A Mixed-Framework Arithmetic Coder

Paolo Prandoni *Martin Vetterli*
LCAV, Ecole Polytechnique Fédérale de Lausanne, Switzerland
email: prandoni@de.epfl.ch, vetterli@de.epfl.ch

ABSTRACT

A mixed-framework adaptive arithmetic coding technique is introduced in which the probability estimation process dynamically switches among a set of allowed models; the sequence of model choices is determined jointly with the relative data segmentation to yield the optimal coding performance for the given set of predictors. A specific application is detailed in which the arithmetic coder can switch between the backward adaptive estimator of the Q-Coder and a forward adaptive bit count scheme.

1. INTRODUCTION

Several picture coding systems like JBIG, JPEG, and EZW, utilize or allow for a binary arithmetic coder as the last processing step. Indeed, given an accurate estimate $P(x)$ of the probability distribution of the binary input data x, arithmetic coding produces a code stream of expected length L within two bits of the source entropy, $L < H(X) + 2$. The practical difficulty in implementing this scheme lies entirely with the estimation of $P(x)$; given the nonstationary nature of image data, dynamically adaptive schemes have proven the most successful.

Adaptivity in the probability estimation process can be of two kinds. *Backward adaptivity* infers from past data a statistical description which is extended to the current input. The Q-Coder [1], for instance, implements a backward predictive model by means of a state machine whose transition table is carefully tailored to the average statistical behavior of bilevel images. The very structure of the adaptation mechanism, however, imposes a constraint on the generality of the predicted distribution; furthermore, causal transition in the data (as in a piecewise stationary memoryless source for which the underlying probability distribution changes abruptly in time) will cause a substantial mismatch overhead in any backward adaptive coder. *Forward adaptivity*, on the other hand, relies on both past and future data to generate an estimate of the underlying statistical model; two-pass, adaptive Huffman coding in JPEG is an example. Clearly, this model offers complete generality in that an exhaustive minimization process can be performed to single out the distribution which minimizes the length of the coder's output. Apart from the higher computational cost of such a procedure, however, the price to be paid for this generality with respect to backward adaptive systems consists in the *side information* which needs to be tagged to the coded data to inform the decoder of the selected statistical model.

The choice of the most advantageous estimator depends entirely on the particular data sequence which needs to be encoded: it is easy to produce data sets for which a backward adaptive model outperforms a forward adaptive model, and viceversa. Therefore, the criterion for choosing the type of estimator can only be the compression ratio at the encoder's output. Moving one step further, it is easy to see that for a nonstationary data segment a suitable time segmentation of the data, together with the "right" coding model choice for each segment, could lead to an even better performance. It will be shown that in fact, under reasonably mild assumptions, a *joint* time segmentation and model choice process provides the *optimal* compression ratio for a given set of backward and forward estimators.

In this paper we will develop an arithmetic coder in which the estimation procedure can be either the backward-adaptive Q-Coder state machine or a forward-adaptive bit count estimation; the framework is however fully general and any set estimators can be successfully used. Model switches will be determined based on the particular properties of the input sequence. Intuitively, the model follows a simple strategy: where the side information associated with a forward adaptive prediction outweighs the coding improvement, the system will select the possibly less efficient but cheaper backward model; conversely, when the backward model fails to produce a good estimate, the forward model will supersede it. Importantly, this series of model choices will be carried out in a *globally optimal* sense together with the segmentation process.

2. THEORETICAL BACKGROUND

For a wide class of *stationary* signals including Markov processes and, as a particular case thereof, Bernoulli sequences, Minimum Description Length (MDL) theory [2] provides a lower bound on coding performance. Specifically, given a *finite* N-point data sequence $x_1^N = x_1 x_2 \dots x_N$, the performance of any coder, backward or forward adaptive, is bounded by the *information* of the sequence, defined as

$$I(x) = \min_{k,\underline{\theta}} \{ -\log P_{\underline{\theta}}(x) + (1/2) k \log N \},$$

where the statistical model $P_{\underline{\theta}}(x)$ depends on the k-element parameter vector $\underline{\theta} = \{\theta_1 \theta_2 \dots \theta_k\}$ and the minimization is carried out over all number of parameters and all corresponding parameter vectors.

