CONTROLLED COMPLEXITY MAP DECODING OF CABAC ENCODED DATA

Salma Ben Jamaa, Michel Kieffer and Pierre Duhamel

LSS – CNRS – Supélec – Université Paris-Sud XI
Plateau de Moulon, 91192 Gif-sur-Yvette, France

ABSTRACT

In this paper, we present a joint source-channel decoding technique based on exact MAP estimation for data encoded by CABAC (Context-based Adaptive Binary Arithmetic Coding) in standards like H.264/AVC. Soft decoding is put at work using an improved sequential decoding technique, achieving a trade-off between complexity and efficiency of the proposed algorithms. Error detection is realized by exploiting the binarization scheme and redundancy left in the code string, so that CABAC compression efficiency is preserved and no additional redundancy is compulsory.

1. INTRODUCTION

To overcome noise and packet loss problems occurring in data transmission through noisy packet-switched channels, *e.g.*, wireless Internet, one popular approach is to use joint source-channel (JSC) decoding. Such decoding techniques keep efficient data compression and improve robustness against errors. A significant part of the work in JSC decoding is related to the reliable decoding of variable length codes (VLC), see, *e.g.*, [1].

Arithmetic Coding (AC) [2] is currently the object of a growing interest as it yields higher compression efficiency when compared to other compression methods. This is why AC has been incorporated, *e.g.*, in JPEG 2000 and H264/AVC [3]. However, this compression efficiency makes AC particularly vulnerable to transmission errors. This point has motivated the recent development of JSC decoding techniques for AC encoded data [4–8]. The error detection and correction are usually performed by introducing redundancy in the compressed bitstream, thus reducing the compression efficiency. In [9], a *forbidden symbol* (FS) is introduced in the coding alphabet and used to detect errors. This technique is coupled with ARQ in [10]. In [7], both depth first and breadth first decoding algorithms have been considered. Error detection is again achieved with the help of a FS and error correction is performed via maximum-likelihood (ML) estimation. Sequential decoding with MAP estimation of the encoded sequence is combined with the use of a FS in [5]. The AC decoder proposed in [6] estimates the state transitions of a Markov process modeling the AC. Redundancy is introduced by considering a reduced precision AC and adding synchronization markers before the encoding process.

The CABAC [11], used in the H.264/AVC standard, supplements the AC with an adaptive binary encoding technique based on context modeling. An improved compression efficiency is achieved especially in presence of non-stationary sources. Nevertheless, as CABAC uses binary AC and precomputed probabilities, techniques for robust decoding of VLC encoded data cannot be applied unless important changes of the CABAC are performed.

This work exploits previous results on exact MAP estimation of CABAC encoded data [8]. Here, MAP estimation is performed by sequential decoding algorithms [12]. As improving error resilience performances usually goes with increasing decoding complexity, an objective test based on the exact MAP metric is proposed to allow an adjustment of the trade-off between complexity and efficiency.

2. CONTEXT OF THE WORK AND NOTATIONS

Transmission scheme

Data are assumed to be compressed by a CABAC encoder, packetized and transmitted over a radio mobile channel. Packets undergo some alteration during the transmission. The purpose is to detect and correct transmission errors. As CABAC handles only binary data, a binarization step converting non-binary information into binary source symbols according to a *binarization scheme* [11] is needed. Soft estimates are obtained from the output of the channel and fed to the CABAC decoder.

On Figure 1, the sequence $S_1^K = \{S_1, ..., S_K\}$ consists of a succession of binarized source symbols belonging to a set of binary words. *Bins* stand for bits obtained by the binarization process. The last binarized source symbol of S_1^K, EOS (for End Of Sequence), indicates the end of the binarized stream. The succession of CABAC output bits, $X_1^N = \{X_1, ..., X_N\}$ is mapped using BPSK signaling into R_1^N with $R_i = \pm\sqrt{E_b}$. R_1^N is assumed to be transmitted within a single channel packet. Finally, the channel output is $Y_1^N = \{Y_1, ..., Y_N\}$. Only the transmitted bitstream length N is supposed to be known at the decoder side. Capital letters are for random variables, and small letters for their values. Integers n and k denote respectively the current length of the decoder input bitstream, and the current number of decoded bins.

