EXACT 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

This paper presents a MAP estimator for CABAC encoded data transmitted through a noisy channel. The decoding process has two characteristics (i) it provides an exact result, without approximation (ii) it is compatible with realistic implementations of CABAC in standards like H.264, *i.e*, taking into account finite precision problems and handling adaptive probabilities and context modeling. Simulation results outline the efficiency of the proposed method when applied in AWGN and UMTS-OFDM frameworks.[1]

I. INTRODUCTION

An increasing number of applications require data transmission through noisy packet-switched channels, *e.g.*, wireless Internet. To overcome noise and packet loss problems, an increasingly popular approach is joint source-channel decoding (JSCD), which improves decoding performances while keeping efficient data compression. A significant part of the work in JSCD is related to the reliable decoding of variable length codes (VLC), see, *e.g.*, [1], [2].

Arithmetic Coding (AC) [3] is currently the object of a growing interest as it yields higher compression efficiency when compared to other compression methods. Context-based Adaptive Binary Arithmetic Coding (CABAC) [4] is the entropy coding method used, *e.g.*, in H264/AVC. By combining an adaptive binary encoding technique with context modeling, an improved compression efficiency is achieved especially in presence of non-stationary sources. However, this high compression efficiency makes CABAC particularly vulnerable to transmission errors. The error detection and correction are usually performed by introducing redundancy in the compressed bitstream, thus reducing the compression efficiency. In [5], a *forbidden symbol* (FS) is introduced in the coding alphabet, used as an error detection device. This technique is coupled with ARQ in [6]. In [7], both depth first and breadth first decoding algorithms have been considered, and error detection is again achieved by testing the presence of a FS during the decoding process. Sequential decoding with MAP estimation of the encoded sequence is combined with the use of a FS in [8]. The AC decoder proposed in [9] estimates the state transitions of a Markov process modeling the encoder. Redundancy is introduced by considering a reduced precision AC and adding synchronization markers into the transmitted bitstream.

[1]This work has been partly supported by the NEWCOM NoE

This work deals with the MAP estimation of H.264/AVC–CABAC [4] encoded data. Its purpose is to provide an *exact* MAP metric which is adapted to an actual CABAC implementation, *i.e.*, a metric taking into account the delays due to the CABAC encoder and decoder buffers. To obtain this result, an estimator of the CABAC encoder state (*observer*) is required at decoder. No extra redundancy is compulsory, only that introduced during the binarization step of the CABAC is exploited to detect errors. Our decoder performs better than the one derived using the metric proposed in [8], [9], as their metric would only be approximate on a realistic CABAC. Note, however, that this exact computation is compatible with existing ways of adding redundancy, and that combination of both strategies would result in further improvements. However, this is not the purpose of the paper.

II. CONTEXT OF THE WORK

Data are assumed to be compressed by a CABAC encoder, packetized and transmitted over a mobile channel. Packets undergo some alteration during the transmission. The purpose is then to detect and correct transmission errors. As CABAC handles only binary data, a binarization step consisting in converting non binary source information into binary words according to a *binarization scheme* [4] is needed. Soft estimates are obtained from the output of the channel and fed to the CABAC decoder, see Figure 1.

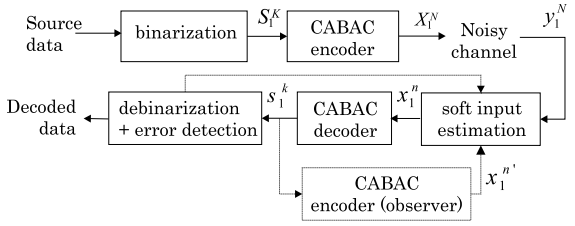

Fig. 1. Transmission scheme

The K bins sequence denoted by $S_1^K = \{S_1, ..., S_K\}$ consists of a succession of binarized source symbols belonging to the set $\mathcal{C}$ (*bins* stand for bits obtained by a binarization process). The last binarized source symbol of S_1^K, EOS, indicates the end of the binarized stream. Let $X_1^N = \{X_1, ..., X_N\}$ be the succession of CABAC output bits, assumed to be transmitted within a single channel packet, and let $Y_1^N = \{Y_1, ..., Y_N\}$ be the corresponding channel output.