Considering the case of Bernoulli sources for simplicity, by taking expectations over all N-point data sequences we obtain the average information

$$I(X) = H(\theta) + \frac{1}{2} \log N, \qquad (1)$$

where θ is the Bernoulli parameter for the source and $H(\theta)$ is the binary entropy function. The last term represents the cost of not knowing θ a priori, and it can be paid for in two ways: in forward adaptive coders, by providing the estimated value for θ to the decoder; in backward adaptive coders, by the estimation mismatches in the causal prediction.

For nonstationary sequences the situation is more complex; the previous bounds in a MDL sense have been extended to *piecewise stationary* Bernoulli sources by Merhav [3]. Intuitively, in this case the extra cost to be paid also includes the information pertaining to the parameter transitions in the data sequence; it can indeed be shown that for N-point sequences containing M switchpoints, the average information can be expressed as:

$$I(X) = \frac{1}{M} \sum_{i=0}^{M} l_i H(\theta_i) + M \frac{1}{2} \log N + M log N, \quad (2)$$

where l_i is the length of the data segment for which the Bernoulli parameter stays constant and equal to θ_i, and $\sum l_i = N$. With respect to (1), the cost associated to the Bernoulli parameter is multiplied by M; furthermore, an additional price of $\log N$ bits for each transition appears. Merhav also proved that this lower bound can be attained by a purely sequential coder (no look-ahead), but the method is hardly practical. These results were recently extended by Willems [4], who addressed more practical issues in the problem of determining the set of parameters for a piecewise-constant Bernoulli sequence. His analysis is however mostly concerned with the attainability of the Merhav bound in a strictly sequential fashion.

While these lower bounds are very useful in stating the best performance one can expect from a coding system, their asymptotic nature makes them less helpful in the case of a single realization of a nonstationary process. In particular, the theoretically equivalent efficiency of most estimators holds only in the limit; for one finite data sequence, practical and computationally efficient estimators can yield substantially different results.

In this work, we somewhat step aside from the theoretical framework described above; rather, our goal is to implement a practical system using widely available building blocks (such as the Q-coder), the baseline performance of which is very well known in a variety of settings. The aim is twofold; on one hand we want to provide a system which easily "tags on" to pre-existing coding scheme. One might want to retain the core structure of such coders for efficiency or compatibility reasons. Secondly, we want this system to be as open as possible, the immediate advantage being its straightforward extendibility to settings (such as the multi-symbol nonstationary source) for which few theoretical guidelines are available.

3. OPTIMAL SEGMENTATION

Given the nonstationary nature of the data sequence, intuitively one might think to first segment the data according to some stationarity criterion and then code each segment separately with the best coder for each segment. For any stationarity measure, however, it would be possible to construct a different segmentation and sequence of models which yields a lower number of coded bits. Indeed, for a single realization of a time-varying process, there is no absolute measure of stationarity; the criterion for determining the time segmentation must be the same as for the choice of the coding model, namely the compression ratio. By *jointly* determining the segmentation and the choice of models, the *globally optimal* result is achieved for the given set of coding models.

A good review of optimal segmentation techniques for data analysis can be found in [5]; the main ideas will be used here to formalize the problem at hand. In the reminder, we will consider memoryless binary sequences $x_1^N = x_1 x_2 \dots x_N$ generated by a switching Bernoulli source whose parameter θ changes abruptly over time (piecewise constant distribution); we will make no assumptions on the number or on the location of the transition points. For convenience, define $x_T(m,n) = x_{(m-1)T+1}^{nT}$, which represents a subset (segment) of the data sequence. For $T = 1$, it is simply $x_1(m,n) = x_m^n$, and $x_1(m,m) = x_m$. By setting $T > 1$, each segment is composed by an integer number of minimal T-point data cells; this will be useful in the next section, and this no-

tation will allow us to describe the minimization algorithm independently of T.

The goal is to find the optimal minimum total coding cost:

$$\hat{C}(N) = \min_{s \in S} \min_{p \in P(s)} \{c(x_T(1,N))\}, \qquad (3)$$

where S is the set of all possible data segmentations, $P(s)$ is the set of all possible model choices for segmentation s, and $c(\cdot)$ is the number of output bits when the segments described by s are sequentially coded with the models described in p.