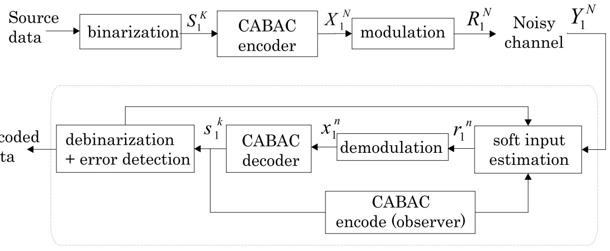

Fig. 1. Transmission scheme

Binary arithmetic coding (BAC)

Basic principles of BAC are explained in [13] and recalled here. At each iteration, a subinterval of $[0, 1)$ is iteratively constructed.

The current interval $[low, low+range)$ is divided into two subintervals, the size of which is proportional to the probabilities of the source bins 0 and 1 and one of these intervals is selected, depending on the value of the current bin. Once the last bin of EOS is encoded, the algorithm computes the real value V, belonging to the obtained interval which can be represented by the minimum number of bits. The binary representation of V forms the *code string*. Based on V, the decoder is able to recover the source bin stream.

For long source sequences, subintervals may get too small to be accurately handled by a finite precision processor. To overcome this precision problem, most AC encoders are implemented using integers [11]. The interval $[0, 1)$ of reals is then replaced by the interval $[0, 2^p)$ of integers, where p is the bit-size of low and $range$. Each time a division and selection has been performed, one checks whether $range$ is smaller than the quarter of $[0, 2^p)$. If it is the case, renormalization, consisting in doubling low and $range$, is performed and some encoded bits may be output. If the current interval (before renormalization) overlaps the midpoint of $[0, 2^p)$, no bit is output. The number of times low and $range$ have been doubled without outputting encoded bits is stored in the $follow$ variable. If the current interval (before renormalization) entirely lies in the upper or lower half of $[0, 2^p)$, the encoder emits the leading bit of low (0 or 1) and $follow$ opposite bits (1 or 0). This is called the *follow-on* procedure [14].

3. MAP ESTIMATION

The encoded data is estimated using the *exact* MAP estimate proposed in [8], briefly recalled here. The aim is to evaluate, at any $n \leqslant N$, the sequence $\widehat{x}_1^n$ maximizing the *a posteriori* probability (APP) given by $P(X_1^n = x_1^n | Y_1^n = y_1^n)$, written as $P(x_1^n|y_1^n)$ to make notations shorter.

The decoder input sequence x_1^n is decomposed into three parts. First, $x_1^{n'(s_1^k)}$ is the code string of $n'(s_1^k)$ bits that would be emitted by an encoder fed with s_1^k. Second, $x_{n'(s_1^k)+1}^{n'(s_1^k)+F(s_1^k)}$ are the *postponed bits* [4], which would be emitted by an encoder fed with s_1^k if the encoder emits bits when encoding the $k+1-$th bin. The postponed bits may only take values $\{1, 0, \dots, 0\}$ or $\{0, 1, \dots, 1\}$. The last part of x_1^n, $x_{n'(s_1^k)+F(s_1^k)+1}^n$, are bits assumed independent of s_1^k.

The channel is assumed memoryless. Variables $X_1^{n'}$, $X_{n'+1}^{n'+F}$, and $X_{n'+F+1}^n$ are assumed independent. As a consequence, it is also the case for $Y_1^{n'}$, $Y_{n'+1}^{n'+F}$, and $Y_{n'+F+1}^n$. Moreover, as X_1^N is the output of an arithmetic encoder, $\forall i = 1 \dots N, P(X_i = 0) = P(X_i = 1)$. Under these assumptions, the APP is expressed as

$$P(x_1^n|y_1^n) = P\left(s_1^k\right) \frac{P\left(y_1^n|x_1^n\right)}{P\left(y_1^n\right)} P(x_{n'+F+1}^n) P(x_{n'+1}^{n'+F}|s_1^k). \quad (1)$$

