

# Optimal Trellis-Based Buffered Compression and Fast Approximations

Antonio Ortega, *Student Member, IEEE*, Kannan Ramchandran, *Student Member, IEEE*,  
and Martin Vetterli, *Senior Member, IEEE*

**Abstract**— We formalize the description of the buffer-constrained adaptive quantization problem. For a given set of admissible quantizers used to code a discrete nonstationary signal sequence in a buffer-constrained environment, we formulate the optimal solution. We also develop slightly suboptimal but much faster approximations. These solutions are valid for any globally minimum distortion criterion, which is additive over the individual elements of the sequence. As a first step, we define the problem as one of constrained, discrete optimization and establish its equivalence to some of the problems studied in the field of integer programming. Forward dynamic programming using the Viterbi algorithm is shown to provide a way of computing the optimal solution. Then, we provide a heuristic algorithm based on Lagrangian optimization using an operational rate-distortion framework that, with computing complexity reduced by an order of magnitude, approaches the optimally achievable performance. Our algorithms can serve as a benchmark for assessing the performance of buffer control strategies and are useful for applications such as multimedia workstation displays, video encoding for CD-ROM's, and buffered JPEG coding environments, where processing delay is not a concern but decoding buffer size has to be minimized.

## I. INTRODUCTION

**I**N RECENT years, spurred by an increased interest in efficient digital compression techniques, variable bit rate (VBR) methods for source coding have received renewed attention. In order to exploit the redundancy present in typical nonstationary signals, one has to resort to VBR coding to approach the compression limits set by the source entropy. On the other hand, when the coded bit stream is transmitted over constant bit rate (CBR) channels, buffering is needed to absorb the variations of the instantaneous source bit rate. Even transmission over asynchronous transfer mode (ATM) networks for synchronous applications makes it necessary to buffer the source as excessive delay would produce loss of information.

Studies of buffered variable-length coding of stationary memoryless sources [1] have shown that the probability of eventual buffer overflow is 1 for any buffer size, even when the average rate of the source is matched to the channel rate.

Manuscript received April 24, 1992; revised March 16, 1993. The associate editor coordinating the review of this paper and approving it for publication was Dr. Hsueh-Ming Hang. This work was supported by the Fulbright Commission and the Ministry of Education of Spain, the New York State Science and Technology Foundation's CAT at Columbia University, and the National Science Foundation under grants ECD-88-11111 and MIP-90-14189.

The authors are with the Department of Electrical Engineering and Center for Telecommunications Research, Columbia University, New York, NY 10027.

IEEE Log Number 9213624.

Hence, a buffer control strategy is necessary *even for stationary sources*. In the case of nonstationary sources (like typical video sequences), defining efficient buffer control policies for the use of VBR coding techniques in delay-limited situations is all the more relevant. Many applications like CD-ROM storage of images and video sequences, windows applications for multimedia workstations, buffer-limited JPEG [2] coding, video broadcast, and MPEG [3] buffer control strategies are *either nonreal time or real time where processing delay is not an issue*. For these finite-buffer constrained coding applications, computationally expensive methods are admissible if the complexity-performance tradeoff is worthwhile.

It is possible to resort to a large enough buffer size at the encoder and decoder in order to absorb source bit rate variations and thus reduce the significance of the buffer control algorithms. However, this approach may not only be unacceptable because of end-to-end delay restrictions but also economically unwise even when delay is not an issue, as there may exist "smarter" shorter-buffer solutions that yield the same performance. In particular, the buffer size at the decoder has often to be minimized since it represents a significant cost (e.g., broadcast applications).

This provides the motivation to investigate optimal buffer control strategies for signal sequences in a finite buffered environment. Buffer control algorithms have been considered in the literature in different contexts, especially for video coding. In early studies of video sequence coding [4] as well as in some recent ones [5], the approach was to deterministically map each buffer occupancy level to a fixed coder mode of operation. Other works [6], [7] propose the use of models of the coder behavior to set up coding rate predictions. The control algorithm sets objectives for buffer occupancy and chooses a mode of operation based on these predictions. The buffer control problem has been studied in [8] using a probabilistic model of the coded sequence. Independently of our work, buffered quantization has been studied in [9].

Buffer control schemes typically have the sole objective of preventing overflow and do not use a global distortion measure over the entire sequence to formulate their strategy. Furthermore, they are "memoryless" in their operation in that they base their algorithm only on the current buffer occupancy level. This leads to the consideration of how much better one could do if one had access to the entire signal sequence to be coded, i.e., what would be the optimal quantization choice for each element in the sequence *if the entire sequence was known a priori*? The optimality criterion would be a minimum total

distortion measure rather than that of only minimum overflow probability as the former is more meaningful from a source coder's point of view.

Such an optimal solution enables us to quantify the performance tradeoffs involving key design parameters like buffer size, buffer occupancy "granularity," and quantization choice. It can also serve as a valuable benchmark for assessing the performance of other control algorithms and quantifying their suboptimality. The optimal solution also serves as a starting point for designing less complex algorithms. In particular, we show how a solution that is only slightly suboptimal can be obtained when *only a finite window of the entire sequence is known* at each decision instant. More generally, as will be described, the problem structure lends itself to fast heuristic approximations.

In this paper, extending our previous work in [10], we formalize the generalized problem of buffer-constrained quantization of a sequence of independent blocks and describe how (given a set of quantizers, a finite buffer, and any additive cost measure over the sequence elements) an optimal solution can be found. We show how this problem (one of discrete optimization with constraints) can be construed as a deterministic dynamic programming problem with the Viterbi algorithm used to compute the optimal minimal distortion solution. We draw parallels between this buffer-constrained quantization problem and the simpler budget-constrained unbuffered quantization problem. We then present a recursive Lagrange-multiplier-based algorithm that provides a fast nearly-optimal solution with much reduced complexity.

This paper is organized as follows. In Section II, we define the problem of buffer constrained quantization. The problem is set in an integer programming framework in Section III. Section IV presents the optimal solution as well as fast heuristic approximations. Finally, Section V shows some applications of the previously described techniques, namely, providing the optimal buffer size for a given application, benchmarking other buffer control algorithms, and quantifying statistical multiplexing gain in ATM networks.

## II. PROBLEM DEFINITION

### A. Quantization and Cost Criterion

Let us define the system under study. The source consists of a sequence of "blocks" (for example, entire video frames, 8 by 8 pixel image blocks, or individual samples) that arrive at a constant rate at the encoder. The blocks are to be quantized before being entropy coded and sent to the output buffer for transmission (see Fig. 1).

For a discrete set of admissible quantizers, the problem consists of choosing that optimal sequence of quantizers that minimizes some cost measure in a *global* sense. Two points have to be emphasized here:

- The set of quantizers is finite. This is an important factor that determines the type of optimization techniques to be used.
- Minimization of a global cost measure is necessary to obtain good coding quality. A simple algorithm that lo-

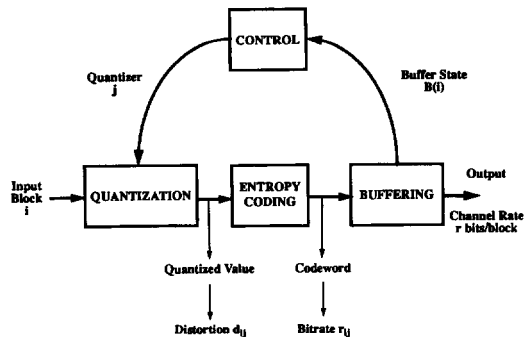


Fig. 1. Block diagram of the encoding system. The buffer control mechanism determines the quantizer to be used for each input block. Given the set of quantizers, the system can be characterized by the rate  $r_{ij}$  and distortion  $d_{ij}$  for each block and quantizer.

cally uses the best quality quantizer (in a greedy fashion) usually results in overall suboptimality.

To find the optimal sequence of quantizers, one has to first define a cost measure consistent with the system's design criteria. The best cost measure is one that reflects the subjective quality of the decoded sequence. For simplicity, we normally use MSE in our experiments, but in general, our approach can be applied whenever the cost is *additive* over all the elements of the sequence, thus making it possible to consider, for instance, MSE weighted by an activity measure or human visual system criteria (see, for example, Section V-B). Note that our additive cost criterion does not exclude the use of additional prequantization stages like a discrete cosine transform (DCT). In addition, we place no restriction on the quantization scheme to be used, thus permitting, for instance, the choice of vector quantization (VQ).

