S_t = s; S_{t+1}; S_{t+2}; \ldots)$ corresponds to a valid sequence of state transitions constituting the post-history, then we can combine the sequences, and $\mathbf{S} = (\ldots; S_{t-2}; S_{t-1}; S_t = s; S_{t+1}; S_{t+2}; \ldots)$ 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 $\mathbf{S}_{t+} = (S_t = \tilde{s}; S_{t+1}; S_{t+2}; \ldots)$ 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 modelling all of its four constituent elements in Figure 1 by their respective FSMs and combining these. For the two component scramblers of Figure 1 it is relatively straightforward to find the corresponding FSMs (see above). For the systematic branch of the encoder in Figure 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 $\mathbf{u}^{(1)}$ must have been inserted into

the interleaver, before the vector of output symbols $\mathbf{u}^{(2)}$ can 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 that is 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 sub-optimum 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 Figure 2 and a simple two-column interleaver of total interleaving length of K bits obeying the following mapping:

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

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 $U_1 \dots U_K$, 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 it will be shown during our further discourse. Let us consider the positions of the first five input symbols, namely that of $U_1 \dots U_5$ in both component trellises, which is illustrated in Figure 4. Entering U_5 in the turbo encoder's FSM corresponds to $t = 5$.

In the upper component trellis in Figure 4 the symbols $U_1^{(1)} \dots U_5^{(1)}$ are in consecutive positions. The transitions $\mathbf{S}_{5-}^{(1)} = (S_0^{(1)} = 0; S_1^{(1)}; \dots; S_5^{(1)})$ 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 $S_0^{(1)} = 0$.) 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 $U_6^{(1)} \dots U_K^{(1)}$ and from the corresponding state transitions $\mathbf{S}_{5+}^{(1)} = (S_5^{(1)}; S_6^{(1)}; \dots)$. 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 Figure 1 in the second component trellis of Figure 4, the same five bits $U_1 \dots U_5$ are transformed into the input symbols $U_1^{(2)}, U_2^{(2)}, U_3^{(2)}$ and

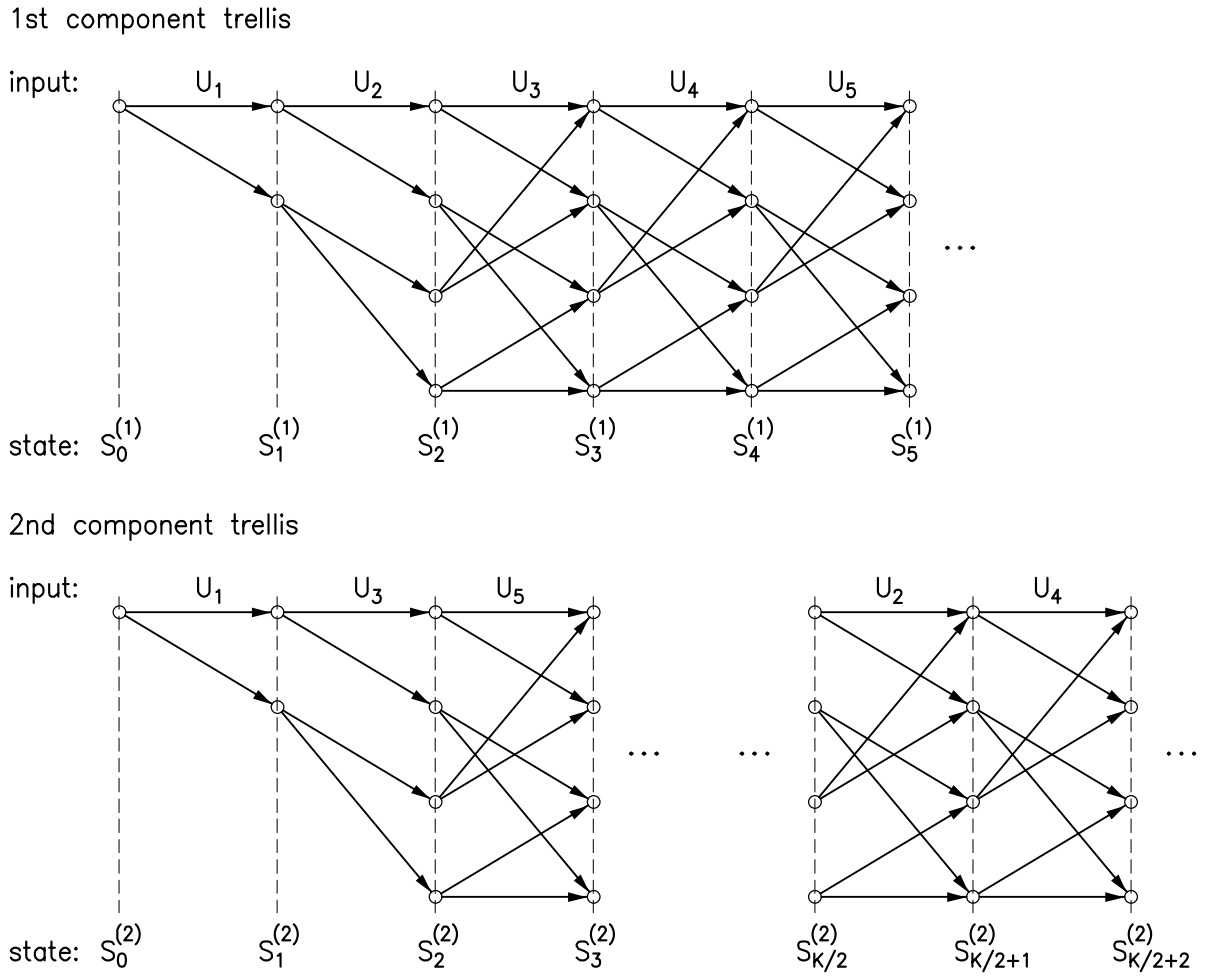

Fig. 4. Trellis transitions in the two component scrambler trellises up to time instant $t = 5$

$U^{(2)}_{K/2+1}, U^{(2)}_{K/2+2}$. Hence, the corresponding transitions are located in the 2nd component trellis seen at the bottom of Figure 4 in two different sections. Consequently, the pre-history *wrt* $t = 5$ - i.e. *wrt* bits $U_1 \dots U_5$, which belong to the first five transitions of the turbo code super–trellis associated with the discrete time instants of $t = 0 \dots 5$ - corresponds to the transitions of the 2nd component trellis in these two distinct trellis sections of Figure 4. The post–history of the 2nd component trellis wrt $t = 5$ is constituted by the two remaining sections of the trellis, which are indicated by dots in Figure 4. The interface between the pre– and post–history, which is associated with $t = 5$, is formed in the 2nd component trellis by the set $(S^{(2)}_3; S^{(2)}_{K/2}; S^{(2)}_{K/2+2})$, which we will refer to as the 'interface' states in the 2nd component trellis.

