Proc, Int. Symp. Turbo Codes (Brest, France), pp. 259-262, Sept. 2003 (with corrections)

Enhanced Max-Log-APP and Enhanced Log-APP Decoding for DVB-RCS

Youssouf Ould-Cheikh-Mouhamedou† *Paul Guinand*‡ *Peter Kabal*†

† Electrical and Computer Engineering, McGill University, Montreal, Quebec H3A 2A7
‡ Communications Research Centre, 3701 Carling Ave., Ottawa, Ontario K2H 8S2

Abstract

We present a new decoding technique for double-binary turbo codes, such as in the Digital Video Broadcasting for Return Channel via Satellite (DVB-RCS) standard. The proposed techniques are referred to as the Enhanced Max-Log *A Posteriori* Probability (APP) Decoding and Enhanced Log-APP Decoding. Results for ATM packets show a degradation of 0.05 dB in BER/FER for enhanced max-log-APP compared to conventional log-APP. For MPEG packets the enhanced max-log-APP outperforms the log-APP at high SNRs. For both packet lengths, enhanced log-APP outperforms log-APP at high SNRs. Simulation results for an effective early stopping criterion are also presented.

1 Introduction

DVB-RCS [1] has been standardized by the ETSI for digital video broadcasting. It is for use on a geostationary earth orbit (GEO) satellite interactive network, which provides multimedia and Internet traffic service.

The 8-state double-binary turbo code used in DVB-RCS is specified for various coding rates (1/3 to 6/7) and block sizes (12 to 216 Bytes) including ATM and MPEG sizes. Compared to classical turbo codes, which use Recursive Systematic Convolutional (RSC) single-binary codes, the double-binary turbo codes have many advantages [2]: (a) Improved performance by reducing the correlation effects between the component decoders. (b) Increased minimum free distance by introducing periodic disorder in the symbols. (The classical turbo codes suffer from severe flattening at low error rates, but the double-binary turbo codes do not). (c) Puncturing can be used to increase the code rate and data rate. Double binary codes are less sensitive to puncturing than the single-binary codes [3].

Practical approaches for reducing the computation complexity (CC) of *a posteriori* probability decoding (APP) for single-binary turbo codes have been introduced in [4], [5]. However, their application to double-binary codes is not straightforward. In this paper, we present a new decoder for the DVB-RCS turbo codes that improves the decoding performance with a low CC. In addition, by applying an effective early stopping (ES) criterion, the computational complexity of the decoder is reduced significantly.

This work was supported by CRC in Ottawa (Canada).

2 DVB-RCS Encoding Scheme

2.1 Encoder Structure

Fig. 1 shows the encoder as described in standardized DVB-RCS. To encode the data sequence, the Circular Recursive Systematic Convolutional (CRSC) encoder must be fed four times, two times in normal operation mode (switch in position 1), and two times in interleaved order (switch in position 2), as shown in Fig. 1. The data sequence is fed to the encoder in the form of packets of length N couples, where $N = 2k$ and N is multiple of 8.

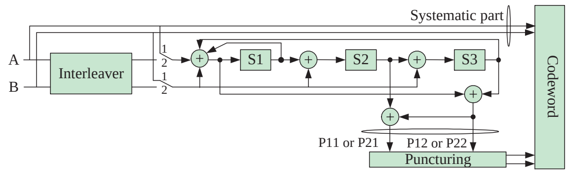

Fig. 1 Double-binary CRSC encoder for DVB-RCS turbo code

2.2 Circular Coding

Forcing the encoder to a known state at the end of the encoding stage by adding tail bits, which are then sent to the decoder, presents two major drawbacks. Firstly, the minimum free distance $\hat{d}_{\text{free}}$ is no longer equal to the original minimum free distance d_{free} for all information data. Secondly, the spectral efficiency of the transmission is degraded [6], especially for shorter blocks.