### B. Delay and Buffer Size

The system under consideration is made up of three elements: the coder/decoder at the transmitting and receiving ends, with their respective buffers, and the transmission channel. In general, although transmission need not be synchronous (for example, video transmission over ATM networks), there exists a constant delay between the following:

- i) the time at which the  $n$ th block from the sequence has been acquired by the encoder at the input ( $T_n^i$ )
- ii) the time at which the  $n$ th block from the sequence has to be available to the decoder at the output ( $T_n^o$ ).

This constant delay  $\Delta T$  (for all  $n$ ,  $T_n^o - T_n^i = \Delta T$ ) is due to the encoder and the decoder both being attached to synchronous devices. If the processing time at the decoder is negligible, there are four delay components for each block:

- i) encoder processing delay  $\delta t_p$
- ii) encoder buffer delay  $\delta t_e$
- iii) channel delay  $\delta t_c$
- iv) decoder buffer delay  $\delta t_d$ .

Clearly, for any block, the total delay through the system is

constant, i.e.

$$\delta t_p + \delta t_e + \delta t_c + \delta t_d = \Delta T. \quad (1)$$

This scheme applies to both synchronous transmission applications and ATM networks. In the former, the transmission delay ( $\delta t_c$ ) is constant, whereas in the latter, it is variable. Moreover, except for nonreal-time decoding applications, such as image retrieval, the total delay constraint is also present in ATM networks, where transmission delay plays a more important role as excessive delay due to network congestion could lead to packet loss and therefore quality degradation.

From the above, we can establish the equivalence, when the channel rate is finite, of a "finite buffer" constraint and a "finite delay" constraint. These constraints are equivalent since some of the blocks could potentially have a delay greater than the maximum permissible delay in the system ( $\Delta T$ ) if infinite buffers were used with a finite channel rate.

Consider a real-time decoding application where the delay between input and output is constant, and the channel has a constant rate (the processing time will normally be constant). An input block that encounters a near empty encoding buffer, i.e., low delay at the encoder ( $\delta t_e$  small) will necessarily, from (1), be delayed at the decoder since  $\Delta T$  is constant. Hence, the decoder buffer will tend to fill up if the encoder buffer is emptying and vice versa. From this point of view, in order to be efficient, it is clear that the *buffer memories have to be of equal sizes at both ends* as they must each be able to accommodate the maximum combined buffering delay of  $\Delta T - \delta t_c - \delta t_p$ .

In some applications, the total delay can be large, but buffer memories should remain small to reduce the cost of decoders. From the above, any buffer memory used at the encoder would have to be duplicated at each decoder. On the other hand, the additional memory used for processing ( $\delta t_p$ ) will just increase the total delay. In fact, using techniques to be described in Section IV, one could optimize the transmission of a sequence based on all the blocks in the sequence (thus using the maximum processing memory) while choosing *any physical buffer size*.

### C. Possible System Configurations

Before we formulate the problem, it is useful to list all possible system configurations in terms of the characteristics of both encoding and decoding. The key point here is whether either operation takes place in real time. The following classification is summarized in Table I.

One application is real-time encoding and decoding (e.g., a videoconferencing system) where delay, as was previously noted, is the limiting factor. Delay is especially important in applications involving interaction between the two ends (for instance a delay of more than 100 ms would be unacceptable for videoconference transmission), although it might be less crucial in other applications like live broadcast where no interaction occurs. Roughly, the maximum permissible delay can be allocated between processing and buffering, thus making the choice of buffer size essentially *delay-limited*. If the transmission has variable delay (as in ATM networks), the

TABLE I  
ALL POSSIBLE CONFIGURATIONS OF ENCODING/DECODING, THEIR IMPLICATIONS IN TERMS OF DELAY/MEMORY CONSTRAINTS, AND SOME EXAMPLES IN THE CONTEXT OF VIDEO CODING

Encoder	Decoder	Constraint	Example
Real time	Real time	Interactive: delay	Videoconferencing
Real time	Real time	Noninteractive: memory	Broadcast
Real time	Nonreal time	Memory	Live Recording
Nonreal time	Real time	Memory	Video CD-ROM
Nonreal time	Nonreal time	No constraints: "Static allocation"	File transfer

constraint on the buffer size at the encoder and the decoder will be tighter. Our algorithm will be less suitable for a situation where delay has to be small. However, if the delay constraint is not too demanding, as would be the case for broadcast, longer processing delays may be acceptable, and our techniques can be used.

If decoding has to be done in real-time but there is no time constraint for the encoding process (as in digital coding for applications such as video on CD-ROM's), delay is no longer the main issue. Taking as an example the CD-ROM device, we note that the only contribution to the delay comes from the decoding buffer since the data is read at a constant rate from the disc. Furthermore, the absolute delay between data reading and display will be noticeable only at the start of each "play" mode. Hence, the system delay can be seen as a latency time, and the maximum delay constraint will be more flexible. The limiting factor in the choice of buffer size would be the cost of the physical memory, rather than the delay; the problem is *memory limited* rather than *delay limited*.

The dual case, real-time encoding and nonreal-time decoding is essentially equivalent. To encode a sequence in real time and store it in coded format, instead of displaying it in some output device, the limiting factor is the amount of memory to be used in interfacing the coder output with the storage device. Delay is not an issue. Finally, nonreal-time problems can be essentially seen as "static" nonbuffer-constrained bit allocation problems that have been studied in the literature [11], [12].

## III. PROBLEM FORMULATION: INTEGER PROGRAMMING

### A. A First Formulation of the Problem

In the previous section, we have motivated the relevance of a finite buffer coding environment. Now that both the cost and the constraints have been defined, a first formulation of the problem can be given:

#### Formulation 1:

*Given a set of quantizers, a sequence of blocks to be quantized, and a finite buffer, select the optimal quantizer for each block so that the total cost measure is minimized and the finite buffer is never in overflow.*

Note that underflow is not included in this formulation. An underflow constraint would be redundant, given that the objective of minimizing distortion would tend to increase the coding bit rate and thus automatically penalize underflow.

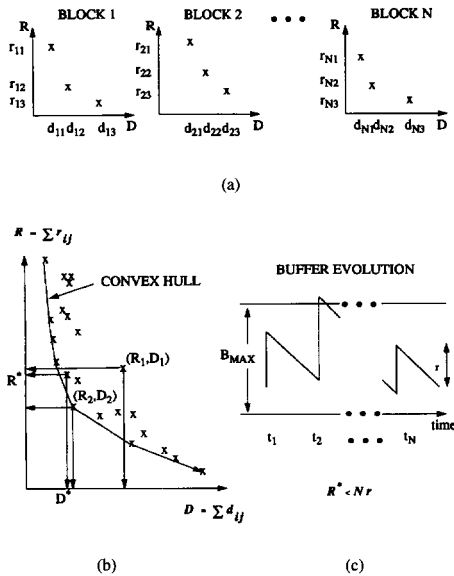


Fig. 2. Individual and composite R-D characteristics. Constrained and unconstrained optimization: (a) Each block in the sequence has a different R-D characteristic. In this example, there are three quantizers available to code each block; (b) for a given choice of quantizers for the blocks in the sequence, we can obtain R-D points to form the composite characteristic. No point in the convex hull meets the budget  $R_1$ . The optimal solution  $R^*$  is not a convex hull solution; (c)  $R^*$  is not a feasible solution with the chosen buffer size.

### B. Continuous versus Discrete Optimization

A fundamental characteristic of the above problem is that the set of available quantizers is *finite*. To see the importance of this, consider the following example, where  $N$  blocks are coded using three quantizers. Fig. 2(a) shows the operational rate-distortion (R-D) characteristics of each of these blocks, and Fig. 2(b) represents the composite rate-distortion characteristic, that is, all the values of total rate and distortion obtained using combinations of all admissible quantizer choices for each block. Refer to [13] for a detailed treatment of an operational rate-distortion framework.

The convex hull of the set of points in the composite R-D characteristic, as will be seen in Section IV-B, contains those solutions that are attainable in a “fast” way and are, if they exist, optimal for each rate. Note that not all rates are attainable, meaning that the convex hull has a discrete number of operating points. As an example, in Fig. 2(b), suppose we had to allocate a budget of  $R_1$  bits among the  $N$  blocks. The only solution that exactly meets the budget  $(R_1, D_1)$  is clearly suboptimal, whereas a convex hull solution  $(R_2, D_2)$  would have to settle for a smaller total rate ( $R_2 < R_1$ ). The optimal solution in this case is  $(R^*, D^*)$ , which is not attainable using convex hull techniques but would be reached by our optimal algorithm.