Let us assume we have K different coding models for the data; $c_k(x_T(m,n))$ will represent the number of output bits when coding the data segment $x_T(m,n)$ with coder number k. To make the minimization problem in (3) computationally tractable, we assume that the coding cost for all models is *nonnegative*, *additive* over disjoint segments and that the coding cost for a segment is *independent* of the models used to code the previous data. With these assumption we define the minimum coding cost for a given segment as

$$C(x_T(m,n)) = \min_{1 \le k \le K} \{c_k(x_T(m,n))\}. \qquad (4)$$

For $i \le q$, consider a set of $i+1$ indices $s = \{l_1, \ldots, l_{i+1}\}$ such that $0 = l_1 < l_2 < \ldots < l_{i+1} = q$. This set uniquely defines a segmentation of the data subset $x_T(1,q)$ in which segment j, for $1 \le j \le i$ is $x_T(l_j + 1, l_{j+1})$; call this an order-q sub-segmentation and, for all s, let $I(s) = i$, the number of segments. With this notation, and exploiting the above assumptions on the coding models, we can simplify (3) for any q as:

$$\begin{aligned} \hat{C}(q) &= \min_{s \in S(q)} \min_{p \in P(s)} \{c(x_T(1,q))\} \\ &= \min_{s \in S(q)} \{\sum_{j=1}^{I(s)} C(x_T(l_j+1, l_{j+1}))\} \qquad (5) \end{aligned}$$

We now describe an efficient way to explore the space of all possible segmentations. Let $S(q)$ be the set of all the $2^{(q-1)}$ possible order-q sub-segmentations; the sets $S(q)$ can be built incrementally in the following way:

Step 0: $S(0) = \{\,\{0\}\,\}$ by definition;
Step q:

$$S(q) = \{s \cup \{p\}, \forall s \in S_j, j = 0, \ldots, q-1\}$$

where, if $s = \{l_1, \ldots, l_i\}$, $s \cup \{q\} = \{l_1, \ldots, l_i, q\}$. As an example, the first three steps yield:

$$\begin{aligned} S_1 &= \{\{0,1\}\}; \\ S_2 &= \{\{0,2\},\{0,1,2\}\}; \\ S_3 &= \{\{0,3\},\{0,1,3\},\{0,2,3\},\{0,1,2,3\}\}; \end{aligned}$$

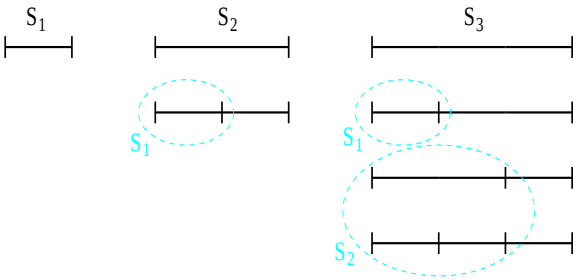

Figure 1. Incremental construction of the segmentation.

and are illustrated graphically in figure 1. Given this structured construction of $S(q)$, we can rewrite (5) for all q as:

$$\hat{C}(q) = \min_{0 \le h < q} \min_{s \in S(h)} \qquad (6)$$
$$\{\sum_{j=1}^{I(s)} C(x_T(l_j+1, l_{j+1})) + C(x_T(h+1, q))\}$$

In other words, using Bellmann's optimality principle [6], the minimization process can be deployed incrementally together with the set of possible segmentations:

Step 0: $\hat{C}(0) = 0$;
Step q:

$$\hat{C}(q) = \min_{0 \le j < q} \{\hat{C}(j) + C(x_T(j+1, q))\} \qquad (7)$$

which is carried out up to $q = N$; at each step we need to keep track of $\hat{C}(q)$ and of the value for j which minimizes it. This dynamic programming approach allows us to find the optimal cost for the complete sequence (and the corresponding optimal segmentation) in an incremental, efficient way, which has only quadratic rather than exponential complexity in the number of operations and linear storage requirements.

4. THE ALGORITHM

4.1. Coding models

We apply (3) to the arithmetic coding problem by allowing for two coding models ($K = 2$).