Circular coding makes it possible to start the encoding at a known state, which is called the *circular state* $\boldsymbol{S}_c$, and end the encoding in the same state without the two drawbacks mentioned above. If the data to be encoded contains N couples, where N is not a multiple of the period of the encoding recursive generator, then the existence of such circular state $\boldsymbol{S}_c$ is ensured. Since the value of the circular state $\boldsymbol{S}_c$ depends on the contents of the sequence to be encoded, determining the circular state requires a pre-encoding operation. For the pre-encoding the encoder is initialized to the *all zero* state. Then the data sequence goes through the encoder. At the end of the encoding stage the Encoder State is $\boldsymbol{S}_{N_1}^0$ for unpermuted and $\boldsymbol{S}_{N_2}^0$ for permuted data. The circular states $\boldsymbol{S}_{c_1}$ and $\boldsymbol{S}_{c_2}$ values are then obtained from the expression $\boldsymbol{S}_{c_{1,2}} = \left(\boldsymbol{I} + \boldsymbol{G}^N\right)^{-1} \boldsymbol{S}_{N_{1,2}}^0$, where $\boldsymbol{I}$ is the unit matrix and $\boldsymbol{G}$ is the generator matrix of the code.

3 Decoder for DVB-RCS Turbo Codes

Fig. 2 shows a block diagram [7] of the iterative decoder of the DVB-RCS double-binary turbo code. Let $\boldsymbol{d}$ be the sent data sequence. For the first iteration of the iterative decoding, the *a priori* log-likelihood-ratio $\boldsymbol{L}^{(z)}(\boldsymbol{d})$, $z \in \{01, 10, 11\}$, is initialized to zero and the first decoder (DEC1) computes the *a posteriori* log-likelihood-ratio $\boldsymbol{L}^{1(z)}(\hat{\boldsymbol{d}})$ based on $\boldsymbol{L}^{(z)}(\boldsymbol{d})$, the *a posteriori* log-likelihood-ratio $\boldsymbol{L}_c^{1(z)}(\boldsymbol{d})$ for unpermuted received systematic symbols, received systematic symbols $\boldsymbol{y}_S$, and Parity1 $\boldsymbol{y}_{P1}$. DEC1 then computes the extrinsic information $\boldsymbol{L}_e^{1(z)}(\hat{\boldsymbol{d}}) = \boldsymbol{L}^{1(z)}(\hat{\boldsymbol{d}}) - \boldsymbol{L}^{(z)}(\boldsymbol{d}) - \boldsymbol{L}_c^{1(z)}(\boldsymbol{d})$. The extrinsic information from DEC1 is interleaved and passed to the second decoder (DEC2), which uses $\boldsymbol{L}_{\mathrm{eI}}^{1(z)}(\hat{\boldsymbol{d}})$ as the *a priori* values in place of $\boldsymbol{L}^{(z)}(\boldsymbol{d})$.

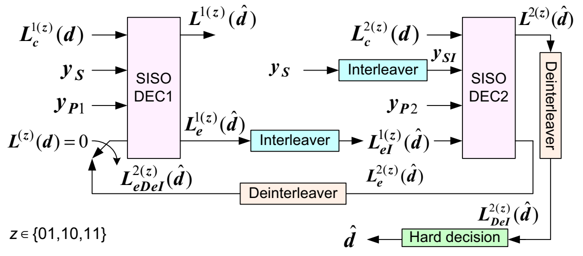

Fig. 2 Iterative decoder for DVB-RCS turbo code

DEC2 computes the *a posteriori* log-likelihood-ratio $\boldsymbol{L}^{2(z)}(\hat{\boldsymbol{d}})$ based on interleaved extrinsic information from the first decoder $\boldsymbol{L}_{\mathrm{eI}}^{1(z)}(\hat{\boldsymbol{d}})$, the *a posteriori* log-likelihood-ratio $\boldsymbol{L}_c^{2(z)}(\boldsymbol{d})$ for permuted received systematic symbols, permuted received systematic symbols $\boldsymbol{y}_{SI}$, and Parity2 $\boldsymbol{y}_{P2}$. DEC2 computes the extrinsic information $\boldsymbol{L}_e^{2(z)}(\hat{\boldsymbol{d}}) = \boldsymbol{L}^{2(z)}(\hat{\boldsymbol{d}}) - \boldsymbol{L}_{\mathrm{eI}}^{1(z)}(\hat{\boldsymbol{d}}) - \boldsymbol{L}_c^{2(z)}(\boldsymbol{d})$. DEC1 will use the de-interleaved extrinsic information values $\boldsymbol{L}_{\mathrm{eDeI}}^{2(z)}(\hat{\boldsymbol{d}})$ as *a priori* information in the preceding iterations. This computation is repeated for each iteration. After a number of iterations, a hard decision is made based on the de-interleaved *a posteriori* log-likelihood-ratio $\boldsymbol{L}_{\mathrm{DeI}}^{2(z)}(\hat{\boldsymbol{d}})$. DEC1 and DEC2 have the same structure since the two component codes of the DVB-RCS turbo code are the same.

