A REDUCED MEMORY MAP ALGORITHM FOR TURBO CODES

Horia BALTA[1], Maria KOVACI[2], Lucian TRIFINA[3]
Electronics and Telecommunications Faculty,
2 Bd. V. Parvan, 300223 Tmisoara, Romania,
Electronics and Telecommunications Faculty,
Bd. Carol I, no. 11, Iasi, Romania,
[1]balta@etc.upt.ro,[2] kmaria@etc.upt.ro,[3] luciant@zeta.etc.tuiasi.ro

Abstract. *In this paper, a new method to reduce up to 90% of the memory necessary (for a four memory code) implied in MAP algorithms (also, LogMAP and MaxLogMAP algorithms) is presented. The proposed solution is to recalculate the backward coefficients in the same direction as the forward coefficients. Even if the volume of the calculation increases with the appearance of an additional recurrence, the decoding time can be reduced by the elimination of some times of the memory access.*
Keywords: *MAP algorithm, memory code, turbo code.*

Introduction

In an unpunctured turbo-coded system [1] (Fig.1), the emitted data are made up by blocks that each have three sequences, x_0, x_1, and x_2, with length N (the interleaving length). Correspondingly, a received block has three sequences of numbers y_0, y_1, and y_2, with the same length N. A decoder that implements a MAP like algorithm (MAP, LogMAP or MaxLogMAP, [2]) in order to decode catches two of the received sequences and, additionally, an extrinsic sequence, with the same length N. *The Maximum A-Posteriori (MAP) algorithm*, proposed by Bahl, Cocke, Jelinek and Raviv [3], is the highest performance algorithm used in turbo decoding. This algorithm calculates the Log Likelihood Ratio, LLR under the form [4]:

$$L(u_k/\underline{y}) = ln\left(\frac{\sum_{\substack{(\hat{s},s)\Rightarrow \\ u_k=+1}} P\left(S_{k-1}=\hat{s}\wedge S_k=s\wedge \underline{y}\right)}{\sum_{\substack{(\hat{s},s)\Rightarrow \\ u_k=-1}} P\left(S_{k-1}=\hat{s}\wedge S_k=s\wedge \underline{y}\right)}\right) = ln\left(\frac{\sum_{\substack{(\hat{s},s)\Rightarrow \\ u_k=+1}} \alpha_{k-1}(\hat{s})\cdot\gamma_k(\hat{s},s)\cdot\beta_k(s)}{\sum_{\substack{(\hat{s},s)\Rightarrow \\ u_k=-1}} \alpha_{k-1}(\hat{s})\cdot\gamma_k(\hat{s},s)\cdot\beta_k(s)}\right) \quad (1)$$

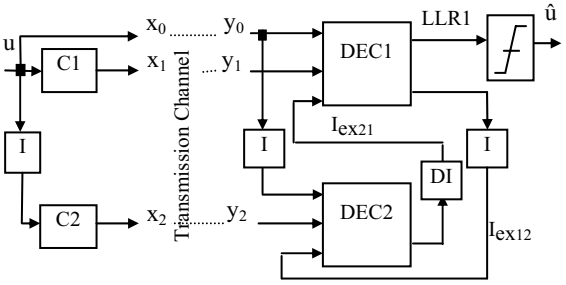

Figure 1. Turbo code – general scheme.

where:

$\alpha_{k-1}(\hat{s}) = P(S_{k-1} = \hat{s} \wedge \underline{y}_{j<k})$ is the probability that the encoder trellis was in $\hat{s}$ state at moment *k-1* and the received channel sequence, before this moment, is $\underline{y}_{j<k}$,

$$\alpha_k(s) = \sum_{all\ \hat{s}} \gamma_k(\hat{s},s)\cdot\alpha_{k-1}(\hat{s}) \quad (2)$$

$\beta_k(s) = P(\underline{y}_{j>k}/\ S_k = s)$ is the probability that, having been given the trellis state *s* at instant *k*, the received channel sequence, after this moment, to be $\underline{y}_{j>k}$,

$$\beta_{k-1}(\hat{s}) = \sum_{all\ \hat{s}} \gamma_k(\hat{s},s)\cdot\beta_k(s) \quad (3)$$