The first model is the backward adaptive predictor implemented in the Q-Coder [1]. This estimator declares one of the two possible input values (say 0) the *least probable symbol* (LPS) and assigns a probability to it; the other input value (1) is labeled the *most probable symbol* (MPS). The LPS probability is constantly adjusted depending on the sequence of LPS's and MPS's which are encoded; if the LPS probability passes the 0.5 mark, the LPS and MPS labels are swapped. The probability estimation is carried out by means of a 29-value state machine; an LPS probability value

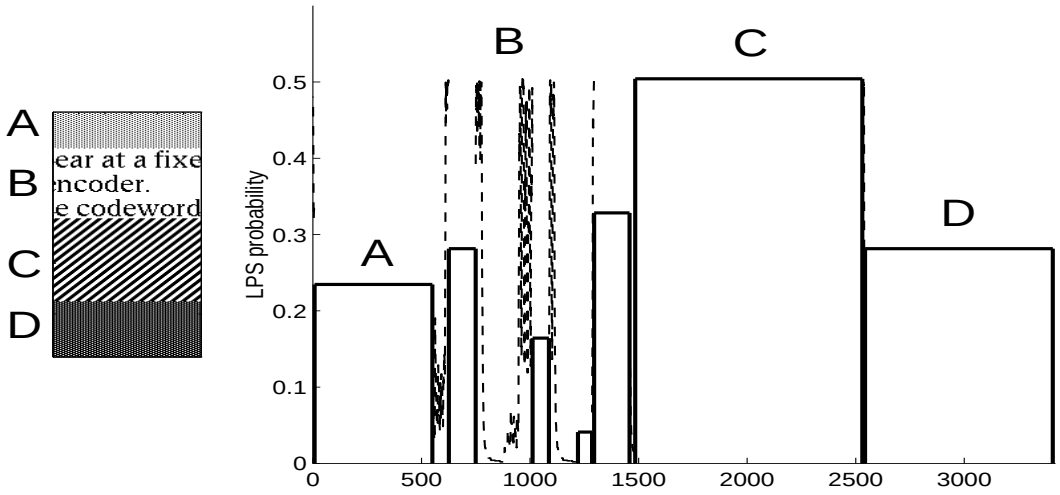

Figure 2. Coding example

and pointers to the next and previous states are associated to each state. The entries in the state machine were found experimentally in the context of bilevel image coding.

The second model is a forward adaptive predictor in which an estimate of the Bernoulli parameter for segment $x_T(m,n)$ is simply $\hat{\theta} = (1/T)H(x_T(m,n))/(n-m+1)$, where $H(\cdot)$ indicates the Hamming weight; this estimate is subsequently quantized to a five-bit index into the Q-Coder probability table.

Strictly speaking, the additivity and independence properties required by the dynamic programming approach of the previous section are not fulfilled by these coding models. However, by constructing a segmentation where the minimal element is a multi-bit cell, the requirements are approximately satisfied at the price of a slight suboptimality for the final segmentation. In our algorithm we set the cell to be one byte long ($T = 8$).

The optimal cost of equation (4) becomes in this case:

$$C(x_T(m,n)) = \min\{c_b(x_T(m,n)), c_f(x_T(m,n))+D\}$$

where c_b and c_f are the output bits for the backward and forward models respectively and D is the number of bits needed to code the side information for the forward model.

4.2. Encoding side information

After the algorithm has found the optimal segmentation and sequence of coding models, the data must be encoded accordingly, including the necessary side information for the decoder about model switches. In order to allow for sequential decoding, with no extra buffering space requirements, we chose to embed switchpoint information in the coded stream as the switches occur, in the form of escape characters. There are several ways to encode an escape by exploiting the particular features of a given implementation of an arithmetic coder (such as forbidden code register values in the Q-Coder [7]); we will however describe a totally general way which fits better in the framework introduced so far.

In a binary arithmetic coder, the output bitstream is a binary number uniquely identifying a final coding interval of length A. Consider a stationary input bitstream for simplicity (the following results are the same in the case of adaptively adjusted probability values): let the (estimated) probabilities for the least probable symbol and most probable symbol be p and $1-p$, respectively; the algorithm starts by setting $A = 1$ and, for each input bit, appropriately scales A by p or $1-p$. At the end of the N-bit input stream, $A = p^{L(N)}(1-p)^{(N-L(N))}$, where $L(N)$ is the number of LPS which appeared in the sequence; the compression rate can be therefore expressed as

$$\alpha_0 = 1 + \frac{\log A}{N},$$

where here and in the following all logarithms are in base 2. In our implementation we consider the escape character as a fictitious third symbol of nominal probability $1-\epsilon$; we then scale the LPS and MPS probabilities to $p\epsilon$ and $(1-p)\epsilon$ respectively, while leaving the estimation procedure unmodified. Each escape character contributes approximately $-\log(1-\epsilon)$ bits to the output stream; on the other hand, independently of the number of escapes, the compression rate is reduced to:

$$\begin{aligned} \alpha(\epsilon) &= 1 + \frac{L(N)\log p\epsilon + (N-L(N))\log(1-p)\epsilon}{N} \\ &= \alpha_0 + \log\epsilon \end{aligned} \quad (8)$$