3.1 APP and Max-Log-APP Decoding

Turbo codes use constituent codes, which admit soft decoding to provide reliable estimates that can be passed between the constituent decoders. The Bahl, Cocke, Jelinek, and Raviv (BCJR) algorithm [8] or APP decoding (also called MAP decoding) algorithm gives an optimal estimate of the information symbols given the received data sequence, but APP decoding is rarely used in practice, because of its high computational complexity.

For the APP decoding algorithm [8], we define $\phi^0 = \{00, 01, 10, 11\}$ as the set of all possible symbols, $\phi = \{01, 10, 11\}$ as the set of all possible symbols except $\{00\}$, d_t the sent symbols at time t, x_t the input to the channel resulting from d_t, y_t the output of the channel resulting from x_t and $\boldsymbol{y}$ the observed samples at the output of the channel. Assume that the Markov process goes from the state s' at time $(t-1)$ to the state s at time t due the input $d_t = z$, where $z \in \phi^0$. APP decoding can be obtained from the following expressions.

$$L^{(z)}(\hat{d}_t) \equiv \ln \frac{p\left(d_t = z \mid \boldsymbol{y}\right)}{p\left(d_t = 00 \mid \boldsymbol{y}\right)} = \ln \frac{\sum\limits_{(s',s);z\in\phi} \alpha_{t-1}(s') \cdot \gamma_t^z(s',s) \cdot \beta_t(s)}{\sum\limits_{(s',s)} \alpha_{t-1}(s') \cdot \gamma_t^{00}(s',s) \cdot \beta_t(s)} \tag{1}$$

$$\begin{aligned}\gamma_t^z(s',s) &= p(s \mid s') \cdot p(y_t \mid s', s) \\ &= p(y_t \mid d_t = z) \cdot p(d_t = z)\end{aligned} \tag{2}$$

Forward recursion $$\alpha_t(s) = \sum_{s';\forall z\in\phi^0} \alpha_{t-1}(s') \cdot \gamma_t^z(s',s) \tag{3}$$

Backward recursion $$\beta_{t-1}(s') = \sum_{s;\forall z\in\phi^0} \beta_t(s) \cdot \gamma_t^z(s',s) \tag{4}$$

The max-log-APP (ML-APP) decoding algorithm is an approximation of the log-APP (L-APP) decoding and is widely used in practice because of its reduced CC. However, it leads to degradation in the error rate performance. Later we will introduce a technique, which overcomes this degradation. The $\overline{\alpha}$, $\overline{\beta}$, $\overline{\gamma}$ and log-likelihood ratios can be obtained from the following expressions.

$$\begin{aligned}\overline{\gamma}_t^z(s',s) &\equiv \ln \gamma_t^z(s',s) \\ &= \ln p(y_t \mid d_t = z) + \ln p(d_t = z)\end{aligned} \tag{5}$$

$$\begin{aligned}\overline{\alpha}_t(s) &= \ln \alpha_t(s) \\ &= \ln \sum_{s';\forall z\in\phi^0} e^{\overline{\alpha}_{t-1}(s') + \overline{\gamma}_t^z(s',s)} \\ \overline{\alpha}_t(s) &\approx \max_{s';\forall z\in\phi^0} \left(\overline{\alpha}_{t-1}(s') + \overline{\gamma}_t^z(s',s)\right)\end{aligned} \tag{6}$$