$\gamma_k(\hat{s},s) = P(\{\underline{y}_k \wedge S_k = s\}/\ S_{k-1} = \hat{s})$ is the probability that the encoder trellis take the transition from state $\hat{s}$ to state *s* and the received channel sequence for this transition is $\underline{y}_k$.

$$\gamma_k(\hat{s},s) = C \cdot e^{(u_k L(u_k)/2)} \cdot exp\left(\frac{E_b}{2\sigma^2} 2 \cdot a \sum_{i=1}^{n} y_{ki} \cdot x_{ki}\right) = \mathbb{C} \cdot e^{(u_k L(u_k)/2)} \cdot exp\left(\frac{L_c}{2} \sum_{i=1}^{n} y_{ki} \cdot x_{ki}\right) \quad (4)$$

where $Lc = 2 \cdot a \cdot E_b/\sigma^2$ is the channel reliability value and $L(u_k)$ is the interleaved extrinsic information, sometimes referred to as a priori.

The MAP algorithm (Fig.2) supposes the covering of the three distinct stages. The first one supposes the calculation of the gamma (coefficients) branches metrics. The second supposes the calculation of the two recurrences, the forward one from the first received symbol y_{01} to the last received symbol y_{0N} ($\alpha_{k-1}(\hat{s})$ coefficients), and the backward one, from y_{0N} toward y_{01} ($\beta_k(s)$ coefficients). The third stage is the calculating of the LLR for all N bits from the received block.

Because when $L(u_k/\underline{y})$ is calculated the $\alpha_{k-1}(\hat{s})$, $\beta_k(s)$ and $\gamma_k(\hat{s},s)$) coefficients are simultaneously present and because of the covering of the trellis in different directions the calculation of the (2) and (3) recurrences, the mentioned coefficients must be memorized. The needed memory for the decoding of a block received by a component decoder, taking into account an m memory component convolutional code, is:

$$M_0 = N \cdot (3 + 2^m + 2^m + 2^{m+1} + 2) \quad (5)$$

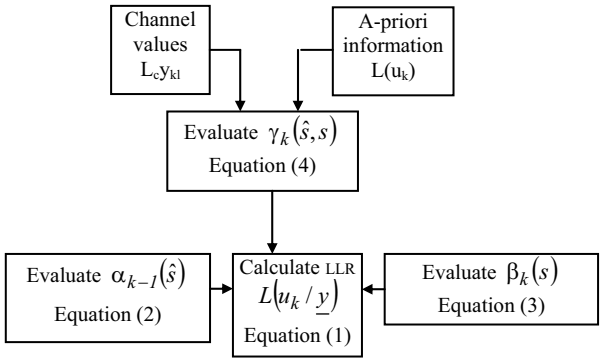

Figure 2. Summary of the key operations in the MAP algorithm.

memory addresses, including the memory space for the initial sequences (two received sequences and $L(u_k)$ a priori), for the alpha (2^m), beta (2^m) and gamma (2^{m+1}) coefficients, for the generated ($L(u_k/\underline{y})$ and $L(u_k)$ interleaving forward - extrinsic) sequences.

Obviously, M_0 increases exponentially with m, fact that will limit the value for the code memory.

Turbo codes achieve the highest coding gain known and should be the best candidates for error correction in high-speed wireless systems. However, the standard implementation of their decoding algorithm suffers from a large latency and high power consumption making them improper for mobile interactive systems [5]. To overcome this drawback, it is systematically analyzed the Maximum A Posteriori algorithm, the key-building block of the decoder, and stated that memory accesses are the bottleneck. Therefore, they have systematically optimized the data transfer and storage [5]. In the same paper a memory hierarchy is defined in order to benefit from the available temporal locality in the data accesses, frequently accessed data can be read from smaller and thus less power consuming memories.

It is used selective recomputation to reduce the amount of state metric storage needed in [6]. Similar with [6] we make a reduction of the memory using the selective memory.