4. SEQUENTIAL DECODING

In order to compute sequence maximizing (1), $P(x_1^n|y_1^n)$ has to be evaluated for all possible x_1^n. Nevertheless, to an observation Y_1^n, one may assign a decoding tree with up to 2^n paths representing possible values of the code string. For relatively large n, examining the whole decoding tree is infeasible. The purpose of sequential decoding is to find the best path, according to a given metric, without examining too many branches. The most popular sequential decoders are the stack algorithm (SA) [15, 16], the M-algorithm (MA) [12] and their derivatives, such as the *generalized stack algorithm* [17]. The proposed sequential decoders use the logarithm of (1) as a metric, denoted by $\mathcal{M}_{MAP}\left(x_1^n\right)$, allowing to sort the stored paths. In addition, drop and stop conditions, relying on some constraints which have to be satisfied by the code string and the decoded sequence, are employed.

Drop conditions

Drop conditions allow the sequential decoders to identify paths not deserving to be further examined. A first drop condition results from the complexity constraint imposed by the sequential decoders. The number of simultaneously stored paths is limited. When the limit is reached, paths with the worst metrics are dropped. If the correct path is dropped in this way, the decoding fails and decoder may not output any solution. This event is called *erasure*.

Using the fact that the code string is N bits long and that the last binarized symbol is the EOS, one may derive two drop conditions. First, if a n length bitstream x_1^n leads to s_1^k containing an EOS while $n < N$, then x_1^n is dropped. Second, if a N length bitstream x_1^N leads to s_1^k not ending with EOS, x_1^N is then dropped. In both cases, paths are dropped with no risk of erasure.

Paths may also be dropped if they have a null APP, as they are very likely to be automatically discarded when explored paths are sorted and the constraint on the maximum number of stored paths holds. First, due to the incomplete binarization scheme (*e.g.*, zero order Exp-Golomb codes [11]), all sequences of bins are not necessarily valid, thus $P(s_1^k)$ may be equal to zero and lead to a null APP. Second, if the postponed bits are neither equal to $\{1, 0, \dots, 0\}$ nor to $\{0, 1, \dots, 1\}$, then $P(x_{n'+1}^{n'+F}) = 0$ leading again to a null APP.

Stop conditions

The decoding tree exploration stops when a N bits path x_1^N yields a sequence of binarized symbols s_1^K ending with EOS and satisfying the binarization scheme. Then, x_1^N is the path maximizing $\mathcal{M}_{MAP}(x_1^N)$ among all the stored paths at the current iteration. In some cases, decoding stops if all paths have been dropped, or if the maximum allowed computational effort has been reached. Both of these situations may lead to erasure. The lost packet may be re-emitted if ARQ is implemented.

4.1. Objective adjustment of the efficiency-complexity trade-off

The decoding performances in terms of error resilience efficiency are improved as more paths are explored in the decoding tree, especially when the beginning of the code string is strongly corrupted. Nevertheless, this improvement usually goes with increasing decoding complexity. Therefore, in order to judiciously adjust this trade-off between the complexity and efficiency, an objective test corresponding to a new drop condition is proposed.

Assume that the path to be extended is x_1^{n-1}. Two extensions can be considered, $x_n = 1$ and $x_n = 0$. The hard decoder discards systematically one extension and thus, explores only one path on the decoding tree. The idea is to derive a test allowing to decide if both of the two choices deserve being considered or if a hard decision on the current bit is sufficient. In [7], a 2Δ width *null zone* is used to determine whether a hard decision on y_n is reliable.

In our case, we consider that if a path has a relatively low metric, it is very likely to be dropped in the next iterations. Thus, omitting exploring and storing such paths saves computational effort. The idea here is to derive an objective criterion in order to characterize a *low* metric for a controlled amount of decision errors.

At the node corresponding to x_1^{n-1} on the decoding tree, three actions may be taken:

- A_0^n: only the branch corresponding to $x_n = 0$ is explored.
- A_1^n: only the branch corresponding to $x_n = 1$ is explored.
- $A_{0|1}^n$: both branches are explored.