$$\begin{aligned}\overline{\beta}_{t-1}(s') &= \ln \beta_{t-1}(s') \\ &= \ln \sum_{s;\forall z\in\phi^0} e^{\overline{\beta}_t(s) + \overline{\gamma}_t^z(s',s)} \\ \overline{\beta}_{t-1}(s') &\approx \max_{s;\forall z\in\phi^0} \left(\overline{\beta}_t(s) + \overline{\gamma}_t^z(s',s)\right)\end{aligned} \tag{7}$$

$$\begin{aligned}L^{(z)}(\hat{d}_t) \approx &\max_{(s',s);z\in\phi} \left(\overline{\alpha}_{t-1}(s') + \overline{\gamma}_t^z(s',s) + \overline{\beta}_t(s)\right) \\ &- \max\left(\overline{\alpha}_{t-1}(s') + \overline{\gamma}_t^{00}(s',s) + \overline{\beta}_t(s)\right)\end{aligned} \tag{8}$$

Double-binary codes have a large number of branches entering and leaving each node in the trellis diagram. This leads to high CC and so necessitates careful attention to efficient implementation.

3.2 Enhanced Max-Log-APP and Enhanced Log-APP Decoding

The enhanced max-log-APP (EML-APP) was first introduced in [9], where the extrinsic output of soft-output-Viterbi-decoder (SOVA) [10] is scaled by an SNR-dependent scale factor (SF). This technique has been applied to ML-APP for UMTS [11] (single-binary code). EML-APP is applied here to DVB-RCS (double-binary code) and uses an SNR-*independent* scale factor.

Note that, iterative decoding for turbo codes with L-APP is not optimal in the sense of making a maximum likelihood decision. However scaling the extrinsic output of L-APP with an appropriate SF which is SNR-independent, can reduce the sub-optimality of iterative decoding. We call this approach enhanced log-APP (EL-APP).

4 Simulation Results

To compare the four algorithms ML-APP, EML-APP, L-APP and EL-APP, all algorithms were simulated using an ATM packet size of 424 bits and an MPEG packet size of 1504 bits. The code rate is 1/3 and the maximum number of iterations is fixed at 8 iterations. For computing the initial α_i and β_i, the overlap for circular coding is set to 30 symbols (60 bits). In order to make a fair comparison, the same noise sequence has been used for the four algorithms at each SNR value.

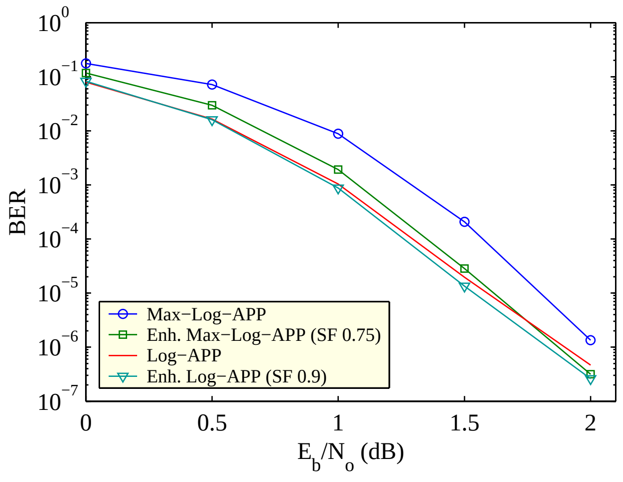

Fig. 3 BER for ATM-Packet

Simulations were run with SFs from 0.05 to 0.95 (steps of 0.05). It was found that the SFs of 0.75 and 0.9 are good choices for EML-APP and EL-APP, respectively, independent of SNR.