### C. Constrained versus Unconstrained Optimization

In the case where there is no buffering constraint and only a desired total bit budget has to be met, well-known optimal

bit allocation techniques have been documented [11]. We will call this situation an *unconstrained* allocation problem, i.e., one where buffers are either infinite or large enough to store the complete sequence. In general, the finite buffered case, or *constrained* allocation, will be different. As each block is coded, it is sent to an output buffer, and therefore, the number of bits available for certain blocks will be limited by the constraints imposed by the buffer, namely, underflow and overflow conditions. As an example, note that the unconstrained solution  $(R^*, D^*)$  of Fig. 2(b) may produce overflow for a certain finite buffer size (see Fig. 2(c)) and thus is not a solution to the constrained problem, even though  $R^* \leq N \cdot r$ ; thus, blocks are coded on average with a number of bits lower than the channel rate  $r$ . This example should motivate the fact that having buffer constraints increases the complexity of the optimization. However, it should be noted that buffering constraints may not always render the unconstrained solution infeasible; therefore, studying the solution to the unconstrained problem may be helpful in solving the constrained one. For instance, in Fig. 2(c), for the same channel rate  $r$ , if the buffer size was increased enough to avoid overflow, the unconstrained solution would *also* be the solution to the constrained problem.

### D. Integer Programming Formulation

The problem described in the above section is one of constrained optimization over a set of discrete operating points. The optimization has to trade off limited resources (bit rate) for quality (distortion). What makes the solution of this problem complex is the discrete nature of the space of possible solutions. This problem can be described and solved using techniques developed in the field of integer programming [14], [15]. We now present an integer programming formulation that will be helpful in developing the optimal solution, although alternative formulations can also be proposed. See Fig. 1 for the notation.

Consider the allocation for  $N$  blocks, and suppose there are  $M$  quantizers available to code each block. Let  $d_{ij}$  and  $r_{ij}$  be, respectively, the distortion and the number of bits<sup>1</sup> produced by the coding of block  $i$  with quantizer  $j$ , and let  $r$  be the channel rate per block. Define an admissible solution  $x$  as a selection of one quantizer for each block, i.e., a mapping from  $\{1, 2, \dots, N\}$  to  $\{1, 2, \dots, M\}$ ,  $x = \{x(1), x(2), \dots, x(N)\}$ , where each  $x(i)$  is the index of one of the  $M$  quantizers for block  $i$ . Therefore,  $(r_{1x(1)}, \dots, r_{Nx(N)})$  and  $(d_{1x(1)}, \dots, d_{Nx(N)})$  are, respectively, the rate and distortion for each block and a given choice of quantizers  $x$ .

Now, define the buffer occupancy at stage  $i$ ,  $B(i)$  for a given admissible solution  $x$ . To account for the fact that the buffer occupancy cannot be negative at any stage (i.e., underflow means the buffer occupancy is 0), we use a recursive definition.

Let  $B(1) = r_{1x(1)} + B(0)$ ,  $B(2) = \max(B(1) + r_{2x(2)} - r, 0)$  and, in general

$$B(i) = \max(B(i-1) + r_{ix(i)} - r, 0) \quad (2)$$

<sup>1</sup>Note that in  $r_{ij}$ , we include the bits of overhead required to specify that quantizer  $j$  was used.

where the buffer occupancy at each block instant is increased by the coding rate of the current block and decreased by the channel rate.  $B(0)$  is the initial buffer state.

*Formulation 2: (Integer Programming)*

The problem is to find the mapping  $x$  that solves

$$\min \left( \sum_{i=1}^N d_{ix(i)} \right), \quad (3)$$

subject to

$$B(i) \leq B_{\max}, \quad \forall i = 1, \dots, N \quad (4)$$

where  $B_{\max}$  is the buffer size.

This problem is similar to the one known in the integer programming literature as the generalized assignment problem (GAP)<sup>2</sup>. We will show in what follows how the problem lends itself to efficient optimal and slightly suboptimal solutions.

#### IV. OPTIMAL ALGORITHM AND FAST APPROXIMATIONS

Throughout this section, experimental results that compare the different approaches will be shown. In the experiments, we use a JPEG-like coding scheme, with the difference from JPEG [2] being that we permit variable quantizer scales for each 8 by 8 block. Our sequence of signal blocks thus consists of 8 by 8 image blocks coded using  $M$  different quantization scales in a JPEG coding environment. Note that while the exact values for attainable signal-to-noise ratio (SNR)<sup>3</sup> are source and coder dependent, the results would still hold for a general class of sources and coders. For convenience, we sometimes consider the normalized buffer size, i.e.,  $B_{\max}/r$ , which indicates the number of blocks that can "fit in the buffer" when coded at the average rate or, in other words, the buffer size expressed in average blocks.

##### A. Optimal Algorithm

1) *Dynamic Programming Solution Using the Viterbi Algorithm:* The problem under consideration can be solved using a typical deterministic dynamic programming (DP) framework, where a solution is found for every possible initial buffer state. Here, it is more natural to resort to the special case of forward DP known as the Viterbi algorithm (VA) [17], [18] for which, given the initial conditions, the best solutions leading to each of the possible final states are found. This technique, which is successfully employed in the decoding of convolutional codes [19], has also been used in the context of trellis-coded modulation (TCM) [20] and trellis-coded quantization (TCQ) [21]. We can apply the VA because our system, due to the finite buffer, can be in only a finite number of states. The basic principle consists of creating a trellis to represent all the viable allocations at each instant, given the buffer constraints. Each path in the trellis has an associated cost (the total distortion accumulated by coding the successive blocks

<sup>2</sup>In the GAP [16], the objective is to minimize a cost using some resources limited by a set of inequality constraints.

<sup>3</sup>We use peak SNR as defined by  $\text{SNR} = 10 \log_{10}(255^2/\text{MSE})$ .

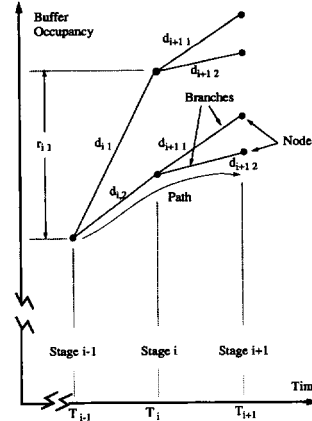


Fig. 3. Trellis diagram to be used for the Viterbi algorithm solution. Each branch corresponds to a quantizer choice for a given block and has an associated cost, whereas its length along the vertical axis is a function of the rate. For instance, quantizer 1 at stage  $i$  produces a distortion  $d_{i,1}$  and requires rate  $r_{i,1}$ . A path will correspond to a quantizer assignment to all the blocks in the sequence.

with the quantizers chosen in the path) and a certain buffer occupancy at each stage.

Let us clarify some of the terms to be used in the discussion. Referring to Fig. 3:

- **Trellis:** the trellis is made of all possible paths that link the initial stage to nodes in the final stage.
- **Stage:** Each stage corresponds to a block to be coded.
- **Node:** Each node is a pair  $(i, b)$ , where  $i \in 0, \dots, N$  is the stage number, and  $b \in 0, \dots, B_{\max}$  is the buffer occupancy state.
- **Branch:** If quantizer  $j$  at stage  $i$  has R-D characteristics  $(r_{i,j}, d_{i,j})$  then node  $(i-1, b)$  will be linked by a branch of cost weight  $d_{i,j}$  to node  $(i, \max(b + r_{i,j} - r, 0))$ , provided no overflow occurs.
- **Path:** A path is a concatenation of branches. A path from the initial node to a final stage corresponds to a feasible choice of quantizers for each successive block.

We can now use the following algorithm to generate the paths in the trellis (refer to Fig. 4):

*Algorithm 1: Algorithm for trellis growth (Viterbi algorithm [17], [18]):*

**Step 0:** Choose an initial node  $(0, B_0)$  and a final node  $(N, B_N)$ , i.e., the initial and final state of the buffer.

**Step 1:** At each stage  $i$  add permissible branches (as determined by the R-D characteristics of block  $i+1$ ) to the end nodes of all surviving paths. At each node, a branch is grown for each of the available quantizers, and the cost of that branch is added to the total accumulated cost of the path arriving to the node in stage  $i+1$ .