Let Λ_n be the logarithm of the APP ratio

$$\Lambda_n = \mathcal{M}_{\text{MAP}}(x_1^{n-1}, 1) - \mathcal{M}_{\text{MAP}}(x_1^{n-1}, 0).$$

The purpose is to derive a decision threshold T such that

$$\Lambda_n \underset{A_1^n}{\overset{A_{0|1}^n}{\lesseqgtr}} T \text{ and } -\Lambda_n \underset{A_{0|1}^n}{\overset{A_0^n}{\lesseqgtr}} -T.$$

According to (1) and assuming X_n independent of X_1^{n-1}, one gets

$$\begin{aligned}\Lambda_n &= \log P(x_1^{n-1}, X_n = 1|y_1^n) - \log P(x_1^{n-1}, X_n = 0|y_1^n) \\ &= \log P(y_n|X_n = 1) - \log P(y_n|X_n = 0). \qquad (2)\end{aligned}$$

Let $P_e = P(A_0^n, X_n = 1) + P(A_1^n, X_n = 0)$ be the probability of error, *i.e.*, the probability of loosing the correct path. Using the assumption that $P(X_n = 1) = P(X_n = 0) = \frac{1}{2}$, P_e may be expressed as

$$P_e = \frac{1}{2} P(\Lambda_n < -T|X_n = 1) + \frac{1}{2} P(\Lambda_n > T|X_n = 0).$$

Let $P_{AT} = P(A_{0|1}, X_n = 1) + P(A_{0|1}, X_n = 0)$ be the probability of unnecessary additional treatment, *i.e.*, the probability of extending a path with two branches, resulting in a complexity increase. One gets

$$\begin{aligned}P_{AT} &= \frac{1}{2} P(-T < \Lambda_n < T|X_n = 1) \\ &+ \frac{1}{2} P(-T < \Lambda_n < T|X_n = 0). \qquad (3)\end{aligned}$$

The idea to obtain T is to minimize P_{AT} for a fixed P_e. The threshold T is derived assuming that the channel is zero-mean AWGN, of variance σ^2. For other kinds of channels, such as an UMTS channel, an equivalent AWGN channel may be estimated, and the derivation of an equivalent objective test is straightforward. For AWGN channel, (2) is written as

$$\Lambda_n = \frac{2\sqrt{E_b}}{\sigma^2} y_n.$$

Thus, P_e and P_{AT} may be expressed as

$$\begin{aligned}P_e &= \frac{1}{2}(1 - \text{erf}(\frac{\frac{-T\sigma^2}{2\sqrt{E_b}} + \sqrt{E_b}}{\sigma\sqrt{2}})). \\ P_{AT} &= \text{erf}(\frac{\frac{T\sigma^2}{2\sqrt{E_b}} + \sqrt{E_b}}{\sigma\sqrt{2}}) + \text{erf}(\frac{\frac{T\sigma^2}{2\sqrt{E_b}} - \sqrt{E_b}}{\sigma\sqrt{2}}).\end{aligned}$$

where $\text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt$.

One can finally express the threshold T as

$$T(\sigma, P_e) = \frac{2\sqrt{E_b}}{\sigma^2} \left(\left(\sigma\sqrt{2}\text{erf}^{-1}(1 - P_e) \right) - \sqrt{E_b} \right). \qquad (4)$$

Several choices may be considered for P_e. Here, we constrain P_e to be a fraction of the probability $P_{\text{Hard}}(\sigma)$ that the hard decoder locally fails, expressed in the AWGN case by

$$\begin{aligned}P_{\text{Hard}}(\sigma) &= \int_0^{+\infty} P(y|X_n = 0)dy + \int_{-\infty}^0 P(y|X_n = 1)dy \\ &= 1 - \text{erf}(\frac{\sqrt{E_b}}{\sigma\sqrt{2}}). \qquad (5)\end{aligned}$$

Then, P_e can be expressed as

$$P_e(\sigma, n) = \alpha \cdot P_{\text{Hard}}(\sigma, n), \text{with } \alpha < 1. \qquad (6)$$

The parameter α allows one to adjust the decoding complexity-efficiency trade-off, as shown in Section 5.