This paper presents a modification of the algorithm implying a recalculation of the beta coefficients in the same recurrent direction as for the alpha coefficients. This allows us to give up the memorizing of the three coefficients to calculate the LLR.

Unidirectional MAP algorithm

Equation (3) can be considered as a matrix equation with the form:

$$\begin{pmatrix} \beta_{k-1}(0) \\ \beta_{k-1}(1) \\ \dots \\ \beta_{k-1}(\mu) \end{pmatrix} = \quad (6)$$

$$= \begin{pmatrix} \gamma_k(0,0) & \gamma_k(0,1) & \dots & \gamma_k(0,\mu) \\ \gamma_k(1,0) & \gamma_k(1,1) & \dots & \gamma_k(1,\mu) \\ \dots & \dots & \dots & \dots \\ \gamma_k(\mu,0) & \gamma_k(\mu,1) & \dots & \gamma_k(\mu,\mu) \end{pmatrix} \begin{pmatrix} \beta_k(0) \\ \beta_k(1) \\ \dots \\ \beta_k(\mu) \end{pmatrix}$$

where $\mu=2^m-1$ and $\gamma_k(\hat{s},s)$ is zero if there is no transition between the $\hat{s}$ and s states. If the matrix:

$$\Gamma_k = \begin{pmatrix} \gamma_k(0,0) & \gamma_k(0,1) & \dots & \gamma_k(0,\mu) \\ \gamma_k(1,0) & \gamma_k(1,1) & \dots & \gamma_k(1,\mu) \\ \dots & \dots & \dots & \dots \\ \gamma_k(\mu,0) & \gamma_k(\mu,1) & \dots & \gamma_k(\mu,\mu) \end{pmatrix} \quad (7)$$

is reversible, naming its inverse Γ_k^{-1}, we can built a „forward" recurrent relation for the beta coefficients, too:

$$(\beta_k(s)) = \Gamma_k^{-1} \cdot (\beta_{k-1}(s)) \quad (8)$$

Using relation (8) supposes the knowledge of the $\beta_1(s)$ coefficients, fact that can be realized by making first of all a backward recurrence. Therefore, to completely renounce the memorizing of the coefficients ($\alpha_{k-1}(\hat{s})$, $\beta_k(s)$ and $\gamma_k(\hat{s},s)$), we can do the following:

- the backward recurrence with relation (3) is made, only with the memorizing of the $\beta_1(s)$ coefficients,
- we compute for each k bit from *1* to N, in order, the $\gamma_k(\hat{s},s)$ coefficients with the relation (4), the $\alpha_k(\hat{s})$ coefficients with the relation (2) and $\beta_k(s)$ coefficients with the relation (6) and, finally, the LLR of the k bit and it extrinsic information. Only the last two values, which will form the output sequences, will be memorized.

This method totally drops the memorizing of the any coefficients for more than one recurrent step. The price paid is the computation of the beta and gamma coefficients twice.

Two hypotheses must be verified: Hyp1–that the Γ_k matrix is reversible and Hyp2–that the recalculation of the beta coefficients with backward recurrence will lead to the same value given by relation (3).

Regarding the first hypothesis, we have to note, first, the fact that for a binary convolutional code its trellis has two branches, which enter one node, respectively leave the node. As a result, there will be on each line and column from Γ_k two non-null gamma coefficients. Second, the gamma coefficients – the metrics of branches proceeded from the product of some exponentials, so they cannot be nulls (practical). Therefore, the only case when $det\Gamma_k$ can be null is when there are two proportional lines. Imposing a minimum absolute value to the determinant formed with the four gamma values can make the elimination of this situation.

Regarding the second hypothesis, the situation is more difficult. The executed simulations show that the equality between the original beta coefficients and the recalculated coefficients through the second recurrence is maintained only for the first 20 steps, after which the differences are significant, having even negative values. To avoid this problem, we can make a selective memorizing of the beta coefficients, from R to R steps, fact which will allow their being brought up-to-date in the backward recurrence. This fact implies a supplementary $2^m \cdot (N/R)$ bookings memorizing.