**Step 2:** Of all the paths arriving at a node in stage  $i+1$ , the one having the lowest cost is chosen, and the rest are "pruned". This is made possible due to the additivity and

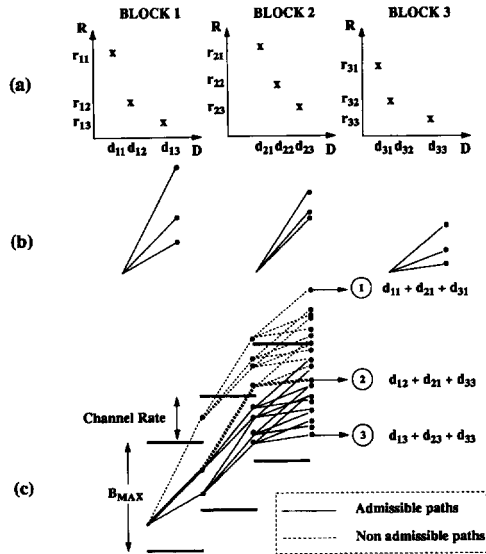


Fig. 4. Problem seen from the VA point of view: (a) R-D characteristics of the blocks with available quantizers; (b) equivalent representation. Each of the branches corresponds to the choice of a specific quantizer and has an attached cost. The length of the branch along the vertical axis is proportional to the rate; (c) all possible paths for the three blocks considered. Paths 1 and 2 cannot be used because of overflow. 1 and 3 are, respectively, the minimum and maximum distortion paths.

*independence of the costs of each branch. Note that, by our definition of branches, paths resulting in overflow are not permitted.*

**Step 3: Increment  $i$  and go to Step 1.**

In the rest of the paper, this “unrestricted” VA will be used to evaluate the suboptimality of other approaches as well as to study the characteristics of the optimal solution. Now, we explore modifications of the basic algorithm in order to reduce complexity. First, we study the suboptimality incurred when the number of representative buffer states is reduced (see Section IV-A-2), or we limit the memory, i.e., the number of blocks over which the optimization is done (see Section IV-A-3). Then, we note the similarity of our problem to an unconstrained optimization problem and present techniques that exploit it (see Section IV-B).

**2) Buffer State Clustering to Reduce Complexity in the VA:**

It is clear that using the VA to find the optimal allocation for a sequence becomes very complex as the dimensions of the trellis, both the number of states per stage (buffer size) and the number of branches (number of quantizers), increase. See Section IV-C for an analysis of the complexity of the algorithms discussed in this paper.

A way to reduce this complexity is to decrease the number of states in each trellis stage by “clustering” together some of the states. Instead of considering all possible buffer states, only the minimal cost branch in a range of states is chosen and grown in the following iteration. In our experiment, we

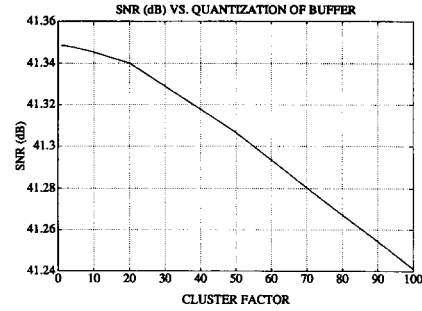


Fig. 5. SNR versus cluster factor: The cluster factor indicates how many states or nodes are clustered at each trellis stage. Note that the suboptimality is negligible for cluster factors as high as 50, i.e., when the number of nodes, and thus the complexity, is reduced by a factor of 50.

use Algorithm 1, but we cluster together a fixed number of neighboring nodes or states (the number of nodes that are clustered into one is called the “cluster factor”). The clustering operation consists in choosing the best, i.e., the minimal cost, path among those arriving at a set of neighboring nodes, and then discarding all the suboptimal paths. Thus, complexity is reduced as the number of surviving paths is limited. As an example, Fig. 5 shows how this approach, with a reduction in the number of VA states by a factor as high as 50 at each stage, produces less than 0.05 dB of degradation over the optimal distortion for our test sequence.

**3) Use of VA Techniques with Limited Memory:** In general, to guarantee global optimality of the VA for a given sequence, one needs to grow the full trellis before allocating bits to any block. Nonetheless, actual implementations require that only a finite amount of memory be used, mainly because of delay restrictions. Therefore, paths are grown and released based on just a limited number of blocks.

In Fig. 6, each plot represents surviving paths after applying the unrestricted VA to the blocks considered. Note that, in both cases, all surviving paths share the same allocation for the first few blocks, i.e., all survivors have common initial branches. This can be interpreted as a finite memory characteristic: As the number of blocks grows, the allocation for the first few blocks is not likely to be influenced by the allocation for the last ones and can thus be done independently. Since, for a sufficient number of blocks, all surviving paths have a common root, we can in effect choose the unique path for the first blocks so that the optimization can be performed in a sliding window fashion. Moreover, note that the sliding window memory  $n$  is a function of the buffer size (compare Fig. 6(a) and (b)). As the buffer size (i.e., the number of possible states) increases, so does the memory of the problem, highlighting a property similar to that of convolutional codes, which typically operate with a finite window of five times the constraint length (i.e., the memory of the finite state machine that generates the code [19]).

We now quantify the performance of a finite memory VA. In the previous section, for a length  $N$  sequence, we generated a

trellis diagram of depth  $N$ . The experiment (see Fig. 7) shows the result of using sliding windows of different sizes  $n$  for a length  $N$  sequence. In order to generate an  $n$ -block path, we have to choose a final state at stage  $i+n-1$ . This accounts for the nonmonotonicity of the SNR with the number of blocks: Given that the final state choice for the intermediate paths is arbitrary, a larger processing memory  $n$  does not always guarantee a lower global cost.

Fig. 7 shows that in the finite memory case, to get within 0.05 dB of the optimal value, one has to consider at least three or four times the normalized buffer size (i.e., the buffer size divided by the rate per block). It can be seen that the full VA cannot be implemented in real time, but the results from Fig. 7 show that even real time, memory-limited implementations of the algorithm produce only slight suboptimality. This is of importance when coding has to be done in real time but delay can be relatively large (e.g., broadcast). At each instant, the encoder can store as many blocks as needed to find the optimal path and then release the coded blocks to the buffer with a delay equal to the length of the sliding window. The buffer size at the decoder would be determined by the actual size of the encoding buffer and could thus be kept small.

### B. Fast Approximations

In Section III-C, we noted how our problem, without the buffer constraint, could be solved using well-known techniques such as Lagrange multipliers [11]. Additionally, examining the optimal solution for a sequence of length  $N$  with channel rate  $r$ , as obtained using the VA, shows that the allocation for large enough groups of  $n$  contiguous blocks ( $n \ll N$ ) is very close to  $nr$  bits.

In this section, we will illustrate how these observations can be exploited in generating a nearly optimal but much faster solution for the buffer constrained problem by solving a series of allocation problems with budget  $nr$  and no buffer constraint. To this end, we now provide a brief review of the Lagrange multiplier optimization technique.

1) *Constant Slope Optimization for Budget-Constrained Allocation*: Lagrangian optimization has been widely applied in the context of differentiable optimization in the presence of constraints [14], [15]. However, in [22], this method was also shown to be useful in the case of discrete constrained optimization. The concept was later used in [11] and [12] to determine the optimal bit allocation, in a rate distortion sense, for a given source with an arbitrary discrete set of quantizers and, more recently, to find an optimal bit allocation for a wavelet packet decomposition [23]. A complete description of this technique goes beyond the scope of this paper; therefore, we only summarize the main ideas; refer to [11], [12], [22], and [23] for further details.

Consider the problem described in Formulation 2 without the buffer constraints (i.e., if the buffer size were unlimited) but instead with a rate budget constraint so that  $R$  is the maximum total number of bits that can be used to code the sequence. We will first show how an efficient solution can be obtained for the budget-constrained problem. This solution will then be shown to be useful for our problem.

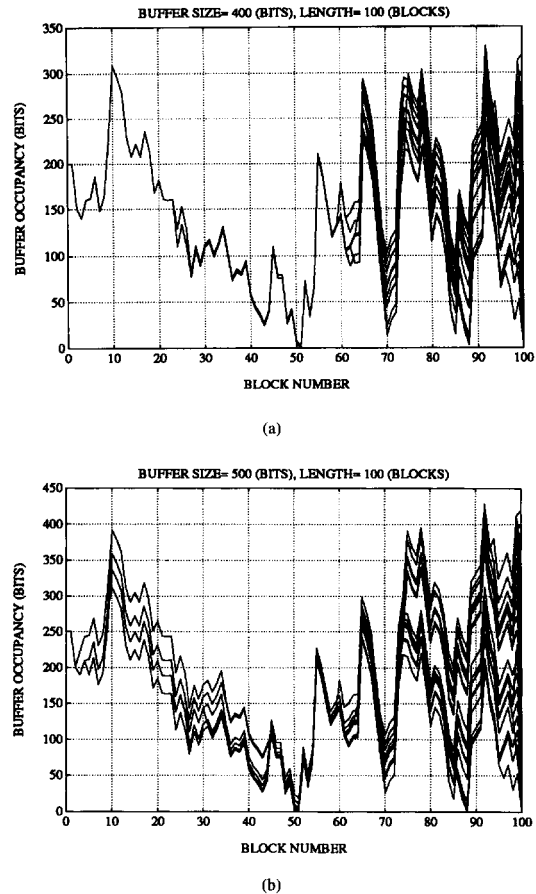


Fig. 6. Surviving paths using the full VA. The problem has finite memory. For a sufficient length, all surviving paths share the same initial path.  $B_{\max}$  is the buffer size in bits, and  $L$  is the number of blocks on which the VA is run: (a)  $B_{\max} = 400$  and  $L = 100$ ; (b)  $B_{\max} = 500$  and  $L = 100$ . Note that as the buffer size is increased the length of the common path decreases.