Only 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 channel input bitstream, and the current number of decoded bins.

II-A. Basic principles of binary arithmetic coding (BAC)

Basic principles of BAC are explained in [10] 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 sources with unbalanced bin probabilities, and for long source sequences, subintervals may get too small to be accurately handled by a finite precision processor. This problem is solved with finite precision BAC.

II-B. Practical implementations of BAC

To overcome the precision problem, most AC encoders are implemented using integers [4]. Since this process is of interest in our implementation, the corresponding algorithm is recalled below.

The interval $[0, 1)$ of reals is 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 [11].

At decoding side, a p bits buffer $value$ formed by the last p received bits is used to determine the divisions and selections of $[0, 2^p)$ which have been performed at encoder side. These operations clearly characterize the source bin stream. Each time a renormalization is performed, new bits are shifted into $value$ without outputting any estimate of the source bins. Moreover decoding starts only after an initialization consisting in loading p bits into $value$; again no source bin is output during initialization.

In the context-based BAC, interval divisions are made according to bins probabilities deduced from contexts. Selection depends on whether the current bin is the most probable (MPS) or the least probable (LPS) one. The context information (probabilities, value of the MPS) is updated each time a bin has been encoded. Encoding and decoding complexity is reduced by using a set of precomputed discrete values of subinterval lengths, as performed within the Q coder [4].

III. MAP DECODER

Assume that for a given n, x_1^n fed to the decoder results in s_1^k at its output. The MAP estimator proposed by [9] and [8] consists in evaluating

$$\begin{aligned} \hat{s}_1^k &= \arg\max_{s_1^k} P\left(S_1^k = s_1^k | Y_1^n = y_1^n\right) \qquad (1) \\ &= \arg\max_{s_1^k} \frac{P\left(S_1^k = s_1^k\right) P\left(Y_1^n = y_1^n | S_1^k = s_1^k\right)}{P\left(Y_1^n = y_1^n\right)}. \end{aligned}$$

The difficulty in using this last formula lies in the evaluation of $P\left(Y_1^n = y_1^n | S_1^k = s_1^k\right)$. In the above referenced papers, this difficulty is circumvented by approximating this quantity by $P\left(Y_1^n = y_1^n | X_1^n = x_1^n\right)$. This corresponds to implicitly assuming that $P(S_1^k = s_1^k | X_1^n = \tilde{x}_1^n) = 0$ for any $\tilde{x}_1^n \neq x_1^n$, *i.e.*, assuming that x_1^n is the only bit sequence leading to s_1^k at decoder output. However, as explained in Section II, the *value* buffer introduces some decoding delay and several decoder input subsequences may lead to the same output subsequence s_1^k. Only when N is reached, a one to one mapping is obtained by ensuring that the buffers are empty. This make the former MAP estimate suboptimal within an actual CABAC implementation. In [12], Sayir proposes a MAP estimation of noisy arithmetic encoded data that has inspired this work, though the decoding delay occurring at the CABAC decoder is not taken into account. In our work, a different definition of the MAP estimate is adopted. As will be shown, this requires the estimation of the encoder state (low, $range$, $follow$, and contexts) at the decoder side.

III-A. Definitions and assumptions

The aim is to determine at any $n \leqslant N$, the sequence $\hat{x}_1^n$ maximizing the *a posteriori* probability (APP)

$$P(X_1^n = x_1^n | Y_1^n = y_1^n), \qquad (2)$$

which will be written as $P(x_1^n | y_1^n)$ to make notations shorter. The evaluation of (2) has account for the fact that x_1^n is the output of a CABAC encoder fed with binarized source symbols. Once N is reached, $\hat{x}_1^N$, maximizing (2), may be decoded as $\hat{s}_1^K$ satisfying all constraints imposed by the binarization. As there is a one to one mapping between $\hat{x}_1^N$ and $\hat{s}_1^K$, $\hat{s}_1^K$ maximizes (1). Using (1) or (2) with an optimal decoder would lead to the same final result, however, as optimal decoding is usually unfeasible, the evaluation of (2) is more accurate with iterative decoders of noisy data encoded with a realistic CABAC.

