Compression of redundancy free trellis stages in Turbo-Decoder

Emmanuel Boutillon, José-Luis Sanchez-Rojas, Cédric Marchand

▶ **To cite this version:**

Emmanuel Boutillon, José-Luis Sanchez-Rojas, Cédric Marchand. Compression of redundancy free trellis stages in Turbo-Decoder. Electronics Letters, 2013, 49 (7), pp.460 - 462. hal-00825275

HAL Id: hal-00825275
https://hal.science/hal-00825275

Submitted on 23 May 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Compression of redundancy free trellis stages in Turbo-Decoder

E. Boutillon, J. Sánchez-Rojas and C. Marchand

For turbo code with coding rate close to one, the high puncturing rate induces long sequences of trellis without redundancy bit. A simplification technique to compute the final state of a sequence of Redundancy Free Trellis Stage (RFTS) is presented. It compresses a sequence of RFTS of length N into a sequence of RFTS of length $m-1+(N \mod (m-1))$, where m is the number of states of the trellis. The computation is reduced accordingly.

Introduction: Turbo codes with coding rate close to one are specified in the LTE (up to 0.95) and HSPA [1] (up to 0.98) standards, leading to highly punctured turbo codes. For such high rates, the trellis of each convolutional code can be viewed as long sequences of Redundancy Free Trellis Stage (RFTS), separated by single trellis stages with a redundancy bit, as shown in Fig. **??**. For example, lengths of RFTS sequence are $N=101$ bits or $N=102$ bits for the code rate 0.98 in HSPA. With such puncturing, the conventional sliding window implementation of the Forward-Backward algorithm [2] becomes inefficient. In fact, the convergence length L required to estimate accurately the initial state metric values of the window borders becomes large (L should be large enough to contain few redundancy bits). Large value of L impacts the hardware efficiency of the decoder. In [3] and related references, architectures with values of L up to 128 are reported for the LTE standard. In this letter, we present a method

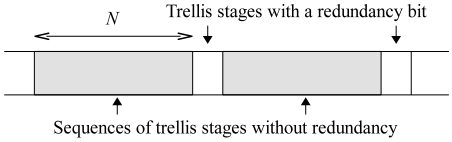

Fig. 1. Model of trellis with and without redundancy sections

that reduces the time of the convergence process of an m-state trellis code by reducing every sequence of RFTS of length N to a sequence of RFTS of length $m'+\phi_{m'}(N)$, where $m'=m-1$ and $\phi_{m'}(N)=N \bmod m'$. The proposed method can be viewed as a generalization of the method proposed in [4], replacing hard information bit by soft information bit.

Compression with hard information bits: Let us first focus our attention on trellis compression in the case of hard information bits. For a convolutional encoder of memory m, the state-space representation of [5] gives:

$$\mathbf{X}_{k+1} = [A]\cdot\mathbf{X}_k + [B]\cdot D_k \tag{1a}$$

$$\mathbf{V}_k = [C]\cdot\mathbf{X}_k + [D]\cdot D_k \tag{1b}$$

where $\mathbf{X}_k$, $\mathbf{V}_k$ and D_k are respectively the state of the encoder, the coded vector and the input information bit at time k, the summation are made on GF(2). Moreover, for a recursive code, the matrix $[A]$ verifies $[A]^{m'}=Id$, where Id is the identity matrix. Starting from a state $\mathbf{X}_0$ and the bit sequence $D_k, k=0..N-1$, the final state of the encoder is given by:

$$\mathbf{X}_N = [A]^N\cdot\mathbf{X}_0 + \sum_{k=0}^{N-1}[A]^k\cdot[B]\cdot D_{(N-1)-k} \tag{2}$$