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

$$(U_1; U_2; U_3; U_4; U_5) = (U^{(2)}_1; U^{(2)}_{K/2+1}; U^{(2)}_2; U^{(2)}_{K/2+2}; U^{(2)}_3)$$

where the corresponding state–transitions in both sections of the 2nd component trellis at the bottom of

Figure 4 – which constitute the pre–history related to $t = 5$ – are represented by:

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

Every valid post–history

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

can therefore be combined with $\mathbf{S}_{5-}^{(2)}$ and forms a valid sequence of state transitions for the 2nd 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 $\mathbf{S}_{5-}^{(2)}$ and $\mathbf{S}_{5+}^{(2)}$.

So far we have elaborated on the details of the pre–history related to $t = 5$ in the 1st and 2nd 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 $\mathbf{S}_{5-}^{*} = (\mathbf{S}_{5-}^{(1)}; \mathbf{S}_{5-}^{(2)})$ can be combined with every valid post–history pair $\mathbf{S}_{5+}^{*} = (\mathbf{S}_{5+}^{(1)}; \mathbf{S}_{5+}^{(2)})$, 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 $\mathbf{S}_{5-}^{*}$ are identical to those in $\mathbf{S}_{5+}^{*}$, 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:

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

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 1st component scrambler we have $U_6 = U_6^{(1)}$, such that the transition associated with U_6 in the 1st component trellis is directly adjacent to the transitions in the pre–history $\mathbf{S}_{5-}^{(1)}$ (upper trellis in Figure 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 1st component code simply extends from $\mathbf{S}_{5-}^{(1)}$ to $\mathbf{S}_{6-}^{(1)} = (S_0^{(1)} = 0; \ldots; S_5^{(1)} = s_5^{(1)}; S_6^{(1)} = s_6^{(1)})$.

With the aid of Figure 4 we can see that for the input symbols of the 2nd component scrambler the relation $U_6 = U_{K/2+3}^{(2)}$ holds. In Figure 4 the corresponding transition (index $K/2 + 3$ of the 2nd 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 2nd component code becomes $\mathbf{S}_{6-}^{(2)} = (S_0^{(2)} = 0; \ldots; S_3^{(2)} = s_3^{(2)}; S_{K/2}^{(2)}; \ldots; S_{K/2+2}^{(2)} = s_{K/2+2}^{(2)}; S_{K/2+3}^{(2)} = s_{K/2+3}^{(2)})$. From this we can easily infer the interfaces of the

pair $\mathbf{S}^*_{6-} = (\mathbf{S}^{(1)}_{6-}; \mathbf{S}^{(2)}_{6-})$, 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 $S^*_5 = (s^{(1)}_5; s^{(2)}_3; s^{(2)}_{K/2}; s^{(2)}_{K/2+2})$ yielding:

$$S^*_6 \stackrel{\text{def}}{=} (S^{(1)}_6; S^{(2)}_3; S^{(2)}_{K/2}; S^{(2)}_{K/2+3}) = (s^{(1)}_6; s^{(2)}_3; s^{(2)}_{K/2}; s^{(2)}_{K/2+3}),$$

where both the state transitions $(s^{(1)}_5; s^{(1)}_6)$ and $(s^{(2)}_{K/2+2}; s^{(2)}_{K/2+3})$ are associated with the value u_6 of the input symbol U_6.

C. Generalised Definition of the Turbo Encoder Super–States

As an extension of our previous introductory elaborations in this section we will present a generalised definition of the super–state transitions in the super–trellis, which also defines implicitely 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 indeces $\mathcal{I}^{(2)}_{\text{pre}}(t)$, which contains the number of transitions belonging to the pre–history of the 2nd component trellis *wrt* time instant t. If and only if an index obeys $i \in \mathcal{I}^{(2)}_{\text{pre}}(t)$, the transition with this index is part of the pre–history of the 2nd component trellis at time t. It follows immediately that a state $S^{(2)}_j$ with index j is an interface state in the 2nd 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: $j \in \mathcal{I}^{(2)}_{\text{pre}}(t) \quad \wedge \quad (j+1) \notin \mathcal{I}^{(2)}_{\text{pre}}(t)$ **or** $j \notin \mathcal{I}^{(2)}_{\text{pre}}(t) \quad \wedge \quad (j+1) \in \mathcal{I}^{(2)}_{\text{pre}}(t)$.

For an arbitrary interleaver $\boldsymbol{\pi}$, which was shown in Figure 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 $S^{(1)}_0 = S^{(2)}_0 = 0$. Hence the following equation holds for the super-state:

$$S^*_0 \stackrel{\text{def}}{=} (S^{(1)}_0; S^{(2)}_0) = (0; 0). \tag{1}$$

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

Evolution from time instant t to $t+1$: the previous super–state s^*_t is assumed to be known. For the 1st component code the relation $U_{t+1} = U^{(1)}_{t+1}$ holds, hence the input symbol $U_{t+1} = u_{t+1}$ specifies the transition $(S^{(1)}_t; S^{(1)}_{t+1}) = (s^{(1)}_t; s^{(1)}_{t+1})$ in the 1st component code. (In the 1st component trellis at time instant t the state with index t, $S^{(1)}_t$ is always the interface state contained in the super–state S^*_t, and hence the value $s^{(1)}_t$ is known from the vector s^*_t.) For the 2nd component code we have $U_{t+1} = U^{(2)}_j$ with $j = \pi_{t+1}$. The index of the associated transition in the 2nd component trellis for $t+1$ is j, such that the corresponding transition becomes $(S^{(2)}_{j-1}; S^{(2)}_j)$. This transition is not part of the pre–history *wrt* t and hence $j \notin \mathcal{I}^{(2)}_{\text{pre}}(t)$. Updating the set of indeces means that $\mathcal{I}^{(2)}_{\text{pre}}(t+1) = \mathcal{I}^{(2)}_{\text{pre}}(t) \cup \{j\}$. In order to proceed from $(j-1)$ to j, we have to distinguish between four different cases or scenarios, as far as

the 2nd 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: $j-1 \in \mathcal{I}^{(2)}_{\rm pre}(t)$ and $j+1 \notin \mathcal{I}^{(2)}_{\rm pre}(t)$. 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 2nd component trellis already belonging to the pre–history in the 2nd 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 Figure 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 1st 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 2nd component trellis a section is extended to the right in Figure 4. Again, these issues will be exemplified in Section III-D and Fig. 6, where this scenario is encountered for transitions $t=0 \rightarrow 1$, $t=2 \rightarrow 3$, $t=3 \rightarrow 4$, $t=4 \rightarrow 5$ and $t=7 \rightarrow 8$.

(2) The condition for encountering our second scenario is: $j-1 \notin \mathcal{I}^{(2)}_{\rm pre}(t)$ and $j+1 \in \mathcal{I}^{(2)}_{\rm pre}(t)$. 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 2nd 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 1st component code by the new $S^{(1)}_{t+1}$ and also the old interface $S^{(2)}_j$ in the 2nd 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 2nd trellis is, when $j-1 \notin \mathcal{I}^{(2)}_{\rm pre}(t)$ and $j+1 \notin \mathcal{I}^{(2)}_{\rm pre}(t)$. 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 2nd 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 2nd component trellis. Observe, however

that neither the value $s_{j-1}^{(2)}$ of $S_{j-1}^{(2)}$ nor the value $s_j^{(2)}$ of $S_j^{(2)}$ is specified by $S_t^* = s_t^*$, which represents the interface of the pre–history *wrt* t in the super–trellis.