The cost of a single escape code and the output expansion due to the possibility of a third symbol are therefore related by equation (8), which

in turn depends directly on d, the number of bits d we want to spend for each escape, since $\epsilon = 1 - 2^{-d}$. Ideally, the value for d should be found as a part of the same global optimization process by iterating the segmentation algorithm for all meaningful values for d until the global minimum for the overall compression ratio is found. In practice, experience has shown that values for d in the range of 10-15 bits provide generally good results. In the examples in the following sections, $d = 13$ throughout; the total side information cost for a forward encoded segment is therefore $D = 2d + 6$ bits, where the term $2d$ accounts for the escape codes at the beginning and at the end of the segment and the remaining 6 bits are the index into the Q-Coder probability table (including one bit to specify the LPS symbol).

4.3. Complexity

As stated in Section 3., the computational complexity of the algorithm is $O(N^2)$. As for all dynamic programming methods, however, this complexity can be reduced to $O(N)$ at the price of a slight suboptimality; as the complexity coefficient (which exactly relates the number of operations to N) grows, the suboptimality becomes more and more negligible. The idea is that past data is not likely to influence the global segmentation after a significant delay; after such a delay we can therefore assume that the early part of the segmentation coincides with the optimal one and code the relative data with the models selected by the algorithm; the early data can be subsequently discarded. With respect to equation (7), this is equivalent to saying that, for large q, the value for j minimizing the expression will not be small; if we fix the value for the "significant delay" to wT samples, the iteration becomes:

$$\hat{C}(q) = \min_{q-w \le j < q} \{\hat{C}(j) + C(x_T(j+1, q))\}. \quad (9)$$

The tradeoff between suboptimality and delay span is obviously dependent on the properties of the class of input signals.

It is to be noted that the complexity of the decoder is equivalent to that of a standard arithmetic decoder, and that the decoding is instantaneous. This complexity asymmetry, together with the encoder's inherent delay, suggest that the proposed algorithm is most useful in lossless data storage settings in which the premium is almost exclusively on compression ratios.

5. A SIMPLE EXAMPLE

In the following example, the input to the dynamic arithmetic coder is the raster scan of the bilevel image displayed on the left of Figure 2. While this is not, of course, a proper image coding system (no image-specific prediction is applied) it illustrates in an intuitive way the properties of the proposed algorithm. The plot on the right of Figure 2 displays the LPS probability for the binary arithmetic coder versus the index of the coded byte. The rectangular blocks in solid line represent portions of the signal for which the coder has identified a constant parameter for the LPS probability by a forward adaptive estimate; the dashed line represents the evolution of the LPS probability when the system works in a backward adaptive mode. We can identify four distinct segments, corresponding to the four textures in the image. Section A and D simply code the gray textures according to their pixel densities; forward adaptation is efficient here due to the regularity of the pattern. In section C, for which the image has the same number of black and white pixels per square region, the system locks to a LPS probability of $1/2$, to avoid the coding inefficiency of an oscillating backward adaptation. Finally, section B is coded mainly in backward adaptive mode, which is consistent with the frequent and short transitions in the image. It is to be noted how the algorithm achieves a very good segmentation of the image. The compression rates are 14.3% for the proposed algorithm versus 12.3% for the Q-Coder.

6. CODING OF BILEVEL IMAGES

The JBIG algorithm is a very efficient coding algorithm for bilevel images [8]. Its context-adaptive arithmetic coding is particularly effective in exploiting patterns and redundancies in a typical image, yielding very high compression rates. In JBIG, a context is a binary number identifying the configuration of bits surrounding the pixel to be encoded; for a typical image containing text and graphics, the most frequent context is context zero, accounting for the white background surrounding characters.

Transparent application of the dynamic arithmetic coder to the JBIG algorithm is difficult due to the fact that the system subdivides the image into numerous narrow stripes; this generally produces input files which are too small for the dynamic algorithm. An exception is however the data relative to context zero. Within the limited improvement margins allowed by the high efficiency of the JBIG algorithm, the mixed-framework arithmetic coder provided in this case some interesting results. Figure 3-(a) displays the improvement (in bytes) obtained by coding context-0 data in each JBIG stripe with the proposed algorithm; the input was the standard CCITT fax test image number five. The total output reduction for the image is 34 bytes, which is not negligible considering that the baseline compression ratio for the standard JBIG is already 91.5%, with an output file of 6528 bytes. Figure 3-(b) displays the segmentation and the LPS probabilty for the data in stripe 17 as an example.