Since $[A]^{m'}=Id$, then, for any k, $[A]^k=[A]^{\phi_{m'}(k)}$. Equation (2) can be simplified by regrouping (or agglomerating) the k values of same modulo m'. Let us assume first that $\phi_{m'}(N)=0$. In that case, equation (2) can be rewritten as:

$$\mathbf{X}_N = \mathbf{X}_0 + \sum_{k=0}^{m'-1}[A]^k\cdot[B]\cdot D^a_{(m'-1)-k} \tag{3}$$

where D^a_l is the l^{th} "agglomerated" bit defined as:

$$D^a_l = \sum_{j=0}^{N/m'-1} D_{m'\cdot j+l},\ l=0..m'-1 \tag{4}$$

When $\phi_{m'}(N)\neq 0$, $m'-\phi_{m'}(N)$ dummy bits equal to 0 are added to the N bits of the original sequence to obtain an extended sequence of length N_e. Since the extra dummy bits are all equal to zero, (2) gives $\mathbf{X}_{N_e}=[A]^{m'-\phi_{m'}(N)}\cdot\mathbf{X}_N$. Multiplying $\mathbf{X}_{N_e}$ by $[A]^{\phi_{m'}(N)}$ we obtain:

$$[A]^{\phi_{m'}(N)}\cdot\mathbf{X}_{N_e} = [A]^{m'}\cdot\mathbf{X}_N = \mathbf{X}_N \tag{5}$$

Since $\phi_{m'}(N_e)=0$, the agglomeration method (3) can be applied to compute $\mathbf{X}_{N_e}$ in m' trellis stages. Finally, $\phi_{m'}(N)$ extra trellis stages with input bits equal to 0 are required to process $\mathbf{X}_N$ (computation of equation (5)). To summarize, every trellis section of length N can be compressed in a trellis section of length $m'+\phi_{m'}(N)$ thanks to the "agglomeration" modulo m' of the input bits. One can consider the state of the trellis (one value among m possible) and the input bit (0 or 1) as Dirac distributions. Thus the question arises whether the same method can also be utilized for the non-Dirac distribution representing state metrics and branch metrics in the forward-backward decoding algorithm.

Compression with soft information bits: In this section, we replace the GF(2) summation of bits in equation (4) by the convolution of their associated probability distribution. For the sake of clarity, let us restrict to an $m=4$ states recursive convolutional encoder defined by the state equation (6) and let us consider the processing of the forward recursion on a sequence of RFTS of length $N=4$, starting from a state metric $\alpha_0(i), i=0..3$, where $\alpha_k(i)$ represents the probability of the encoder to be at state i at time k. Since there is no redundancy, the branch metrics are just given by the information bit (p_k,q_k), where $p_k=P(D_k=0)$ and $q_k=P(D_k=1)$ at time k, computed from both a-priori and channel values. Figure **??** shows the 4 sections of the trellis. The branches represented by filled lines (respectively dotted lines) are associated to an input bit 0 (respectively 1). On the graph, we show also in bold lines the 4 paths between state 0 at time $k=0$ and time $k=4$.

$$\mathbf{X}_{k+1} = \begin{bmatrix}1 & 1\\ 1 & 0\end{bmatrix}\cdot\mathbf{X}_k + \begin{bmatrix}1\\ 0\end{bmatrix}\cdot D_t \tag{6}$$

From this graph, it is easy to derive $\alpha_4(0)$ from its initial state metric α_0 and the a-priori bits $(p_k,q_k)_{k=0..3}$ as:

$$\begin{aligned}\alpha_4(0) = {}& \alpha_0(0)\cdot(p_0p_1p_2p_3 + p_0q_1q_2q_3 + q_0p_1p_2q_3 + q_0q_1q_2p_3)\\ &+\alpha_0(1)\cdot(p_0p_1p_2q_3 + p_0q_1q_2p_3 + q_0p_1p_2p_3 + q_0q_1q_2q_3)\\ &+\alpha_0(2)\cdot(p_0p_1q_2p_3 + p_0q_1p_2q_3 + q_0p_1q_2q_3 + q_0q_1p_2p_3)\\ &+\alpha_0(3)\cdot(p_0p_1q_2q_3 + p_0q_1p_2p_3 + q_0p_1q_2p_3 + q_0q_1p_2q_3)\end{aligned} \tag{7}$$