Experimental results

To illustrate the method, we suppose the component code with *G=[1, 5/7]* generating matrix. The trellis of this code is presented in Fig.3. The branches drawn with continuous lines correspond to the $u_k = -1$ input bit, while the branches with the broken line correspond to the $u_k = 1$ input bit. Using the notations from Fig.3, the Γ_k matrix is:

$$\Gamma_k = \begin{pmatrix} \gamma_0 & 0 & \gamma_1 & 0 \\ \gamma_2 & 0 & \gamma_3 & 0 \\ 0 & \gamma_4 & 0 & \gamma_5 \\ 0 & \gamma_6 & 0 & \gamma_7 \end{pmatrix} \quad (9)$$

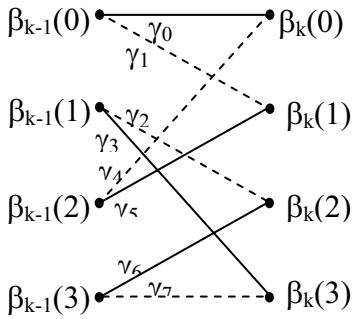

Figure 3. The trellis of the [1, 5/7] code.

If we suppose that:

$$\Delta u = \gamma_0 \cdot \gamma_3 - \gamma_1 \cdot \gamma_2 \neq 0 \quad \text{and} \quad \Delta w = \gamma_4 \cdot \gamma_7 - \gamma_5 \cdot \gamma_6 \neq 0 \tag{10}$$

it results that:

$$\Gamma_k^{-1} = \begin{pmatrix} \gamma_3/\Delta u & -\gamma_1/\Delta u & 0 & 0 \\ 0 & 0 & \gamma_7/\Delta w & -\gamma_5/\Delta w \\ -\gamma_2/\Delta u & \gamma_0/\Delta u & 0 & 0 \\ 0 & 0 & -\gamma_6/\Delta w & \gamma_4/\Delta w \end{pmatrix} \tag{11}$$

In the implementation of the unidirectional MAP algorithm to Δu and Δw the minimum values (absolutes) equals with 10^{-10} are imposed.

In the diagrams from Fig.4 the simulation results, made with a turbo code (TC), with decoders that implement the unidirectional MAP algorithm described above, are shown. Three values for the step numbers which restore the exact value of the beta coefficients *R=10*, *15* and *20* were used.

The other parameters of the TC are presented in the Table 1. The length of the block was chosen according to the [7]. For the interleaving we used a S-interleaver with *S=29*.

A Log Likelihood Ratio (LLR) stop criterion was selected. So, if all the LLR values of a block are outside of the interval [*-threshold LLR*, *threshold LLR*], then the iterative process is stopped, and the remaining iterations up to 15, are not executed anymore.

Table I. Parameters of the turbo codes.

	Parameter	Values
1	Component code	5/7
2	The TC configuration	parallel
3	Puncturing Coding rate	no $(N-m)/(3 \cdot N) \approx 1/3$
4	Interleaving The block lengths	S-ilv. (*S=29*) *N=1784* (CCSDS)
5	Channel	AWGN, *SNR=1dB*
6	Modulation	BPSK
7	Decoding algorithm	MAP unidirectional
8	The trellis closing	C1-closed, C2-unclosed $\beta(\,)$ of start = equiprobable
9	Quantization level	infinite
10	Number of iterations Stop criterion	*15* *threshold LLR = 10*