The pair $(S_{j-1}^{(2)}; S_j^{(2)})$ can therefore assume all legitimate values $(s_{j-1}^{(2)}; s_j^{(2)})$ within the vector S_{t+1}^*, which represent valid state transitions and which correspond to the input symbol $U_j^{(2)} = u_{t+1}$. 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 $S_{j-1}^{(2)}$ – which is contained in the associated super-state vector – can assume all possible states $S_{j-1}^{(2)} \in \mathcal{S}_s$ and the other new interface state $S_j^{(2)}$ will be $s_j^{(2)}$, determined by the transition $(s_{j-1}^{(2)}; s_j^{(2)})$ due to the input symbol $U_j^{(2)} = u_{t+1}$. We will refer to this scenario as the *opening* of a new section in the 2nd component trellis. These aspects will be revisited in Section III-D and Fig. 6, where the transition $t = 1 \rightarrow 2$ corresponds to this specific scenario.

(4) The last possible scenario encountered by the 2nd trellis is, when $j - 1 \in \mathcal{I}_{\text{pre}}^{(2)}(t)$ and $j + 1 \in \mathcal{I}_{\text{pre}}^{(2)}(t)$. The state associated with the index $j - 1$, i.e. $S_{j-1}^{(2)}$, as well as the one with the index j, i.e. $S_j^{(2)}$, are contained in the vector corresponding to super-state S_t^*. The specific state values $s_{j-1}^{(2)}$ and $s_j^{(2)}$ representing the interfaces $S_{j-1}^{(2)}$ and $S_j^{(2)}$ 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 $(S_{j-1}^{(2)}; S_j^{(2)})$ – which is the transition at time instant $t + 1$ in the 2nd 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 $s_{j-1}^{(2)}$ and $s_j^{(2)}$, respectively. We will revisit these issues in the context of Section III-D and Fig. 6, where the transition $t = 6 \rightarrow 7$ constitutes an example of scenario (4).

If this fixed transition $(s_{j-1}^{(2)}; s_j^{(2)})$ is legitimate and if it is associated with the value u_{t+1} of the most recent input symbol $U_j^{(2)}$, then two interface states are connected by means of this particular transition in the 2nd 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 $S_{j-1}^{(2)}$ and $S_j^{(2)}$ and by substituting $S_t^{(1)}$ by $S_{t+1}^{(1)}$.

However, it may happen that for the interface states $s_{j-1}^{(2)}$ and $s_j^{(2)}$ – which are determined by $S_t^* = s_t^*$ – the transition $(s_{j-1}^{(2)}; s_j^{(2)})$ is illegitimate, since this state transition is non-existent 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 $(s_{j-1}^{(2)}; s_j^{(2)})$ 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 2nd 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 $\|\mathcal{S}_s\| = 2^M$ 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 $s_{j-1}^{(2)} \in \mathcal{S}_s$, which is determined by the super-state value s_t^*, and for any input symbol value $u_j^{(2)} \in \mathcal{U}_s$, there exists only *one* of 2^M possible interface state values $s_j^{(2)} \in \mathcal{S}_s$, such that the component scrambler transition $(s_{j-1}^{(2)}; s_j^{(2)})$ is associated with the input symbol $u_j^{(2)}$.) We will further augment this concept by means of an example in Section D. Note that the trellis section in the 1st component trellis with its interface state $S_t^{(1)}$ is in all four above scenarios extended to the right, i.e. the new interface state is $S_{t+1}^{(1)}$.

Let us now elaborate a little further. The output symbols of the turbo encoder FSM – which are associated with a super–state transition at 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 $\mathbb{F}_{\mathrm{T}}$ at time instant $t+1$ is therefore the vector

$$\bar{C}_{t+1} = (U_{t+1}; C_{t+1}^{(1)}; C_{\pi_{t+1}}^{(2)}),$$

and hence the relation $\mathcal{C}_{\mathrm{T}} = [\{0;1\}]^3$ 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 $\bar{C}$ 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 Figure 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 Figure 6. The index $j = \pi_{t+1}$ denotes in the 2nd 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 Figure 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 $\mathcal{I}_{\mathrm{pre}}^{(2)}(0) = 0$.

$t = 0 \rightarrow 1$*:* We have $j = \pi_{t+1} = \pi_1 = 1$, since U_1 is at position 1 at the bottom of Table I. As $j - 1 = 0 \in \mathcal{I}_{\mathrm{pre}}^{(2)}(0)$ and $j + 1 = 2 \notin \mathcal{I}_{\mathrm{pre}}^{(2)}(0)$, this corresponds to scenario (1) described in Subsection III-C, i.e. the *right-extension* of an existing section in the 2nd component trellis ($S_{j-1}^{(2)} = S_0^{(2)}$ is one of the

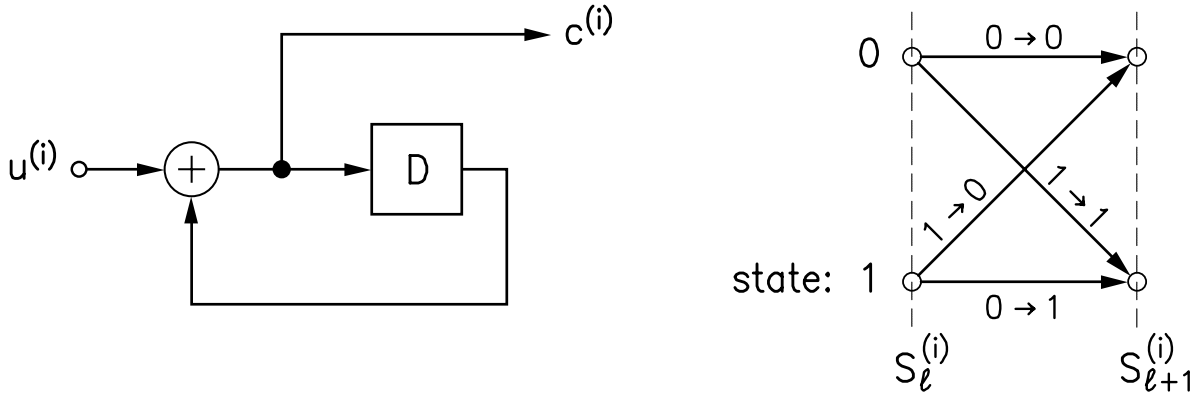

Fig. 5. Simple component scrambler and an associated trellis segments

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

TABLE I

Interleaving scheme for the component trellises of Example 2

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.

$t = 1 \rightarrow 2$*:* 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.