5. SIMULATIONS

Simulations are performed using the CABAC defined in the H.264 standard. Binarized source symbols belong to the first 9 binary codewords of the zero-order Exp-Golomb scheme (EG0) [11]. A simplified context modeling with three contexts is considered. The Symbol Error Sate (SER) is evaluated for different values of the Signal to Noise Ratio (E_b/N_0). When an erasure occurs, the decoder may not output any solution and all symbols emitted by the source are considered as erroneous, and counted in the SER. Hard decoding provides the bit value x_n using the sign of the channel output y_n. Hard decoding fails if debinarization fails or if the EOS is not decoded from this bitstream.

MAP decoders based on both SA and MA are considered, and the following abbreviations are adopted. BSA and BMA(M) stand for the basic SA and MA, M being the number of paths kept at each MA iteration. GSA(ℓ) is the generalized SA, extending ℓ paths at each iteration. Finally FGSA(ℓ, α) and FMA(M, α) denote the Fast SA and the Fast MA embedding the test presented in Section 4.1. In all the SA based decoders, the maximum number of simultaneously stored paths is 10000.

Figure 2 illustrates the error resilience, in terms of SER, and complexity performances, in terms of average number of visited branches during erasure-free decoding, corresponding to the hard decoder and four versions of the decoder using the MA. Figure 3 depicts the same performances for decoders using the SA.

When compared to a standard decoder carrying out hard decisions on noisy bits, sequential decoders present an important gain in terms of error resilience: up to 4 dB for the MA based decoders and 3 dB for the SA. These performances are improved as the number of simultaneously explored paths (M and ℓ) increases, and as α decreases. On the other hand, one can notice that the more decoding is efficient in recovering errors, the higher is the complexity. At SER = 10^{-3}, FMA($20, 10^{-4}$) presents 20 times less complexity when compared to MA(10), with a gain of 2.5 dB. For the same SNR, compared to the BSA, the FGSA($3, 10^{-4}$) reaches a gain of 1.8 dB, at SER=10^{-3} for a doubled complexity at 12 dB.