### Formulation 3: (Budget - Constrained Optimization)

The problem is to find the mapping  $x$  that solves

$$\min\left(\sum_{i=1}^N d_{ix(i)}\right), \quad \text{subject to:} \quad \sum_{i=1}^N r_{ix(i)} \leq R, \quad (5)$$

where  $R$  is the total budget.

Now, the main result states that, for any real positive number  $\lambda$ , the Lagrange multiplier:

*Theorem 1* [11], [22]: If the mapping  $x^*(i)$  for  $i = 1, 2, \dots, N$  minimizes

$$\sum_{i=1}^N d_{ix(i)} + \lambda \cdot r_{ix(i)}, \quad (6)$$

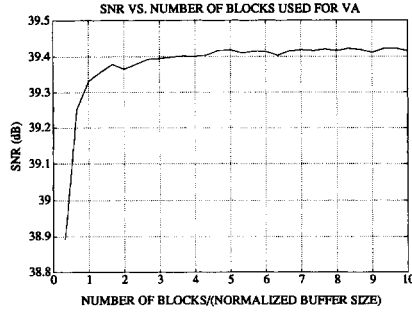


Fig. 7. SNR versus number of blocks  $n$  used to obtain the VA solution. We divide  $n$  by the normalized buffer size so that the horizontal axis represents the window of the VA expressed in “number of buffer sizes.” In the experiment, a quantizer is chosen for block  $i$  based on the best path from  $i$  to  $i + n$ . To select the path, we force it to end at an arbitrary buffer position at stage  $i + n$  (normally midbuffer), thus explaining the nonmonotonicity of the resulting function.

then it is also the optimal solution to the problem of Formulation 3 for the particular case where the total budget is

$$R = R(\lambda) = \sum_{i=1}^N r_{ix^*(i)}, \quad (7)$$

so that

$$D(\lambda) = \sum_{i=1}^N d_{ix^*(i)} \leq \sum_{i=1}^N d_{ix(i)} \quad (8)$$

for any  $x$  satisfying (5) with  $R$  given by (7).

Since we have removed the constraints, for a given operating “quality”  $\lambda$ , (6) can be rewritten as

$$\min \left( \sum_{i=1}^N d_{ix(i)} + \lambda r_{ix(i)} \right) = \sum_{i=1}^N \min(d_{ix(i)} + \lambda r_{ix(i)}) \quad (9)$$

so that the minimum can be computed independently for each block. Note also that for each block  $i$ , the point on the R-D characteristic that minimizes  $d_{ix(i)} + \lambda r_{ix(i)}$  is that point at which the line of absolute slope  $\lambda$  is tangent to the convex hull of the R-D characteristic (see Fig. 8). For this reason, we normally refer to  $\lambda$  as the slope, and since  $\lambda$  is the same for every block on the sequence, we refer to this algorithm as “constant slope optimization.”

From Theorem 1, for a fixed  $\lambda$ , we obtain the best possible solution that meets the budget constraint of (7). The budget for which we obtain a solution is itself a function of the chosen  $\lambda$ . Therefore, to find a solution to the initial problem, we need to iteratively change  $\lambda$  until we find the multiplier  $\lambda^*$  such that the total number of bits used  $R(\lambda^*) = R$  within a convex hull approximation. It has been shown [11] that this is a fast convex search. If for some  $\lambda^*$  we have  $R(\lambda^*) = R$ , then the theorem guarantees that the solution to the unconstrained problem is also the solution to the constrained one. A fast way of searching for  $\lambda^*$  using the bisection algorithm is now described. We again refer the reader to [11], [12], [22], and [23] for a more detailed description.

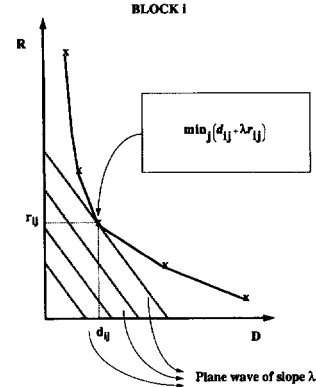


Fig. 8. For each block, minimizing  $d_{ix(i)} + \lambda r_{ix(i)}$  for a given  $\lambda$  is equivalent to finding the point in the R-D characteristic that is “hit” first by a “plane wave” of slope  $\lambda$ .

*Algorithm 2: Lagrangian optimization:*

**Step 0:** Start with two values  $\lambda_u$  and  $\lambda_l$  such that  $R(\lambda_u) \leq R \leq R(\lambda_l)$  where  $R(\cdot)$  is as defined above.

**Step 1:** Set  $\lambda_{\text{next}} = \left| \frac{D(\lambda_l) - D(\lambda_u)}{R(\lambda_l) - R(\lambda_u)} \right| + \epsilon$ , where  $\epsilon$  is an arbitrarily small positive number added to make sure that the smallest rate is picked if  $\lambda_{\text{next}}$  is a singular slope value.

**Step 2:** Repeat the optimization of (9) for  $\lambda = \lambda_{\text{next}}$ . If  $R(\lambda_{\text{next}}) = R$ , stop. Else if  $R(\lambda_{\text{next}}) \geq R$  set  $\lambda_l = \lambda_{\text{next}}$ , or else if  $(R(\lambda_{\text{next}}) \leq R)$  set  $\lambda_u = \lambda_{\text{next}}$ . Go to **Step 1**.

The main advantage of this algorithm is its efficiency. Indeed, our experiments verify that usually a relatively small number of iterations (typically less than 10) is sufficient for convergence in the problems we have considered. An analysis of the comparative complexities of the VA and a constant slope-based algorithm for the buffer-constrained quantization problem is given in Section IV-C.

As pointed out earlier, a solution for budget  $R$  may not exist for any  $\lambda$ , but using the constant slope algorithm will at least yield a solution that is optimal for a budget  $R(\lambda)$  that is slightly less than  $R$ , i.e., within a convex hull approximation of  $R$ . Fig. 9 shows all the constant slope solutions that do not produce overflow or underflow for the allocation of the first 128 blocks of a sequence. As all final buffer states are not attainable, it is clear that there are no constant slope solutions for all desired budgets. However, it must be emphasized that those “constant slope paths” that can be constructed using the above described optimization are *optimal* in the sense that there is no better way of coding the sequence if the initial and final stages had to be those determined by the constant slope path.

2) *Iterative Constant Slope Optimization for Buffer-Constrained Allocation:* We now apply the above ideas to set up the following algorithm for the buffer-constrained allocation problem:

*Algorithm 3: Iterative constant slopes:*

**Step 1:** At every stage  $k$ , use Lagrangian optimization, with budget constraint  $n \cdot r - B(k) + B_{\text{max}}/2$ , to obtain the



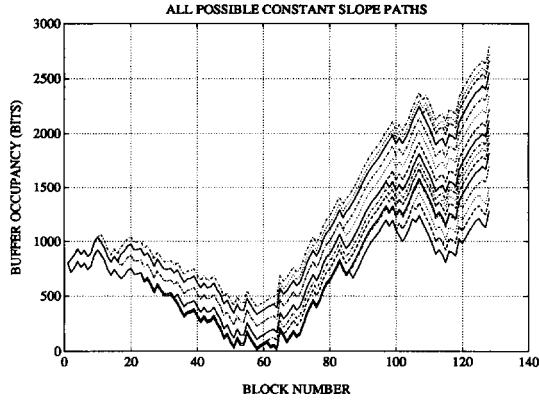


Fig. 9. All possible "constant slope" solutions. Note that not all final states can be reached with a constant slope path. However, if a path exists, it is optimal. In this example,  $B_{\max} = 3000$  bits.

best non-buffer-constrained allocation for the following  $n$  blocks.

**Step 2:** Using the quantizer indicated by the previous step for block  $k$ , release that block to the buffer and repeat **Step 1** for stage  $k + 1$ .