The decoder input sequence x_1^n may be decomposed in three parts. First, $x_1^{n'(s_1^k)}$ are the bits that would (in any case, with any value taken by the rest of the sequence) be output by an encoder fed with s_1^k. The second part $x_{n'(s_1^k)+1}^{n'(s_1^k)+F(s_1^k)}$ are the *postponed bits* [12], which may only take values $\{1, 0, \ldots, 0\}$ or $\{0, 1, \ldots, 1\}$, depending on the coder internal state (low,

$range$, $follow$, contexts) after being fed with s_1^k. The last part, $x_{n'(s_1^k)+F(s_1^k)+1}^n$, are bits assumed independent of s_1^k. Here, $F(s_1^k)$ is chosen equal to $follow + 1$, keeping in mind that $follow$ depends on s_1^k. As $n'(s_1^k)$, $F(s_1^k)$, low, $range$ and the contexts are not directly available at the decoder, a possible way to estimate these quantities is to *reencode* s_1^k. This additional encoder is called *observer* to avoid confusion, see Figure 1. In the remainder of the paper, the notations $n'(s_1^k) = n'$ and $F(s_1^k) = F$ are adopted.

We assume that the channel is memoryless and that $X_1^{n'}$, $X_{n'+1}^{n'+F}$, and $X_{n'+F+1}^n$ are independent. As a consequence, it is also the case for $Y_1^{n'}$, $Y_{n'+1}^{n'+F}$, and $Y_{n'+F+1}^n$.

III-B. Derivation of the APP

Since s_1^k is fully determined by x_1^n at the decoder side and since s_1^k yields $x_1^{n'}$ at the observer side, one can easily prove that

$$P(X_1^n = x_1^n) = P(S_1^k = s_1^k, X_{n'+1}^n = x_{n'+1}^n). \quad (3)$$

Using (3), the APP can be expressed as

$$P(x_1^n|y_1^n) = \frac{P(s_1^k)P(x_{n'+1}^n|s_1^k)}{P(y_1^n)} P(y_1^n|s_1^k, x_{n'+1}^n). \quad (4)$$

Using the decomposition of x_1^n presented in Section III-A and the independence between $X_{n'+F+1}^n$ and S_1^k, one may write

$$\begin{aligned} P\left(x_{n'+1}^n|s_1^k\right) &= P(x_{n'+1}^{n'+F}, x_{n'+F+1}^n|s_1^k) \\ &= P(x_{n'+1}^{n'+F}|s_1^k)P(x_{n'+F+1}^n). \end{aligned} \quad (5)$$

Using again (3), the last term in (4) may be written as

$$P(y_1^n|s_1^k, x_{n'+1}^n) = P(y_1^n|x_1^n). \quad (6)$$

Then, combining (4) and (6) and using (5), the APP becomes

$$P(x_1^n|y_1^n) = \frac{P\left(s_1^k\right) P\left(y_1^n|x_1^n\right)}{P\left(y_1^n\right)} P(x_{n'+F+1}^n) P(x_{n'+1}^{n'+F}|s_1^k). \quad (7)$$

By observing (7), it appears that the metric obtained in [8], [9] is part of our result. Two additional terms are involved; $P(x_{n'+F+1}^n)$ and $P(x_{n'+1}^{n'+F}|s_1^k)$, which evaluation is described in the next section. The other quantities are estimated as in [8].

III-C. Practical implementation of the MAP estimator

To evaluate $P(x_{n'+F+1}^n)$, we assume that all sequences $x_{n'+F+1}^n$ have equal *a priori* probability, *i.e.*, $P(x_{n'+F+1}^n) = 2^{-n+(n'+F+1)}$. If a different information is available (iterative decoding, for example), it may obviously be used.