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

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

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

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

$t = 6 \rightarrow 7$*:* $j = \pi_{t+1} = \pi_7 = 4$, 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 2nd 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

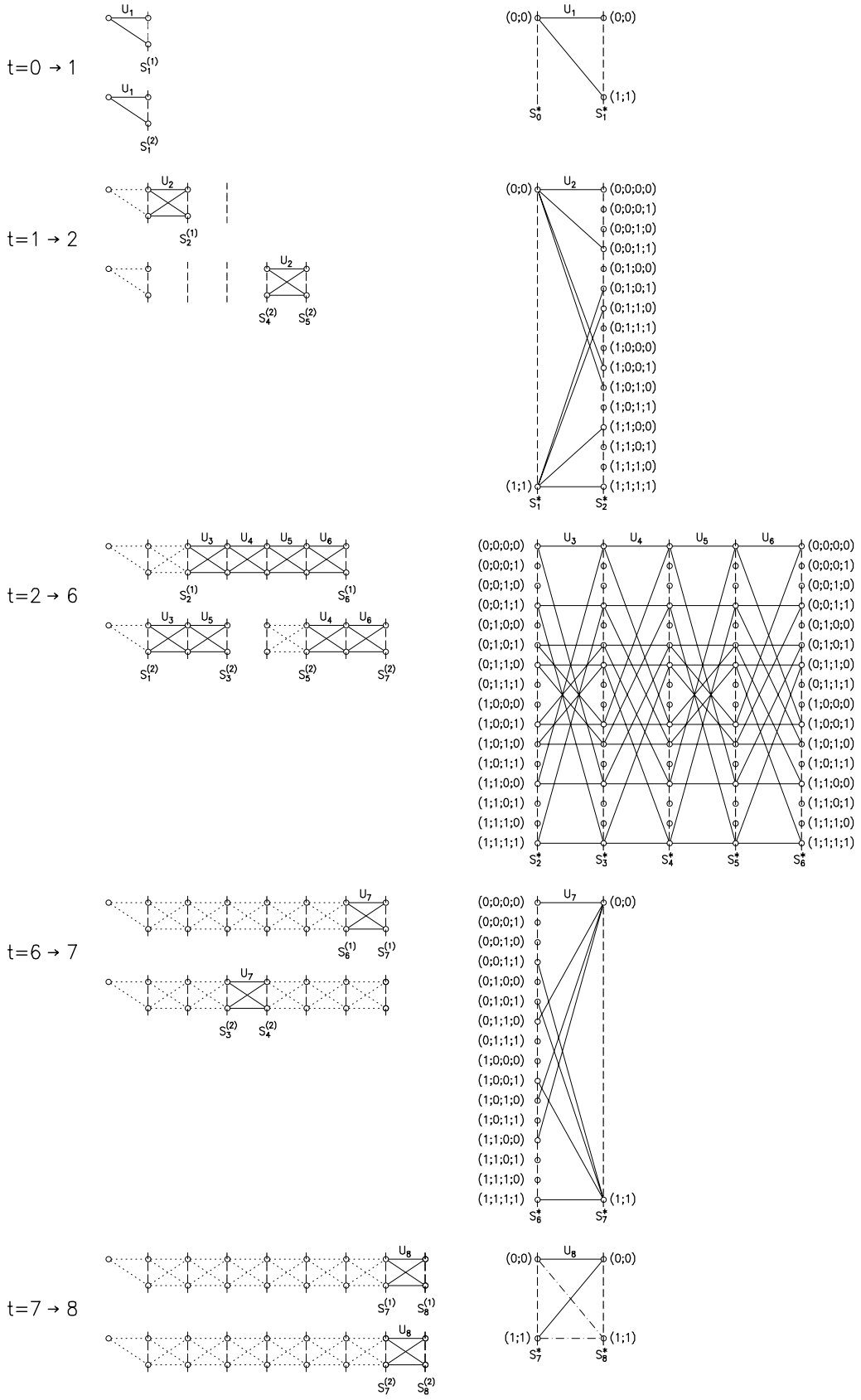

Fig. 6. Segments of the super-trellis for the turbo encoder FSM of Example 2

$(0;1)$ 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).

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

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}^*	$\bar{C}_{t+1}$
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	$(0;1;0;1)$	$(0;0;1)$

Continued on next page

Continued...				
t	S_t^*	U_{t+1}	S_{t+1}^*	$\bar{C}_{t+1}$
		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	$(0;0)$	$(1;0;0)$

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 $\mathbf{u}$ 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-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 Figure 6 ($t = 7 \rightarrow 8$) are marked with dash–dotted lines.

From Table II and Figure 6, we can clearly see that the turbo code super-trellis is time-variant, ie the structure of the trellis sections depends on the time instant t, exhibitig 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 sturcture 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 indentical 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 1st component trellis is only extended to the right (scenario (1) in Subsection III-C) upon increasing t. In the second component trellis for every $t = 2 \ldots \chi$ the *opening* (scenario (3) in Subsection III-C of a new section will occur, respectively. As a result of this, in the 2nd 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 left-most section in the 2nd 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 1st 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 Figure 6, from the set of 16 possible states only 8 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 8 of the 256 possible super-states (cf. Example 2 and simply swap the first and the second component).

In order to eliminate this ambiguity as regards to the trellis complexity (one of the complexity upper bounds is associated with 16 states while the other one 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 1st component trellis and χ sections in the 2nd 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 1st component trellis and one section in the 2nd component trellis, while the definition of the super–states is analogous to that in Subsection III-C, where only the 1st and 2nd 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 of the above elaborations, 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 1st component trellis, which is extended to the right upon increasing t.

The specific input symbol of the 2nd component scrambler – which belongs to the input symbol U_t and hence is input to the 2nd scrambler simultaneously with U_t entering the 1st one – is $U^{(2)}_{\pi_t}$. Therefore the

corresponding transition in the 2nd component trellis has the index π_t. Let us consider a state $S_l^{(2)}$ with arbitrary index $l = 1 \ldots (K-1)$ in the 2nd component trellis at time instant t. This state $S_l^{(2)}$ forms the interface at the right of a section in the 2nd 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 $S_l^{(2)}$. 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 2nd component trellis. The input symbol $U_l^{(2)}$, 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 $S_l^{(2)}$, 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 $S_l^{(2)},\ l = 1 \ldots K-1$ the probability that it represents the right interface to a section in the 2nd 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 $S_K^{(2)}$, since the end of the trellis is located directly to the right of it. If the 2nd component trellis is terminated, then we have $S_K^{(2)} = 0$, and hence $S_K^{(2)}$ does not appear as an interface state.

An arbitrary state therefore constitutes the right hand side interface of a section in the 2nd 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 $n_r(t) = K \cdot p_3(t) = t \cdot (K+1-t)/K$ right hand side (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 2nd component trellis. Similarly, one can deduct that there are $n_l(t) = n_r(t)$ left hand side (LHS) interfaces on average in the 2nd component trellis, when the edge effects at the start and end of the trellis are ignored.