Factorizing (7), we obtain:

$$\begin{aligned}\alpha_4(0) = {}& \alpha_0(0)\cdot(p_1p_2\,(p_0p_3+q_0q_3) + q_1q_2\,(p_0q_3+q_0p_3))\\ &+\alpha_0(1)\cdot(p_1p_2\,(p_0p_3+q_0q_3) + q_1q_2\,(p_0q_3+q_0p_3))\\ &+\alpha_0(2)\cdot(p_1q_2\,(p_0p_3+q_0q_3) + q_1p_2\,(p_0q_3+q_0p_3))\\ &+\alpha_0(3)\cdot(p_1q_2\,(p_0p_3+q_0q_3) + q_1p_2\,(p_0q_3+q_0p_3))\end{aligned} \tag{8}$$

According to (4), $D^a_0=D_0+D_3$. Therefore, the probability density function (p^a_0,q^a_0) of D^a_0 is equal to $(p_0p_3+q_0q_3, p_0q_3+p_3q_0)$ (soft output of a parity constraint). Thus, we can reduce the $N=4$ trellis stages into $m'=3$ trellis stages, by replacing $(p_0p_3+q_0q_3)$ with p^a_0 and $(p_0q_3+p_3q_0)$ with q^a_0 in the first trellis section, as shown in figure **??**. This factorization is also explicitly given in (8). Since $\phi_3(4)=1$, an extra trellis section of length 1 is required to re-order the α_4 vector using only the 0-branches. This method can be generalized for any size of RFT sections and values of m. In particular, it could be applied for the 8-state encoder of the LTE and HSPA standards.

Implementation issues: In the above section, we have derived the "agglomeration" method in the probability domain. It can also be implemented in the logarithm domain using either the log-map or the max-log-map algorithms [2]. In the latter case, the agglomerated bit amplitude and sign are given by:

$$|LLR(D^a_l)| = \min\{|LLR(D_{m'\cdot j+l})|, j=0..N_e/m'-1\} \tag{9}$$

$$sign(LLR(D^a_l)) = \prod_{j=0}^{N_e/m'-1} sign(LLR(D_{m'\cdot j+l})) \tag{10}$$