The evaluation of $P(x_{n'+1}^{n'+F}|s_1^k)$ requires more care. More bins $s_{k+1}^{k'}$ are necessary to perfectly determine $x_{n'+1}^{n'+F}$ at the observer. A possible way to estimate $P(x_{n'+1}^{n'+F}|s_1^k)$ is to write this quantity as

$$P(x_{n'+1}^{n'+F}|s_1^k) = \sum_{s_{k+1}^{k'}} P\left(s_{k+1}^{k'}|s_1^k\right) P(x_{n'+1}^{n'+F}|s_{k+1}^{k'}, s_1^k),$$

which requires to feed the observer with all possible $s_{k+1}^{k'}$ until a follow-on procedure is performed. This procedure has a combinatorial complexity. Here, the procedure for a single additional bin s_{k+1} is described, its generalization is straightforward. Two cases have to be considered $S_{k+1} = \text{MPS}$ and $S_{k+1} = \text{LPS}$. Thus,

$$\begin{aligned} P(x_{n'+1}^{n'+F}|s_1^k) &= P_{\text{MPS}} P(x_{n'+1}^{n'+F}|s_{k+1} = \text{MPS}, s_1^k) \\ &+ P_{\text{LPS}} P(x_{n'+1}^{n'+F}|s_{k+1} = \text{LPS}, s_1^k), \end{aligned}$$

where $P_{\text{MPS}} = P(S_{k+1} = \text{MPS}|s_1^k)$ and $P_{\text{LPS}} = P(S_{k+1} = \text{LPS}|s_1^k)$ are deduced from the binarization scheme and the contexts estimated at the observer. If postponed bits are emitted, their values may be either $\{0, 1, \ldots, 1\}$ or $\{1, 0, \ldots, 0\}$. If $x_{n'+1}^{n'+F}$ is not equal to one of these two sequences, $P(x_{n'+1}^{n'+F}|s_1^k) = 0$. Now, if $x_{n'+1}^{n'+F}$ is equal to one of them, and

- only an MPS produces $x_{n'+1}^{n'+F}$ then $P(x_{n'+1}^{n'+F}|s_1^k) = P_{\text{MPS}}$,
- only an LPS produces $x_{n'+1}^{n'+F}$ then $P(x_{n'+1}^{n'+F}|s_1^k) = P_{\text{LPS}}$,
- both MPS and LPS produce $x_{n'+1}^{n'+F}$ then $P(x_{n'+1}^{n'+F}|s_1^k) = 1$.

In all other cases, it is assumed that $P(x_{n'+1}^{n'+F}|s_1^k) = \frac{1}{2}$.

Practical implementations could fed the observer with more than a single bit to improve the evaluation of $P(x_{n'+1}^{n'+F}|s_1^k)$. In practical situations, however, satisfactory results are obtained with less than 3 additional bits, and the additional complexity remains limited.

IV. SIMULATIONS

Simulation are performed using the CABAC taken from the H.264. Binarized source codewords belong to the first 8 binary codewords of the zero-order Exp-Golomb scheme (EG0) [4]. A simplified context modeling with three contexts is considered. Simulations using AWGN channel and Pedestrian-B UMTS soft error patterns, see, *e.g.*, [13] are considered. For AWGN channel, no channel coding is used and error correction relies only on the redundancy due to the binarization scheme. For the Pedestrian-B channel, the CABAC is followed by a rate $1/2$ convolutionnal code with constraint length 9 and generators $(561, 753)_o$. A standard UMTS interleaver of length 640 is also considered. At the channel output, a SOVA decoder is implemented, providing log-likelihood ratios (LLR_n). In both cases, a sequential M-algorithm [14] explores and stores the M best estimates of X_1^n in terms of APP (2). The Symbol Error Sate (SER) is evaluated for different values of the SNR, symbols being the binarized source words. When the correct *path* is lost by the M-algorithm, 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 from the sign of channel output y_n in the AWGN case, and of the LLR_n in the UMTS case. Hard decoding fails if debinarization fails or if the EOS is not decoded from this bit stream.