The budget constraint requires that the number of bits used for the next  $n$  blocks is such that at the  $n$ th stage the buffer will be in mid-position ( $B_{\max}/2$ ). This is equivalent to performing a moving window optimization so that the bit rate of the  $k$ th block depends only on the R-D characteristics of the following  $n$  blocks and on the buffer state at the  $k$ th stage. Thus, we exploit both the finite memory of the problem (see results in Fig. 7) and the effect of the buffer in terms of getting the average bit rate of a finite number of blocks close to the channel bit rate.

Results show that this approach yields a solution very close to the optimal one as obtained using the VA. When comparing the allocations obtained using Algorithms 1 and 3 for a typical sequence, experiments showed that a disparity in quantizer choice between the two approaches occurred for less than 5% of the signal blocks. Indeed, it can be seen that the optimal solution obtained with the VA follows a constant slope path for several blocks; the optimal solution is a *piecewise constant slope path*. Intuitively, as the optimal solution for a given budget without buffering is a constant slope solution, when buffering constraints are added, the solution becomes "piecewise constant slopes."

3) *Fast Iterative Constant Slope Optimization Based on Heuristics:* A straightforward way of improving the speed of the algorithm while sacrificing little quality is the following: Use Algorithm 3, except perform optimization over  $n$  blocks only when the buffer size is above or below certain bounds. Call the heuristic threshold the fraction of the buffer size used as a bound (see Fig. 10). Thus a "10% threshold" means that the path is recomputed whenever the buffer occupancy

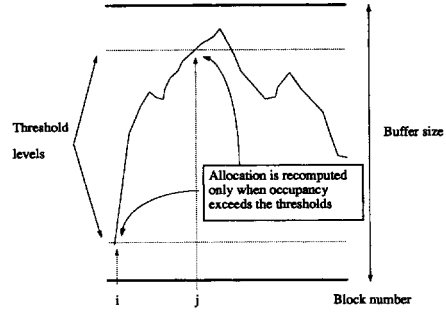


Fig. 10. Example of Algorithm 4. From block  $i$  to block  $j$ , the allocation is not recomputed. Then, as the buffer state exceeds the threshold, the allocation is recomputed.

is below 10% or above 90% of the total buffer size. With this notation, a 50% threshold corresponds to using Algorithm 3. Note that although we use symmetric thresholds for simplicity, nonsymmetric thresholds for overflow and underflow are also possible. The algorithm can be expressed in a more formal way as follows:

*Algorithm 4: Heuristic constant slopes:*

**Step 0:** Initialize the stage count  $i$  to 1.

**Step 1:** Use Algorithm 3 to compute the allocation for the following  $n$  blocks ( $i, \dots, i+n$ ). Reset the window counter  $w$  to 0.

**Step 2:** At stage  $i$ , use the allocation previously determined for block  $i$ . Increment  $i$  by 1. If the buffer occupancy is within the desired thresholds and  $w < n$ , increment  $w$  and go to **Step 2**. Else, go to **Step 1**.

Results show that for typical sequences, for a heuristic threshold as low as 10%, for which the paths are seldom recomputed, more than 85% of the quantizer choices match those selected in the optimal solution. In Fig. 11, the SNR of both the VA solution with limited memory and the heuristic (10%) approximation are compared. For a sufficiently large number of blocks, the heuristic approximation comes within 0.05 dB of the optimal value for our experiments.

### C. Complexity

In this section, the following notation will be used:  $M$  is the number of quantizers available,  $N$  is the number of blocks to be coded, and  $B_{\max}$  is the buffer size or, more precisely, the number of different states or nodes considered in the buffer.

1) *Optimization Using the VA:* Since there are  $B_{\max}$  nodes per stage, a total of  $B_{\max} \cdot N$  nodes have to be considered. Growing a branch to each individual node  $b$  involves choosing, among at most  $M$  branches arriving to that node, the branch with minimum total cost so that there are  $M$  comparisons per node. Denoting by  $C_i(b)$  the cost accumulated up to stage  $i$  on the best path arriving at node  $b$ , we have that  $C_i(b) = \min_{j=1, \dots, M} (C_{i-1}(b - r_{ij} + r) + d_{ij})$ , where the minimum is over all  $M$  quantizers. The cost of each incoming branch is computed as the sum of the cost accumulated in the

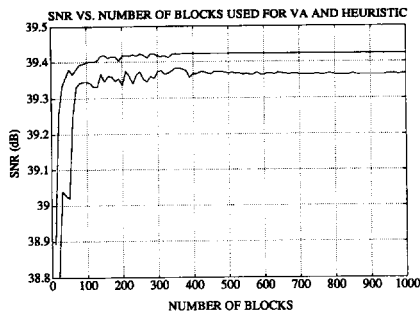


Fig. 11. SNR of suboptimal VA (top) and Algorithm 4 with 10% heuristic threshold. When a sufficient number of blocks is considered, the heuristic approximation comes within 0.05 dB of the distortion for the optimal solution.

path up to the previous stage and the cost of the branch itself. Hence, for each node, the complexity increases linearly with the number of quantizers  $M$ . Once the full trellis has been grown, determining the optimal path involves backtracking from the final stage to the initial stage. This requires one addition per stage, which we can neglect in estimating the order of complexity. The total complexity of growing the trellis is thus

$$C_v = \mathcal{O}(B_{\max}NM) \quad (10)$$

Note that in the sliding window version of the VA (see Section IV-A-3), the complexity of generating the trellis remains the same, but the cost of backtracking is no longer negligible since backtracking has to be performed for every iteration.

2) *Solution Using Constant Slope Optimization:* Consider Algorithm 3, with  $n$  blocks being used for each iteration of the Lagrangian optimization. Finding the path for each set of  $n$  blocks requires, on the average,  $l$  iterations of Algorithm 2. In each iteration of Algorithm 2, i.e., for a fixed  $\lambda$ , one has to find, for each of the  $n$  blocks, the quantizer that minimizes  $(r_{ij} + \lambda d_{ij})$ , which, as before, makes the complexity increase linearly with  $M$ . Note that recomputing  $\lambda$  in Step 1 of each iteration of Algorithm 2 is negligible compared with the  $n \cdot M$  comparisons required in Step 2. This procedure is repeated for each of the  $N$  blocks. Since  $l$  remains essentially independent of the other parameters of the problem, an estimate of the complexity is

$$C_s = \mathcal{O}(nNM). \quad (11)$$

In practice, values of  $n$  that yield a good performance are of the order of  $k \log(B_{\max})$ , where  $k$  is a constant; therefore, in general, we have that  $n \ll B_{\max}$ .

3) *Experimental Comparison:* To give a better idea of the tradeoffs between complexity and performance involved in choosing one of the algorithms of Section IV, we now look at the results of a typical example (see Table II). We fix the sequence, rate, and quantizer set and compute, for buffer

TABLE II  
COMPARISON OF DISTORTION AND EXECUTION TIME FOR (i) VA, (ii) CONSTANT SLOPE OPTIMIZATION WITH LENGTH 200 AND (iii) LENGTH 400, AND (iv) HEURISTIC APPROXIMATION WITH 10% THRESHOLD

	$B_{\max}$	2000	3000	4000
	$\log_2(B_{\max})$	10.96	11.55	11.96
(i)	Time (s)	82.8	121.8	162.3
	SNR (dB)	37.93	37.96	37.98
(ii)	Time (s)	77.8	81.1	83.6
	SNR (dB)	37.92	37.93	37.95
(iii)	Time (s)	150.1	156.1	157.7
	SNR (dB)	37.92	37.94	37.96
(iv)	Time (s)	7.9	4.1	5.2
	SNR (dB)	37.91	37.90	37.84

sizes (in bits) of 2000, 3000, and 4000, the solution using the following algorithms:

- i) Algorithm 1 (optimal solution)
- ii) Algorithm 3 with  $n = 200$  blocks (“short look-ahead”)
- iii) Algorithm 3 with  $n = 400$  blocks (“long look-ahead”)
- iv) Algorithm 4 with 10% heuristic.

With the obvious disclaimer that the precise values depend on the chosen sequence, some general observations can be made:

- The suboptimal algorithms afford good approximations to the optimal values.
- As just derived, the computation time for the VA increases linearly with the the buffer size, whereas the increase is only logarithmic for the Lagrangian-based algorithms. In the limit, as the buffer size goes to infinity, we would be dealing with an unconstrained allocation problem for which the VA is clearly nonpractical, whereas Lagrangian approaches can be efficiently used [11], [12], [23].
- A comparison between the execution times of the VA and Algorithm 3 is unfair since Algorithm 3 suffers from the overhead required by a sliding window optimization. A fairer comparison would be between Algorithm 3 and the sliding window version of the VA. Our results indicate that the former is significantly faster than the latter.
- Clearly, the heuristic methods of Algorithm 4 have an edge in terms of speed, although the gains are very dependent on the sequence. In the limit, if a “constant slope solution” to the problem exists (i.e., if the buffers play no role and the optimal solution is the same as in the unbuffered case for the same total rate), then Algorithm 4 would find the solution in one iteration.