Along with the single interface state $S_t^{(1)}$ in the 1st 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:

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

From Equation 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 $\mathbf{c}_i$ with the highest probability of having been transmitted, upon reception of the sequence $\mathbf{y}$, which is formulated as identifying the index i:

$$i = \underset{l}{\text{argmax}} \left(P(\mathbf{c}_l | \mathbf{y}) \right), \tag{3}$$

where argmax() returns the index of the maximum and $P(\mathbf{c}_l|\mathbf{y})$ is the conditional probability that $\mathbf{c}_l$ is the transmitted codeword when $\mathbf{y}$ 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 Eq. (3) for MAPSE decoding can be simplified to the following Maximum Likelihood Sequence Estimation (MLSE) type decoding using Bayes' rule, if all codewords $\mathbf{c}_l$ appear with the same probability:

$$i = \underset{l}{\text{argmin}} \left(\|\mathbf{c}_l - \mathbf{y}\|^2 \right).$$

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

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

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

$$i = \underset{l}{\text{argmin}} \left(\mathbf{M}_{\mathbf{c}_l} \right). \tag{5}$$

When the code used can be described by a trellis, i.e. when the code symbols are generated by a 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 Equation (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], [14], [15] and since the complexity of an MLSE decoder is considerably lower than that of a 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 $K = 998$. Figure 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 1st and 16th iteration. 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 $M = 2$,

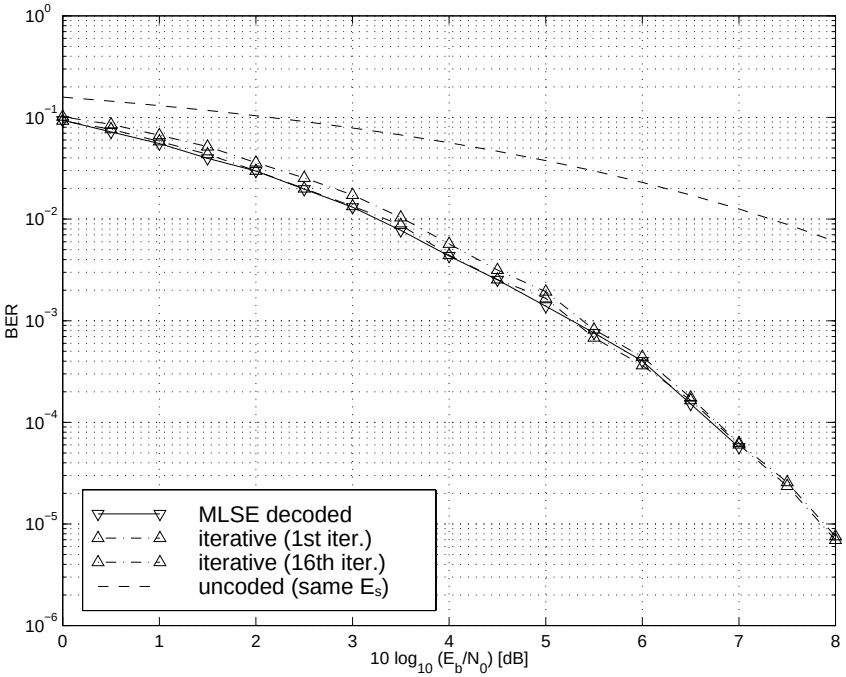

Fig. 7. Turbo coded BER performance for $M = 1$, 499×2 rectangular interleaver: comparison between optimum non-iterative and iterative decoding

which was chosen in order maintain a low complexity.

From Figure 8 we can see that the BER differences between the non-iterative 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 non-iterative MLSE decoding and iterative decoding. The 1st 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 non-iterative MLSE decoder. The remaining SNR gap is about 0.5dB 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. Figure 9 shows the performance of the iterative algorithm for the turbo code with the 333×3 interleaver. Specifically, the BER is shown after 1, 2, 4, 8 and 16 iterations, as is the BER curve for the non-iterative MLSE decoder. The performance improvements of the iterative decoder appear to saturate after a few iterations, exhibiting a performance gap with respect to the non-iterative MLSE decoder. In Figure 10 we see that a clear gap

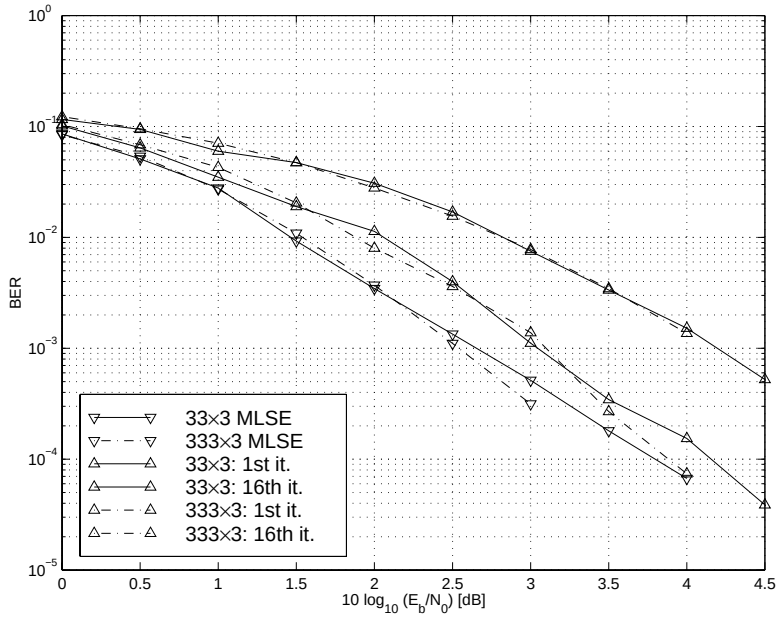

Fig. 8. Turbo coded BER performance for $M = 2$, 33×3- and 333×3 rectangular interleavers: comparison between optimum non-iterative and iterative decoding

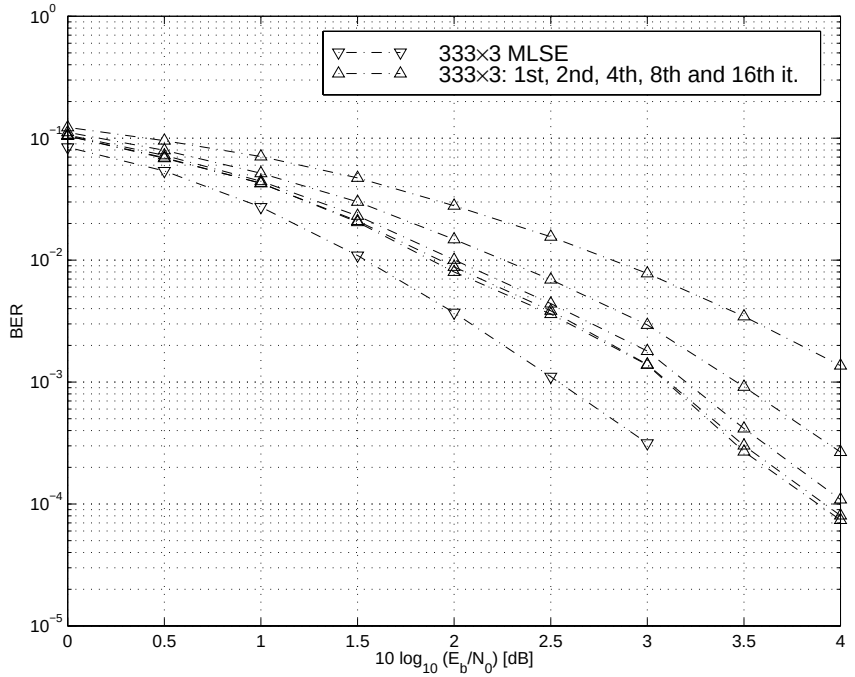

Fig. 9. Turbo coded BER performance for $M = 2$, 333×3 rectangular interleavers: convergence behaviour of the iterative algorithm compared to optimum non-iterative decoding