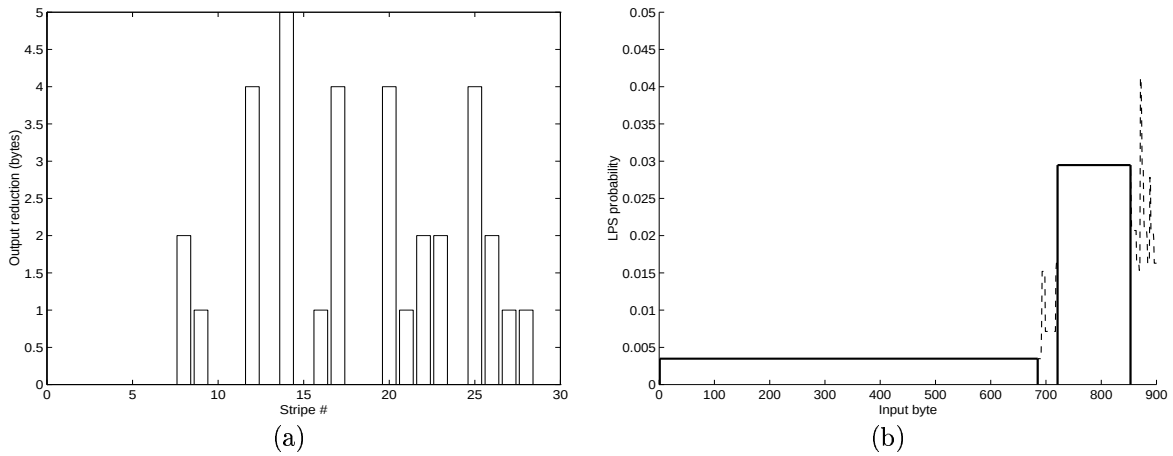

Figure 3. (a) Byte savings per stripe for the mixed-framework coder applied to the JBIG algorithm. (b) Segmentation for stripe 17

Future developments of this line of research are aimed at a closer integration of the proposed algorithm with an existing image coding system.

7. CONCLUSIONS

We have presented an arithmetic coder in which the probability estimation procedure switches between forward and backward adaptivity in conjunction with the optimal segmentation of the nonstationary input data. While the algorithm has been exemplified using the probability estimator deployed in the Q-Coder, the proposed technique is completely general and can be directly applied to any set of coding models; a method to encode side information is also proposed which is transparent to the implementational details of the arithmetic coder. The proposed algorithm has been tested within the framework of the JBIG image coding system with positive results.

ACKNOWLEGDGEMENTS

The first author would like to thank Claudio Weidmann for his insightful comments and for his help in presenting the material.

The second author would like to thank Kannan Ramchandran of University of Illinois for suggesting to consider the problem of "kicking" an arithmetic coder back in the early '90s.

REFERENCES

[1] W. B. Pennebaker, J. L. Mitchell, G.G. Langdon, and R.B. Arps. An overview of the basic principles of the Q-Coder adaptive binary arithmetic coder. *IBM J. Res. Develop.*, 32(6):717–726, November 1988.

[2] J. Rissanen. Universal coding, information, prediction, and estimation. *IEEE Tran. on IT*, 30(4):629–636, July 1984.

[3] N. Merhav. On the minimum description length principle for sources with piecewise constant parameters. *IEEE Trans. on IT*, 39(6):1962–1967, November 1993.

[4] F. M. J. Willems. Coding for a binary independent piecewise-identically-distributed source. *IEEE Trans. on IT*, 42(6):2210–2217, November 1996.

[5] Z. Xiong, K. Ramchandran, C. Herley, and M. T. Orchard. Flexible tree-structured signal expansions using time-varying wavelet packets. *IEEE Trans. SP*, 45(2):333–345, Feb 1997.

[6] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[7] R. B. Arps, J. M. Cheng, and G. C. Langdon. Control character insertion into arithmetically encoded strings. *IBM Tech. Disclosure Bulletin*, 25:2051, 1982.

[8] International Telecommunication Union. *ITU-T Reccommendation T.82*. ITU, Geneva, Switzerland, March 1993.