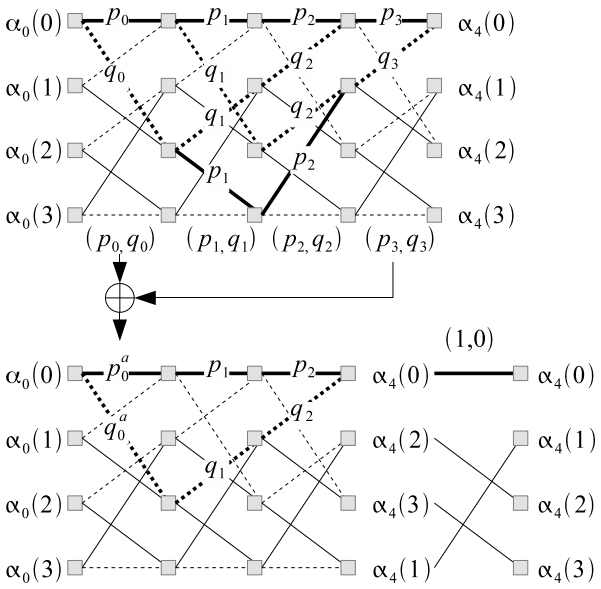

Fig. 2. Example of the agglomeration method for a 4-state trellis

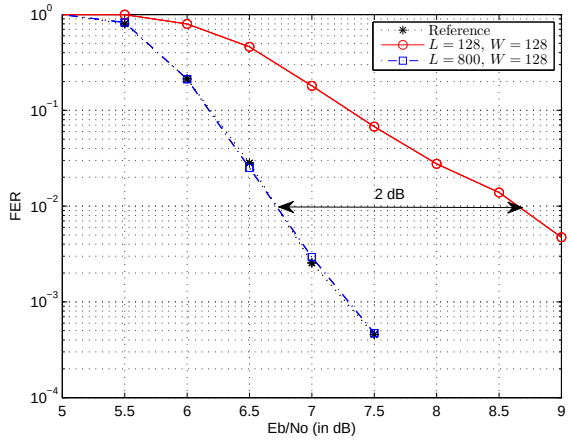

Fig. 3 *Simulation results for HSDPA, K=5114, R=0.98 with 4 decoding iterations*

where $LLR(D_k)$ is the Log-Likelihood Ratio of bit D_k defined as $LLR(D_k) = \ln(p_k/q_k)$. In the context of a turbo code, the computation of the agglomerated bits can be done on the fly during the generation of the extrinsic information of the previous iteration. Using these properties and without any change in the trellis structure, the convergence of the forward (or backward) algorithm can be significantly accelerated. For example, for a $(m = 8)$-state turbo code, a window of length $L = 64$ is processed in 64 cycles using conventional methods (one trellis stage per clock cycle). If the window does not contain any redundancy bit, trellis agglomeration allows to process it in $7 + \phi_7(64) = 8$ clock cycles. If the window contains a single section with a redundancy bit in position u (with $7 < u < 56$), then the number of clock cycles needed to process the window is $7 + \phi_7(u-1)$ to obtain the state vector α_{u-1}, plus one trellis section for obtaining α_u (trellis section with redundancy), and finally, $7 + \phi_7(64-u)$ clock cycles for obtaining α_{64}. The total number of clock cycles is thus equal to $15 + \phi_7(u-1) + \phi_7(64-u) = 22$. One should note that the last $\phi_{m'}(N)$ clock cycles are used only for shuffling on the state metric: with extra hardware (muxes), this operation can be done in one clock cycle. In that case, the 22 clock cycles reduce to $15 + 2 = 17$ clock cycles. Figure **??** shows the performance of a rate 0.98 HSPA turbo code of payload $K = 5114$, with 4 decoding iterations (for high coding rate, 4 decoding iterations is almost optimal). The curve reference is obtained performing exactly the forward-backward algorithm. The curves $L = 128$, $W = 128$ and $L = 800$, $W = 128$ are classical sliding windows implementation with a convergence length L and windows size W. With trellis compression, in $L = 128$ clock cycles, a convergence length of size $L = 800$ can be processed (length of RFTS sequence are $N = 101$ or 102 for this code rate). As shown in figure **??**, trellis compression gives optimal performance while the classical windows implementation degrades significantly the performances (2 dB for a Frame Error Rate (FER) of 10^{-2}).

Conclusion: In this paper, we show that for a m-state convolutional decoder, a sequence of RFTS of length N can be reduced to a sequence of RFTS of length $m' + \phi_{m'}(N)$ steps thanks to the bit agglomeration method ($m' + 1$ if extra muxes are used to perform the final shuffle). This method opens a new efficient way to perform sliding window based algorithms for high rate turbo codes, and it should have an impact on future architecture developments.

Acknowledgment: This work has been supported by the GIGADEC project from Région Bretagne, as well as the CPER projet PALMYRE II (Région Bretagne and FEDER funding).

E. Boutillon and C. Marchand (*Lab-STICC, UMR 6285, Université de Bretagne Sud, FRANCE*),
J. Sánchez-Rojas (*INICTEL-UNI, PERU*).

E-mail: emmanuel.boutillon@univ-ubs.fr

References

1 http://www.3gpp.org/ftp/Specs/html-info/36212.htm, version 10.0.0.
2 E. Boutillon, C. Douillard, G. Montorsi, 'Iterative Decoding of Concatenated Convolutional Codes: Implementation Issues', Transactions of the IEEE, vol. 95, no.6, June 2007.
3 M. May, T. Ilnseher, N. Wehn, W. Raab, 'A 150 Mbit/s 3GPP LTE Turbo Code Decoder', Design, Automation & Test in Europe Conference & exhibition (DATE), pp.1420-1425, Dresden, Germany, March 2010.
4 A.S. Barbulescu, S.S. Pietrobon, 'Terminating the trellis of turbo codes in the same state', Electronics Letters , vol.31, no.1, pp.22-23, Jan. 1995.
5 C. Weiss, C. Bettstetter, S. Riedel, D.J. Costello 'Turbo decoding with tail-biting trellises', International Symposium on Signal, System and Electronics, pp.343-348, Pisa, Italy, 1998.