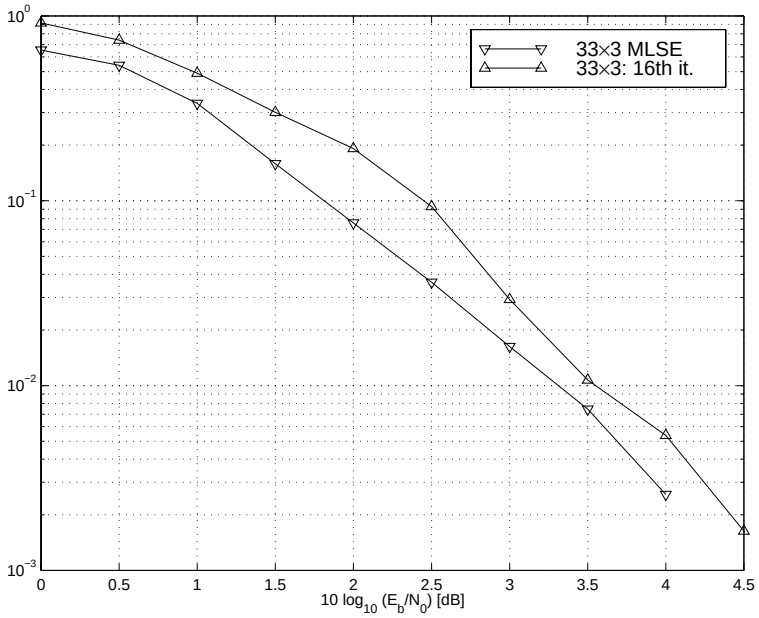

Fig. 10. Turbo coded Word Error Rate (WER) performance for $M = 2$, 33×3 rectangular interleaver: comparison between optimum non-iterative and iterative decoding

also exists between the Word Error Rate (WER) curves obtained for the non-iterative MLSE and the iterative decoder after 16 iterations using the 33×3 interleaver.

Figure 11 portrays the BER results for a turbo code incorporating a 39×5 rectangular interleaver, yielding $K = 195$. Observe that a gap of about 0.25dB remains between the performance of the iterative and the non-iterative MLSE decoder. Compared to Figure 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 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 non-iterative 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 far, are not employed in practical turbo codes, since their performance is relatively low in comparison to other interleavers. For short information word lengths,

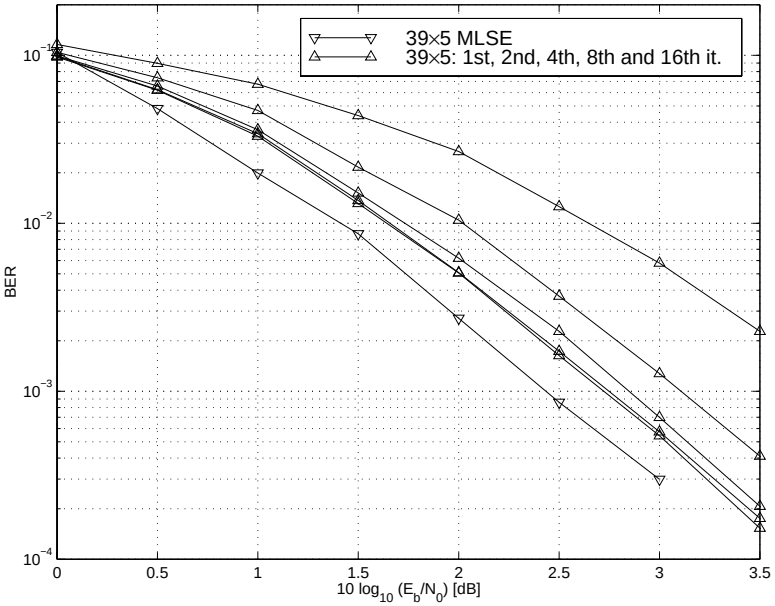

Fig. 11. Turbo coded BER performance for $M = 2$, 39×5 rectangular interleaver: comparison between optimum non-iterative and iterative decoding

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 non-iterative MLSE decoding in pratical 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, 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 8 super–states are actually encountered in the super–trellis. Therefore in Figure 12 we compared the performance of the turbo code ("TC") having a 499×2 rectangular interleaver and invoking non-iterative MLSE decoding of the super–trellis with that of a convolutional code ("CC") having a memory of $M = 3$ (8 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

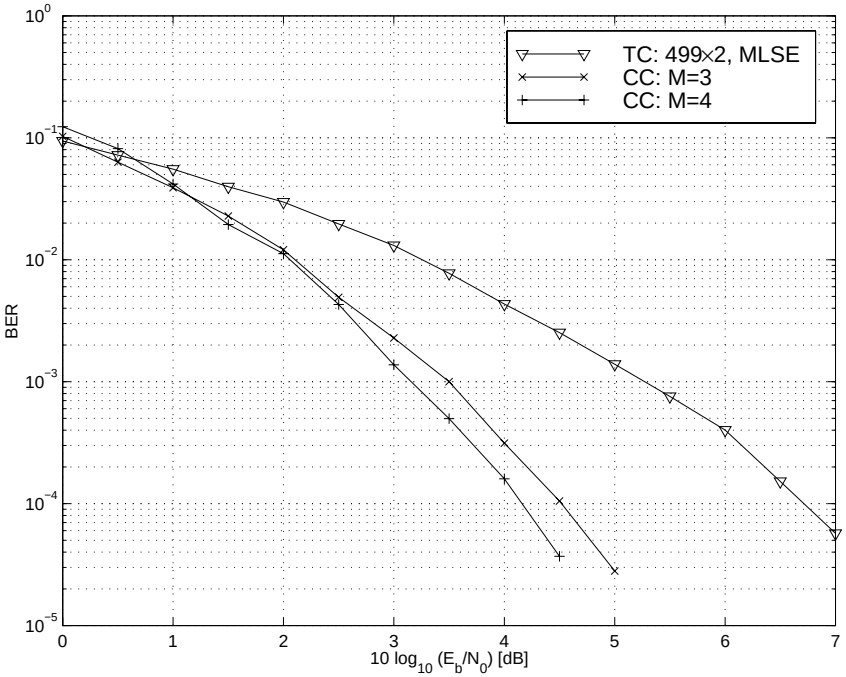

Fig. 12. Turbo coded BER performance with a maximum of 16 super-states, of which 8 are reached, compared to convolutional codes with 16 and 8 states

decoded.

Furthermore, in Figure 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 non-iterative MLSE–decoded turbo code of the same trellis complexity.