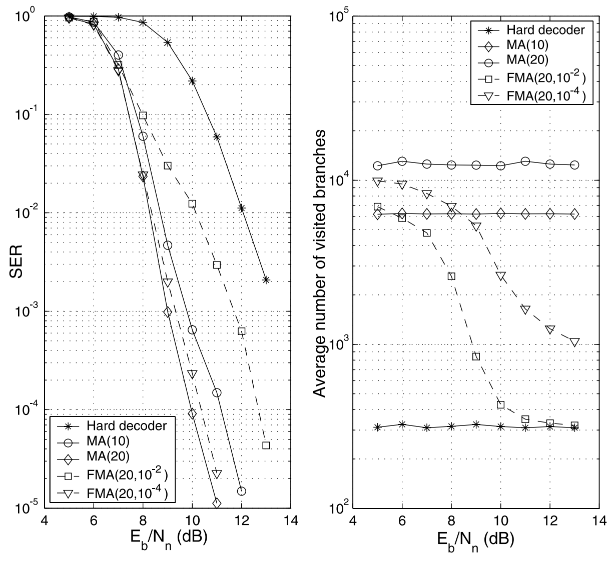

Fig. 2. Error resilience and complexity performances of the decoder using the MA

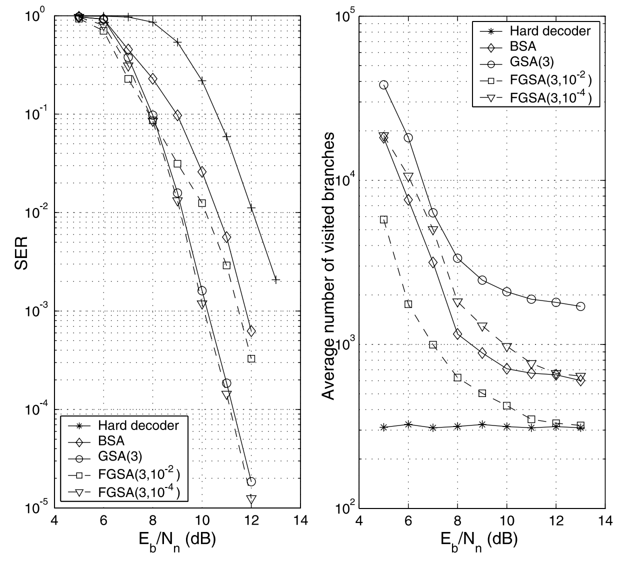

Fig. 3. Error resilience and complexity performances of the decoder using the SA

6. CONCLUSIONS

The main drawback of the high compression rate CABAC is its vulnerability to transmission errors. In this paper, we have presented a soft CABAC decoding technique based on an exact MAP estimation associated to sequential decoders. An objective test allowing to adjust the trade-off between decoding complexity and error resilience performances is proposed. Current work is dedicated to embedding the soft MAP decoder and the objective test within the H.264/AVC decoder.

7. REFERENCES

[1] M. Park and D. J. Miller, “Joint source-channel decoding for variable length encoded data by exact and approximate MAP sequence estimation,” *IEEE Trans. on Comm.*, vol. 48(1), pp. 1–6, 2000.

[2] P. G. Howard and J. S. Vitter, “Practical implementations of arithmetic coding,” *Image and Text Compression*, vol. 13(7), pp. 85–112, 1992.

[3] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, “Overview of the H.264/AVC video coding standard,” *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13(7), pp. 560–576, 2003.

[4] J. Sayir, *On Coding by Probability Transformation*, PhD Thesis Nr. 13099, EE Department, ETH Zurich, Switzerland, 1999.

[5] M. Grangetto, P. Cosman, and G. Olmo, “Joint source/channel coding and MAP decoding of arithmetic codes,” *IEEE Trans. on Comm.*, vol. 53(6), pp. 1007–1016, 2005.

[6] T. Guionnet and C. Guillemot, “Soft decoding and synchronization of arithmetic codes: Application to image transmission over noisy channels,” *IEEE Trans. on Image Processing*, vol. 12(12), pp. 1599–1609, 2003.

[7] B. D. Pettijohn, M. W. Hoffman, and K Sayood, “Joint source/channel coding using arithmetic codes,” *IEEE Trans. on Comm.*, vol. 49(5), pp. 826–836, 2001.

[8] S. Ben Jamaa, M. Kieffer, and P. Duhamel, “Exact map decoding of cabac encoded data,” *submitted to Proceedings of ICASSP*, 2006.

[9] C. Boyd, J. Cleary, I. Irvine, I. Rinsma-Melchert, and I. Witten, “Integrating error detection into arithmetic coding,” *IEEE Trans. on Comm.*, vol. 45(1), pp. 1–3, 1997.

[10] J. Chou and K. Ramchandran, “Arithmetic coding-based continuous error detection for efficient ARQ-based image transmission,” *IEEE Trans. on Comm.*, vol. 18(6), pp. 861–867, 2000.

[11] D. Marpe, H. Schwarz, and T Weigand, “Context based adaptative binary arithmetic coding in the H.264/AVC video compression standard,” *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13(7), pp. 620–636, July 2003.

[12] J. B. Anderson and S. Mohan, *Source and channel coding: an algorithmic approach*, Kluwer Academic Publishers, Norwell, MA, 1991.

[13] P. Elias, “Universal codeword sets and representations of the integers,” *IEEE Trans. on Information Theory*, vol. 6(3), pp. 194–203, 1975.

[14] I. H. Witten, R. M. Neal, and J. G. Cleary, “Arithmetic coding for data compression,” *Communications of the ACM*, vol. 30(6), pp. 520–540, 1987.

[15] K. Zigangirov, “Some sequential decoding procedures,” *Probl. Peredach. Inform.*, vol. 2(4), pp. 13–15, 1966.

[16] F. Jelinek, “A fast sequential decoding algorithm using a stack,” *IBM J. Res. Develop.*, vol. 13, pp. 675–685, 1969.

[17] D. Haccoun and M. J. Ferguson, “Generalized stack algorithms for decoding convolutional codes,” *IEEE Trans. on Information Theory*, vol. 21(6), pp. 638–651, 1975.