Figure 2 compares the results obtained using the proposed MAP and the metric proposed by [8]. Packets of 640 bits, containing sequences involving many *follow-on* procedures

during encoding, are transmitted over an AWGN channel. A M-algorithm with $M = 10$ is used for decoding. For a SER of 10^{-3}, an improvement of 0.6 dB is achieved.

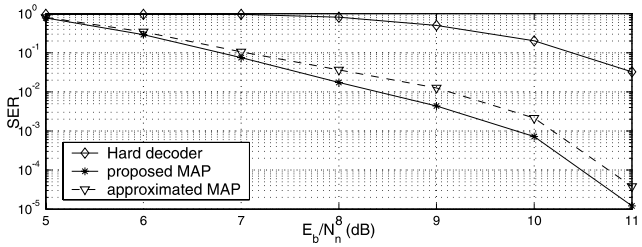

Fig. 2. Performances of the proposed MAP vs. the MAP proposed in [8]

Figure 3 illustrates the results obtained using packets of 640 bits and the M-algorithm with $M = 20$. Performances of a MAP decoder using (2) are compared to those obtained using a Maximum-Likelihood (ML) decoder. For a SER of 10^{-3}, the improvement (in dB) obtained when using (2), compared to the ML metric is around 0.7 dB for both channels. The gain achieved by the soft decoding when compared to the hard one is up to 4 dB in the AWGN channel and 2 dB in the Pedestrian-B channel.

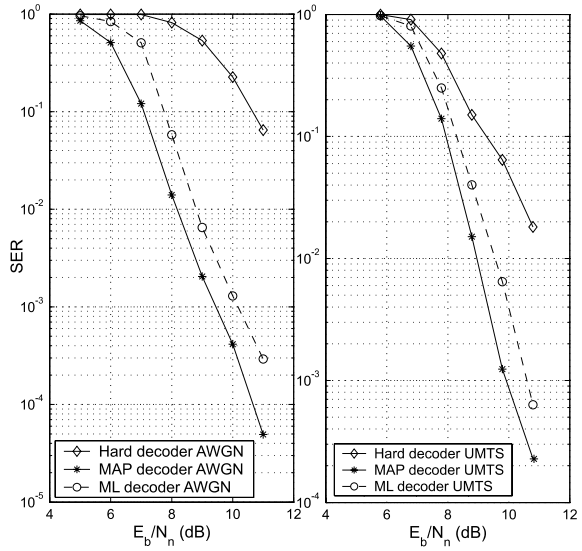

Fig. 3. MAP performances vs. hard and ML decoding for AWGN channel (without channel coding), and UMTS-OFDM channel (with convolutional coding).

Note, however that our purpose in this section is to emphasize that, despite the fact that the approximation used in the previous works is often quite accurate, exact computation may lead to improved decoding in some situations. We must clearly express that, on many sequences, both computations provided almost the same performance, but that on some of them the phenomenon shown on the Figure 2 was observed. This clearly illustrates the usefulness of using an exact computation rather than an approximate one.

V. CONCLUSIONS

This paper, has presented a soft decoding technique based on a MAP estimation exploiting the *a priori* knowledge about the encoding process, the source statistics, and the channel noise. The proposed decoder is able to handle adaptive probabilities and context modeling, and is compatible with standard implementation of CABAC without changing its compression efficiency. Current work is dedicated to embedding the soft MAP decoder within the H.264 decoder.

VI. 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] K. Sayood, H. H. Otu, and N. Demir, "Joint source-channel coding for variable length codes," *IEEE Trans. on Comm.*, vol. 48(5), pp. 787–794, 2000.

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

[4] 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.

[5] 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.

[6] 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.

[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] M. Grangetto, P. Cosman, and G. Olmo, "Joint source/channel coding and map decoding of arithmetic codes," *IEEE Trans. on Comm.*, pp. 1007–1015, 2005.

[9] 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.

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

[11] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Comm. of the ACM*, vol. 30, pp. 520–540, 1987.

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

[13] M. Jeanne, I. Siaud, O. Seller, and P. Siohan, "Application of a joint source-channel decoding technique to umts channel codes and ofdm modulation," *Proc. of ICT, Fortaleza, Brazil*, pp. 912–923, 2004.

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