VI. Discussion of the results

As mentioned earlier, when inspecting the super–trellis of Figure 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, exhibitig 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 wass 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

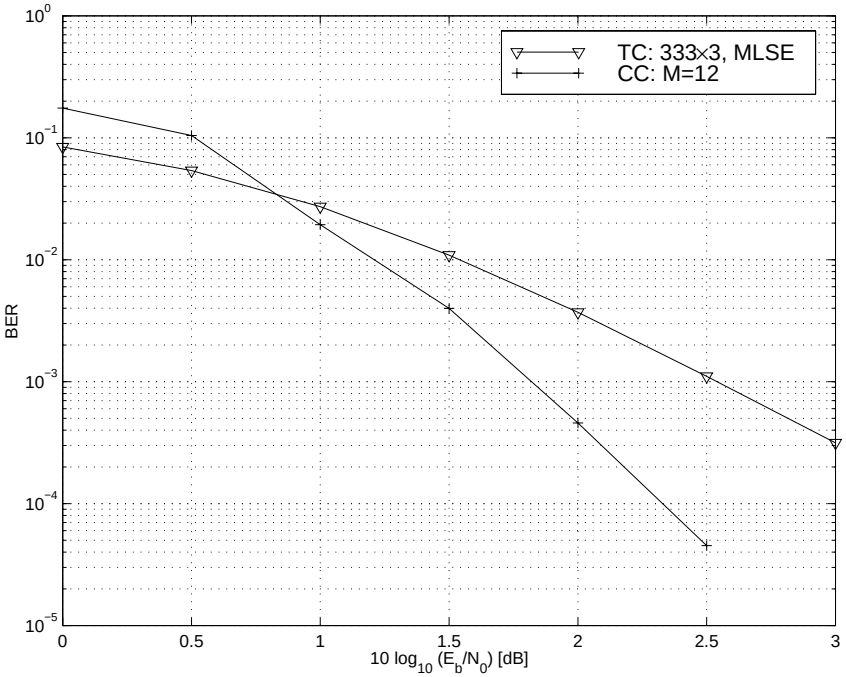

Fig. 13. Turbo coded BER performance with 4096 super-states compared to convolutional code with 4096 states

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 re-label the super–state vectors as follows:

$$\begin{aligned}
\tilde{S}_2^* &= (S_2^{(1)}; S_5^{(2)}; S_4^{(2)}; S_1^{(2)}) \\
\tilde{S}_3^* &= (S_3^{(1)}; S_2^{(2)}; S_4^{(2)}; S_5^{(2)}) \\
\tilde{S}_4^* &= (S_4^{(1)}; S_6^{(2)}; S_4^{(2)}; S_2^{(2)}) \\
\tilde{S}_5^* &= (S_5^{(1)}; S_3^{(2)}; S_4^{(2)}; S_6^{(2)}) \\
\tilde{S}_6^* &= (S_6^{(1)}; S_7^{(2)}; S_4^{(2)}; S_3^{(2)}),
\end{aligned}$$

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 re-labelled super-states. Note that re-labelling of the super-states does not affect the output vector $\bar{C}_t$ associated with a (re-labelled) super-state transition, which thus remains the same. Figure 14 shows a part of the super-trellis for Example 2, which has been rendered time-invariant by the above re-labelling 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 $\delta_{\text{free}} = 5$,

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

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 8 states (constraint length 4) and rate 1/3 has $\delta_{\text{free}} = 10$ [17], which explains the convolutional code's superior BER performance.

In general, the time–invariant super–trellis of a turbo code having a $\rho \times \chi$ rectangular interleaver, which has been obtained by relabelling 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.

Although turbo codes, which exhibit a complex super–trellis, cannot be non-iterative MLSE- decoded for complexity reason, they can be subjected to iterative decoding. Indeed, the more complex the super–

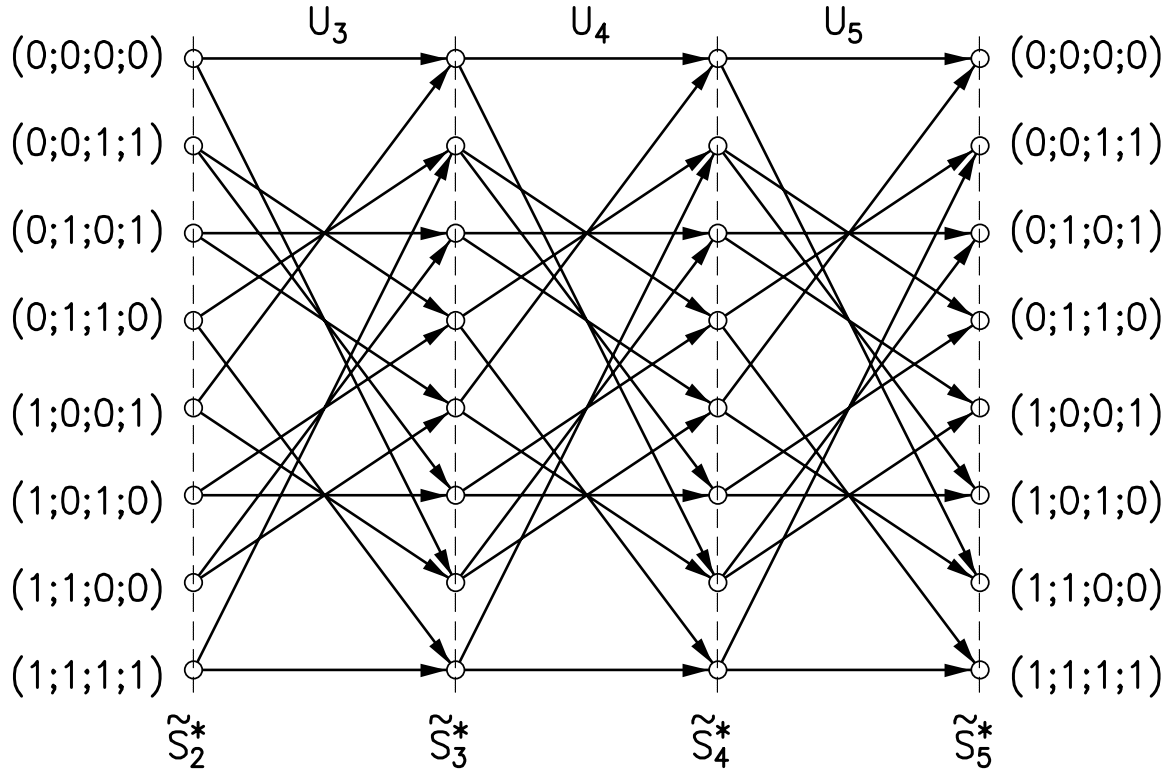

Fig. 14. Three segments of the new time-invariant super-trellis of Example 2 obtained by relabelling the super-states

trellis, the closer the performance of the iterative decoding seems to be to that of the non-iterative 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 non-iterative MLSE decoding, while according to [6] this is possible on average accross 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 maximising 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 bit error rate curve [6], [8]. In general the code performance is only determined at medium to high Signal–to–Noise rations (SNR) 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 endeavour to optimise $\delta_{\min}$, 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 non–recursive 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 non-zero 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 non-recursive 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

I. Proof of algorithmic optimality

In order to show that using a Viterbi algorithm along the super-trellis is optimum in the sense of non-iterative 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 $\mathbf{c}_{\text{opt}}$ in the sense of Equation (5) (for BPSK-transmission over an AWGN channel) is never discarded.