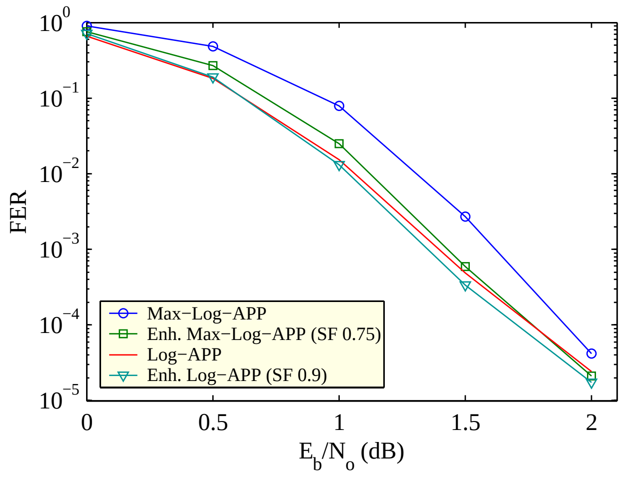

Fig. 4 FER for ATM-Packet

EML-APP versus ML-APP and L-APP

Simulation results are shown in Figs. 3–6. Compared to ML-APP, EML-APP (SF=0.75) has an improvement of 0.2 dB for both BER and FER for both packet sizes. For ATM packets, EML-APP (SF=0.75) reduces the gap from L-APP to 0.05 dB for both BER and FER. For longer packets, such as for MPEG, EML-APP (SF=0.75) outperforms L-APP at high SNRs (see Fig. 5 and 6).

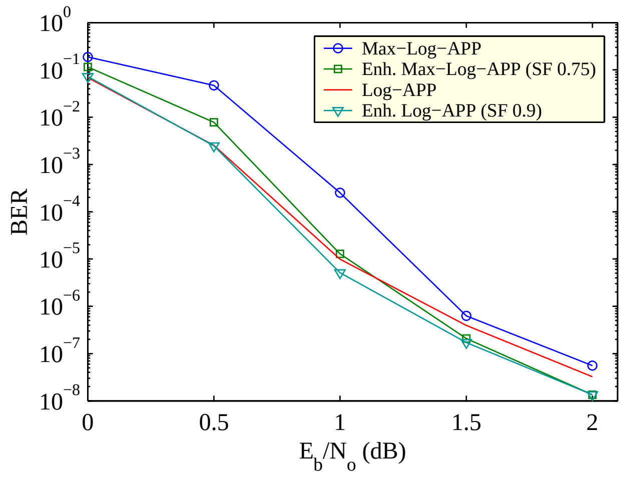

Fig. 5 BER for MPEG packet

EL-APP versus L-APP

For ATM packets, EL-APP (SF=0.9) has an improvement of 0.05 dB versus L-APP for both BER and FER at SNRs $\geq$ 1 dB. For MPEG packets the EL-APP (SF=0.9) has an improvement of 0.1 dB to 0.2 dB compared to L-APP for both BER and FER at SNRs $\geq$ 1 dB.

The BER/FER of EML-APP (SF=0.75) and EL-APP (SF=0.9) can be better than the BER/FER of L-APP due to the fact that iterative decoding is sub-optimal, as a result of the correlation effects between the component decoders.

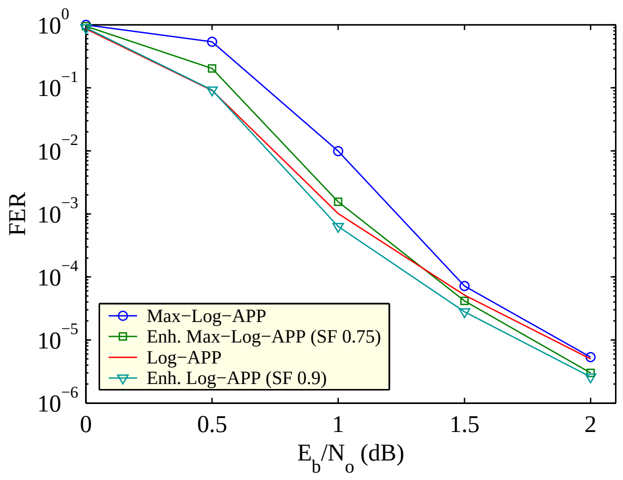

Fig. 6 FER for MPEG packet

4.1 Early Stopping

We used EML-APP (SF=0.75) to test the effect of early stopping. The ES-Criterion employed [12] requires that the hard decisions from the constituent decoders must agree for W consecutive decodings.

Fig. 7 shows almost no degradation in BER/FER by choosing $W = 2$. For SNRs > 1.5 dB, more than half of the packets were decoded with early stopping after 2 full iterations. This reduces the average number of full iterations from 8 to less than 5 leading to a reduction in the average computational complexity by a factor of about 2.