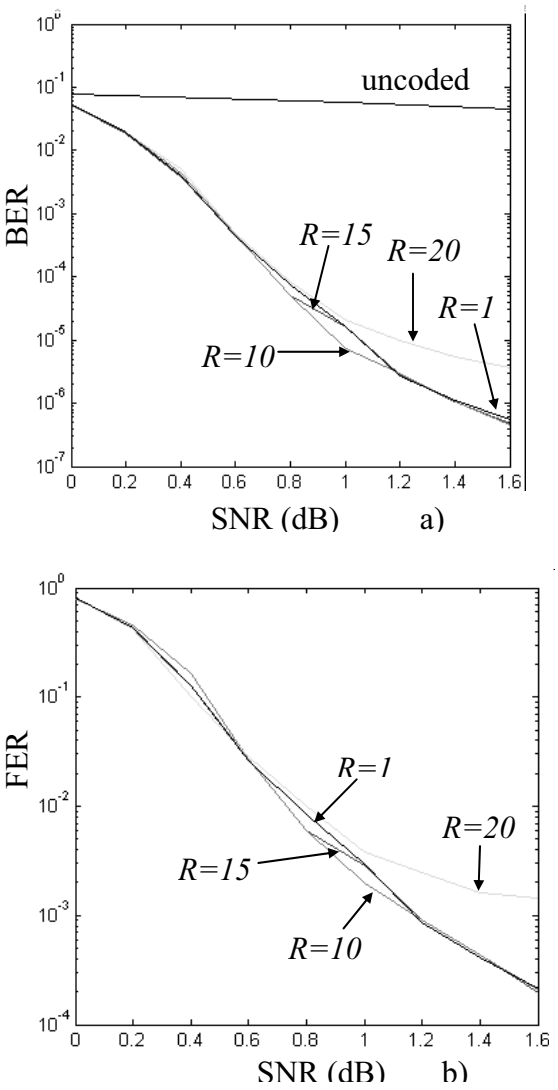

Figure 4. Bit Error Rate (BER) and Frame Error Rate (FER) for TC with R=1 (classic MAP), R=10, R=15 and R=20.

Conclusions

A simple modification of the famous MAP algorithm for linear block codes is proposed in [8].
In this paper, we present a modification of the MAP algorithm, which allows the drastic diminution of the needed memory. So, if the memory necessary for a component decoder in the classical MAP algorithm is of $N\cdot(5+2^{m+2})$ locations, using the new method, the necessary memory is $N\cdot(5+2^{m}/R)$ where $R\approx16$.
For component codes with an $m=2$ memory, the diminution of the needed memory is 75%, while for the 4 memory code, the diminution is *90%*.

The price of this implied memory diminution is given by increasing the calculating volume. Practically, the volume is almost double. The memory diminution, however, can lead to the diminution of the working times by the elimination of some times of the memory access. On the other hand, the diminution of the memory necessary for storing the alpha, beta and gamma coefficients can allow the involvement of the $m>4$ memory component code.

Acknowledgement

The authors are grateful to Professor Miranda Naforniță and to Professor Catherine Douillard, from ENST Bretagne, Brest, France, for the constructive discussions.

References

[1] C. Berrou, A. Glavieux, P. Thitimajshima (1993) *Near Shannon limit error-correcting coding and decoding: Turbo-codes*, Proc.ICC'93, Geneva, pp. 1064 – 1070.
[2] P. Robertson, E.Villebrun, P.Hoeher (1995) *A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain*, Proceedings of the International Conference on Communications, Seattle, USA, pag. 1009-1013.
[3] L.R. Bahl, J. Cocke, F. Jelinek, and J. Raviv (1974) *Optimal Decoding of Linear Codes for Minimising Symbol Error Rate*, IEEE Transactions on Information Theory, Vol. 20, pp. 284-287.
[4] L.Hanzo, T.H.Liew, B.L.Yeap (2002) *Turbo Coding, Turbo Equalisation and Space-Time Coding for Transmission over Fading Channels*, John Wiley & Sons Ltd.
[5] F. Maessen, A. Giulietti, B. Bougard, V. Derudder, L. Van der Perre, F. Catthoor, M. Engels *Memory power reduction for the high-speed implementation of turbo codes*, http://www.imec.be/wireless/turbocoding/pdf/turbo_codes_sipsBB.pdf.
[6] C. Schurgers, F. Catthoor, M. Engels (2001) *Memory Optimization of MAP Turbo Decoder Algorithms*, IEEE Transactions on VLSI Systems, Volume 9 , Issue 2, pp. 305-312.
[7] Consultative Committee for Space Data Systems (2002) *Telemetry Channel Coding*, CCSDS 101.0-B-6, Blue Book.
[8] A. Trofimov, T. Johansson (2004) *A memory-efficient optimal APP symbol decoding algorithm for linear block codes*, IEEE Transactions on Communications, vol. 52, N.9, pp.1429-1434.