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

We claim that: $\mathbf{c}_{\text{opt}} \in \mathcal{M}_t \ \forall t$

Proof:

Commence Induction from: $\mathbf{c}_{\text{opt}} \in \mathcal{M}_0$

Inductive assumption: Let us assume that for an arbitrary t the following is true: $\mathbf{c}_{\text{opt}} \in \mathcal{M}_t$

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 $\mathbf{c}_{\text{opt}}$ is not discarded, i.e. $\mathbf{c}_{\text{opt}} \in \mathcal{M}_{t+1}$, as shown below:

From the mapping γ_{s} of the component scrambler FSM $\mathbb{F}_{\text{s}}$ 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:

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

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:

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

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 $(u_{k+1}^{(i)}; \ldots; u_l^{(i)})$:

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

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 sofar unexplored sections. Every turbo codeword $\mathbf{c}$ can be split into sub-vectors of two types, namely into J sub-vectors $\mathbf{c}_{\mathrm{pre},j},\ j=1\ldots J$ of code symbols, which are associated with the J component trellis sections belonging to the pre-history as regards to $t+1$, and in $\mathbf{c}_{\mathrm{post},n} = (c_{k_n+1}^{(i_n)};\ldots;c_{l_n}^{(i_n)}),\ n=1..N$, which are associated with the N sections belonging to the post-history.

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

$$\mathbf{c}_{\mathrm{pre}} = (\mathbf{c}_{\mathrm{pre},1};\ldots;\mathbf{c}_{\mathrm{pre},J}),$$

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

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

Any codeword $\mathbf{c}$ is hence constituted by these two vectors. Explicitly, $\mathbf{c}$ 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:

$$\mathbf{c} = \Pi\left(\mathbf{c}_{\mathrm{pre}};\mathbf{c}_{\mathrm{post}}\right). \tag{9}$$

According to Equation (8), for the post–history $\mathbf{c}_{\mathrm{post}}$ of any codeword $\mathbf{c}$, its N constituent parts $\mathbf{c}_{\mathrm{post},n}$ 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:

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

where

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

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

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

Again, $\mathbf{g}_2$ 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 $\mathbf{c}$ is merging into at time instant $t+1$, as discussed in Subsection III-C. The codeword symbols belonging to the post-history can therefore be expressed by

$$\mathbf{c}_{\mathrm{post}} = \mathbf{f}_4\left(s^*;\mathbf{u}_{\mathrm{post}}\right), \tag{12}$$

and this vector $\mathbf{c}_{\text{post}}$ exists, if and only if $(s^*; \mathbf{u}_{\text{post}})$ is a valid pair according to the condition of Equation (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 Subsection III-C.

Suppose that the optimum codeword $\mathbf{c}_{\text{opt}}$ is associated with the super–path $\overset{>}{p}_{\text{opt}}$ (the $>$ identifies it as a path) merging into the super-state s^*_{opt} at decoding stage $t+1$. Since we have imposed that $\mathbf{c}_{\text{opt}} \in \mathcal{M}_t$, 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 $\mathbf{c}_{\overset{>}{p}_{\text{opt}}}$ denote the code symbols belonging to the super-path $\overset{>}{p}_{\text{opt}}$ (i.e. the output of the turbo encoder FSM associated with this super-path of length $t+1$). Thus, $\mathbf{c}_{\overset{>}{p}_{\text{opt}}}$ is identical to $\mathbf{c}_{\text{opt,pre}}$ (pre–history part of $\mathbf{c}_{\text{opt}}$), and $\mathbf{c}_{\text{opt,post}}$ represents the remaining code symbols of $\mathbf{c}_{\text{opt}}$. We denote the information symbols associated with the post-history of $\mathbf{c}_{\text{opt}}$ by $\mathbf{u}_{\text{opt,post}}$. Then we see from Equation (12) that

$$\mathbf{c}_{\text{opt,post}} = \mathbf{f}_4\left(s^*_{\text{opt}}; \mathbf{u}_{\text{opt,post}}\right), \tag{13}$$

and from Equation (9) we have:

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

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

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

where $\mathbf{c}_{\text{opt,post}}$ is given in Equation (13).

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

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

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

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

$$\mathbf{M}_{\mathbf{c}_{\text{opt}}} = \mathbf{M}_{\overset{>}{p}_{\text{opt}}} + \mathbf{M}_{\text{opt,post}} \leq \mathbf{M}_{\mathbf{c}_2} = \mathbf{M}_{\overset{>}{p}_2} + \mathbf{M}_{\text{opt,post}} \tag{17}$$

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

$$\mathbf{M}_{\overset{>}{p}_{\text{opt}}} \leq \mathbf{M}_{\overset{>}{p}_2}. \tag{18}$$

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

References

[1] Claude Berrou, Alain Glavieux, and Punya Thitimajshima. Near shannon limit error-correcting coding and decoding: Turbo codes. *International Conference on Communications*, pages 1064–1070, 1993.

[2] Patrick Robertson. Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes. *Global Conf. on Comm.*, pages 1298–1303, 1994.

[3] R. Gallager. Low-density parity-check codes. *IEEE Transactions on Information Theory*, pages 21–28, 1962.

[4] L. Bahl et al. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Transactions on Information Theory*, pages 284–287, 1974.

[5] Joachim Hagenauer, Elke Offer, and Lutz Papke. Iterative decoding of binary block and convolutional codes. *IEEE Transactions on Information Theory*, pages 429–437, 1996.

[6] Sergio Benedetto and Guido Montorsi. Unveiling turbo codes: Some results on parallel concatenated coding schemes. *IEEE Transactions on Information Theory*, 42(2):409–428, 1996.

[7] Gérard Battail. On random-like codes. *Canadian Workshop on Inf. Th.*, 1995.

[8] Lance Perez, Jan Seghers, and Daniel Costello. A distance spectrum interpretation of turbo codes. *IEEE Transactions on Information Theory*, 42(6):1698–1709, 1996.

[9] A. Khandani. Design of turbo-code interleaver using Hungarian method. *Electronics Letters*, 34(1):63–65, 1998.

[10] Kenneth Andrews, Chris Heegard, and Dexter Kozen. Interleaver design methods for turbo codes. *Intern. Symposium on Inf. Th.*, page 420, 1998.

[11] A. Ambroze, G. Wade, and M. Tomlinson. Turbo code tree and code performance. *Electronics Letters*, pages 353–354, 1998.

[12] Claude Shannon. A mathematical theory of communication. *Bell System Technical Journal*, pages 379–427, 1948.

[13] J. Sadowsky. A maximum-likelihood decoding algorithm for turbo codes. *IEEE Communications Theory Workshop, Tucson*, 1997.

[14] Peter Höher. New iterative ("turbo") decoding algorithms. *International Symp. on Turbo Codes, Brest*, pages 63–70, 1997.

[15] Claude Berrou. Some clinical aspects of turbo codes. *International Symp. on Turbo Codes, Brest*, pages 26–31, 1997.

[16] P. Jung and M. Naßhan. Dependence of the error performance of turbo-codes on the interleaver structure in short frame transmission systems. *Electronics Letters*, pages 287–288, 1994.

[17] John Proakis. *Digital Communications*. McGraw Hill, 2nd edition, 1989.

[18] A. Jimenez and K. Zigangirov. Time-varying periodical convolutional codes with low-density parity-check matrix. *IEEE Transactions on Information Theory*, pages 2181–2191, 1999.