## V. APPLICATIONS

This section is devoted to describing several specific applications of the theoretical analyses and algorithms presented thus far. We first use an optimal solution to the buffer-constrained quantization problem to determine the appropriate buffer size for a given source and application. We also show the relevance of our methods in providing a benchmark for

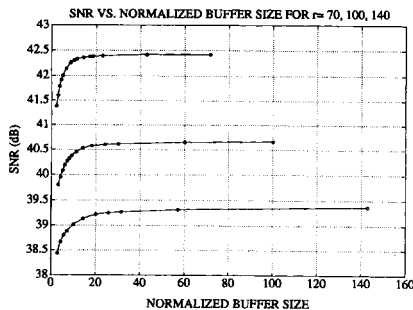


Fig. 12. SNR versus normalized buffer size. The normalized buffer size ( $B_{\max}/r$ ) represents the size of the memory expressed in number of blocks coded at the nominal rate of  $r$  bits. Note that for different values of  $r$ , the curves have a similar shape, and the optimal normalized buffer size is roughly the same.

buffer control algorithms as well as to analyze the statistical multiplexing gain in ATM networks. Our intention here is not to be exhaustive but rather illustrative.

#### A. Optimal Buffer Size

The first question that arises in the study of our buffer constrained problem is that of the optimal buffer size. As we have seen, a budget-constrained unbuffered quantization provides the lower bound for the distortion incurred. How much does the buffer size have to be increased so that the buffer constraints no longer make a difference and the allocation is the same as in an unbuffered situation? It might be interesting to examine what the minimal buffer size might be where the buffer becomes “transparent” to the bit allocation problem. This is relevant in light of the different applications that were outlined in Section II-C. Although in some cases, like real-time services, the maximum buffer size that can be used in the system is limited by the total delay  $\Delta T$ , it is apparent that in other applications, like video CD-ROM’s, the memory to be used is essentially a design choice with no limitation other than cost. In such a case, the buffer size need not be bigger than the maximum length of oscillation in buffer occupancy when a “constant slope” is maintained.

In our experiments, for a given sequence, channel rate, and set of quantizers, we find the optimal solution for several buffer sizes. As can be seen in Fig. 12, the marginal gain of increasing the buffer size decreases as the buffer size gets large. This gain becomes zero once the maximum buffer size that a “constant slope” solution generates has been reached. The buffer size to choose, if the choice is possible, is that one for which the optimal value is “just” reached and necessarily depends on the variability of the source. In the curves, in order to remove the dependence on the channel rate, the buffer size is normalized by the channel rate so that the horizontal axis represents the maximum number of blocks that could fit in the buffer if coded at the nominal rate. It can be seen that all three curves, corresponding to three channel rates for the same sequence, have the “knee” at roughly the same normalized

buffer size. This result indicates that, as expected, the optimal buffer size is a function of the sequence to be coded. Note that the knee is reached first for the higher channel rates. This happens because the set of quantizers is maintained constant in all three experiments so that as rate increases there are fewer possible choices. In the limiting case, if the channel rate was above the maximum rate of the highest quality quantizer in the set, the optimal solution would be reached for any buffer size since the buffer would be permanently in underflow.

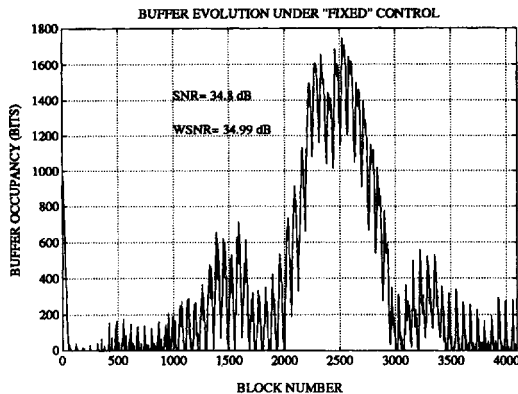
Although one cannot guarantee that a given buffer size will be appropriate for a given nonstationary video sequence, if the range of variability of the source is known, a good choice for the buffer size can still be made. Finally, it is worth noting that for some applications, the buffer size can indeed be tailored for each specific sequence. Consider, for instance, a workstation display of a previously coded sequence. In this environment, since delay is not a factor, it would be possible to decide on different buffer sizes, depending on the specific characteristics of the sequence by prefixing, in the sequence header, the optimal buffer size that was chosen during the coding process.

#### B. Benchmarking

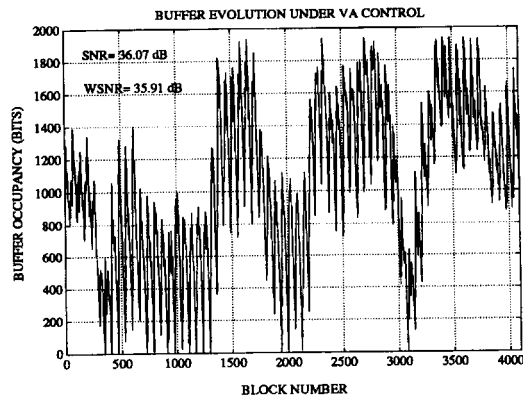
Many different applications require a real-time buffer control strategy, where complexity considerations may not permit the use of the methods of Section IV-B. In that case, given a sequence, we can compute the optimal buffer-constrained allocation and thus have a *benchmark* serving as an upper bound for performance in assessing real-time algorithms. Typical examples would be workstation display, video on CD-ROM, buffered JPEG, or MPEG. In this section, we present an example of the use of benchmarking for an activity-weighted buffered JPEG-like coding environment (unlike JPEG, we allow different quantizer scales to be used for each block within a frame).

As discussed in Section II-A, the only requirement we impose on the cost measure to be minimized is that it be additive. As an example, we consider a typical DCT-based coding scheme, such as JPEG, and take as an activity measure the sum of the absolute values of the DCT coefficients of each block. Then, we choose as cost measure for each block the MSE divided by the activity (so that higher activity blocks are permitted to have a higher MSE).

We now consider a practical scheme that might be used and assess its performance against the “optimal” benchmark provided by the VA. While distributing the allocated rate among the blocks in a frame, the criterion should tend to use finer quantizers for the lower activity blocks (for which coding errors are more noticeable), whereas coarser quantization can be afforded for the higher activity blocks. Given the nominal bit allocation budget per frame, the mean activity of the frame  $\bar{A}$  is computed, and an activity classification  $A_i$  is performed for each block  $i$ . Blocks with an activity index  $A_i$  close to the mean  $\bar{A}$  are assigned the “nominal” quantizer, whereas those deviating from the mean are coded using finer quantizers (if their activity is less than average) or coarser quantizers (if their activity is more than average). Thus, for a real-time coding



(a)



(b)

Fig. 13. Comparison of “fixed” activity-based allocation and optimal buffer-constrained allocation. In both figures, the channel rate is  $r = 64$  bits/block, and the buffer size  $B_{\max} = 2000$  bits.  $WSNR$  denotes the activity-weighted SNRL: (a) Buffer evolution for “fixed” control. For the chosen buffer size and nominal quantizer, there is underflow at both the beginning and the end of the frame; (b) buffer evolution under VA allocation.

scheme, a fixed mapping is set up between the quantizer grade chosen and the relative activity ( $A_i/\bar{A}$ ). Note that the nominal quantizer is chosen so that the number of bits actually allocated to the frame is close to the desired objective.

We can now estimate the suboptimality of such a rate control scheme when it is used in a finite buffer environment. Suppose the practical scheme described above is used with a buffer size  $B_{\max}$  that just prevents overflow when the buffer empties out at a “channel rate”  $r$  (see Fig. 13(a)). Note that for the example considered, in order to accommodate the coded stream in the buffer, we need to use a channel rate higher than the average coding rate of the source due to the peakiness of the source. In Fig. 13(b), we show how we can get an optimal upper bound to the performance given the same conditions (set of quantizers  $r$ ,  $B_{\max}$ ) by using the VA with an activity-weighted MSE cost measure. In our results, for the coding of one frame,

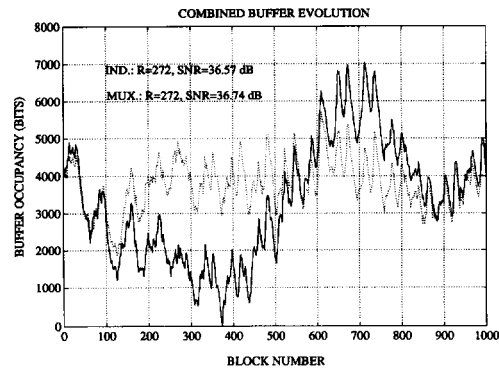


Fig. 14. Buffer evolution in the common buffer for the four sources when the allocation is computed using a combined or multiplexed VA (solid line) or an independent VA (dashed line). The resulting total SNR (obtained by averaging the MSE of all sources) increases, as was to be expected when a combined VA is used for the allocation. The independent VA uses one fourth of the rate and buffer size of the combined scenario independently for each source.