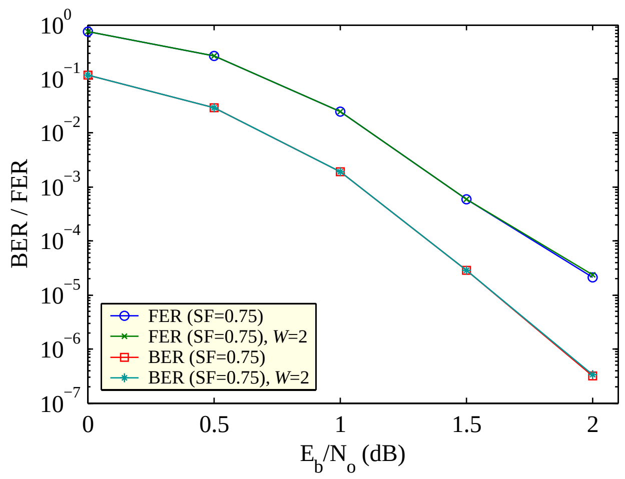

Fig. 7 BER and FER with ES for ATM packet

5 Conclusion and summary

Decoding techniques EML-APP (SF=0.75) and EL-APP (SF=0.9) have been introduced. The BER/FER performance of EML-APP is very close to that of L-APP. At high SNRs, EML-APP can outperform the L-APP. EL-APP (SF=0.9) outperforms L-APP at high SNRs.

An effective early stopping criterion, which reduces significantly the computation complexity with almost no degradation in BER/FER, has been used.

6 Acknowledgements

The authors would especially like to thank Dr. John Lodge at CRC (Ottawa) for his great support. Special thanks to Dr. Stewart Crozier, Dr. Ron Kerr and Andrew Hunt at CRC for their comments and feed back.

References

[1] “Interaction channel for satellite distribution systems.” ETSI EN 301 790, V1.3.1, Mar. 2003.

[2] C. Berrou, M. Jézéquel, C. Douillard, and S. Kerouédan, “The advantages of non-binary turbo codes,” *Proc. IEEE Inform. Theory Workshop*, pp. 61–63, Sept. 2001. (Cairns, Australia).

[3] P. J. Lee, “Constructions of rate $(n-1)/n$ punctured convolutional codes with minimum required SNR criterion,” *IEEE Trans. Commun.*, vol. 36, pp. 1171–1174, Oct. 1988.

[4] P. Robertson and P. Hoeher, “Optimal and suboptimal maximum a posteriori algorithms suitable for turbo decoding,” *European Trans. Telecommun.*, pp. 119–125, Mar.–Apr. 1997.

[5] W. J. Gross and P. G. Gulak, “Simplified MAP algorithm suitable for implementation of turbo decoders,” *Electronics Letters*, vol. 34, pp. 1577–1578, Aug. 1998.

[6] C. Douillard, M. Jézéquel, and C. Berrou, “The turbo code standard for DVB-RCS,” *Proc. 2nd Int. Symp. Turbo codes*, pp. 551–554, Sept. 2000. (Brest, France).

[7] J. Hagenauer, E. Offer, and L. Papke, “Iterative decoding of binary block and convolutional codes,” *IEEE Trans. Inform. Theory*, vol. 42, pp. 429–445, Mar. 1996.

[8] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate,” *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284–287, Mar. 1974.

[9] L. Papke and P. Robertson, “Improved decoding with the SOVA in a parallel concatenated (turbo-code) scheme,” *Proc. IEEE Int. Conf. Commun.*, pp. 102–106, June 1996. (Dallas, TX).

[10] J. Hagenauer and P. Hoeher, “A Viterbi algorithm with soft-decision outputs and its applications,” *Proc. IEEE Globecom*, pp. 1680–1686, Nov. 1989. (Dallas, Texas).

[11] J. Vogt and A. Finger, “Improving the max-log-map turbo decoder,” *Electronics Letters*, vol. 36, pp. 1937–1939, Nov. 2000.

[12] K. Gracie, S. Crozier, and A. Hunt, “Performance of a low-complexity turbo decoder with a simple early stopping criterion implemented on a SHARC processor,” *Proc. Sixth Int. Mobile Satellite Conf.*, pp. 281–286, June 1999. (Ottawa, Canada).