VA proves to be better than a “fixed” mapping by 1 dB in both SNR and weighted SNR (WSNR).

### C. Statistical Multiplexing of ATM Sources

Within an ATM network, the user negotiates with the network the parameters of each connection, specifying the mean channel rate, maximum peak rate available to the user, etc. [24]. Thus, several VBR sources would be able to share transmission resources and buffers within the network, so that advantage can be taken of what has been called *statistical multiplexing gain* (SMG). The bit stream combining the multiplexed sources has a less peaky behavior (i.e., its “peak to mean rate” ratio is smaller than that of the individual sources), and therefore, the probability of buffer overflow is lower. Studies of SMG for packet video [25], [26] have expressed the gain only in terms of reducing overflow probability, i.e., looking at the problem from a network point of view. In [27], it was shown how, for certain sources such as speech and video, end-to-end SNR is a more meaningful measure of performance than packet loss. This fact was used to propose a system of packet priorities so that under congestion, the least “important” information, in the distortion sense, is lost first. We look at the problem of statistical multiplexing also from a source coding point of view (i.e., taking into account both rate and distortion) and show an example of how to use the previously described techniques to evaluate the maximum performance gain of source multiplexing in an ATM environment.

It has to be noted that our experiment does not take into account the fact that sources will be policed by the network [24], and thus, no restrictions are imposed on the parameters (peak and mean rate) of each individual source being multiplexed. Our results provide a bound on the achievable performance given the set of sources and the bit rate available to transmit them and would also be applicable to cases, for instance within MPEG, where several video sources are simultaneously coded

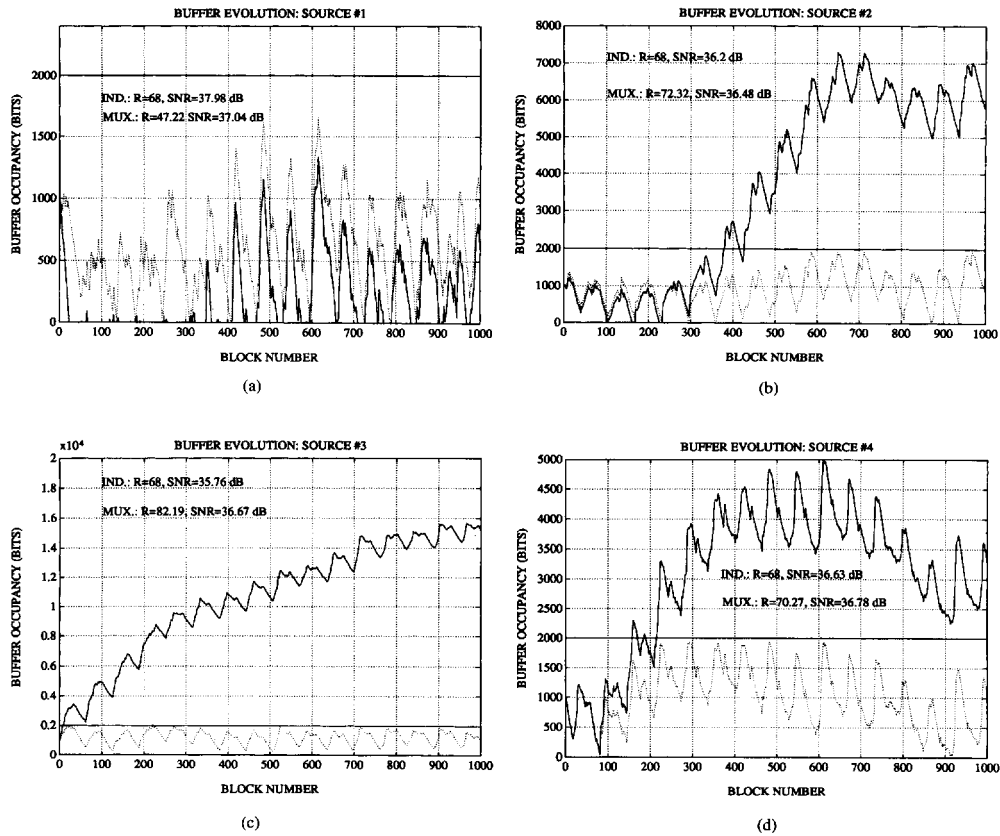


Fig. 15. Individual buffer occupancies for each source: (a) Source 1; (b) source 2; (c) source 3; (d) source 4. The independent buffer evolution (dashed line) corresponds to a buffer size of 2000. The combined or multiplexed buffer evolution (solid line) is simulated based on the combined allocation and assuming that the channel rate was the same as in the independent case, i.e., one fourth of the total rate. Observe that the most demanding sources (2, 3, 4) increase their effective rates in the combined case at the expense of the least demanding source (1). Overall, the same total rate is used and, as seen in Fig. 14, the overall SNR is improved when doing a combined allocation.

by a single coder and then multiplexed into a single CBR bit stream. The problem of optimizing source coding under policing constraints is studied in [28].

The experiment is necessarily source dependent and is given as an example of what a typical behavior might be. One could expect higher gains if the sources are highly uncorrelated (different sources having peaks during different time intervals), whereas gains would have to be modest for correlated sources. In fact, in the limit of high correlation, when all sources are identical, there would be no gain in using multiplexing.

In our experiment, we consider four sources, each consisting of 1000 blocks, that are multiplexed into a buffer of size  $4 \cdot B$  before being sent through a channel of rate  $4 \cdot R$ . As stated, this problem is equivalent to the one we have treated previously; therefore, we can use the VA to determine the optimal allocation given the buffer constraints. This can be done in two ways:

- (i) Allocate the rate *independently* for each source, i.e., use the VA on each one assuming a rate of  $R$  and a buffer size of  $B$ . The VA minimizes the distortion of each source.
- (ii) Perform the VA in a *combined* fashion, i.e., for a rate of  $4 \cdot R$  and a buffer size of  $4 \cdot B$ . At each stage, the algorithm determines the quantizers to be used in each of the sources, taking into account the combined rate distortion characteristic. With this approach, the VA minimizes the total distortion of the four sources.

Fig. 14 shows the evolution of the multiplexing buffer under methods (i) and (ii) (dashed and solid lines, respectively). Note that when the independent allocation is used, *the buffer is not fully used*. In contrast, when a combined allocation is performed, the buffer reaches near-full and near-empty levels during the simulation. For this particular example, the gain achievable by using a combined VA is nearly 0.2 dB for the average distortion. However, the individual distortions of each of the sources with either method are of more interest (see

Fig. 15). Our results show that in order to improve the overall quality, one source is allocated fewer bits by the combined algorithm (see Fig. 15(a)), whereas the other three (Fig. 15(b), (c), and (d)) increase their total allocation. It can be seen that source 3 increases its SNR by almost 1 dB, whereas source 1 loses close to 1 dB. Overall, minimizing the total distortion tends to average out the SNR's of the individual sources. Note that there is also a gain from the buffer size point of view: for sources 2, 3, and 4 to reach the same distortion values as with method (ii), the buffers used with method (i) would need to be much larger than the original size of  $B_{\max} = 2000$  bits.

## VI. CONCLUSION

In this work, a global view of the problem of buffer-constrained quantization has been presented. A detailed study of the problem from a theoretical point of view is undertaken, and based on it, practical methods to compute the optimal solution to the problem are described. In particular, we present a class of nearly optimal fast algorithms exhibiting a substantial reduction in computational complexity. We have shown some possible applications for our algorithms, such as benchmarking for buffer control algorithms, determination of the optimal buffer size for a given coding environment, and analysis of statistical multiplexing gain in ATM networks. Additionally, our optimal algorithm is useful for applications where processing delay is not important, like broadcast, video on CD-ROM, workstation display, or buffer-constrained JPEG. The case of dependent quantization, that is, when the bit rate of a given block depends both on the quantizer used and on the previous block (the simplest example would be interframe DPCM video sequence coding), is currently under investigation.

## ACKNOWLEDGMENT

The authors wish to thank Prof. M. Pinedo from Columbia University for useful discussions on Lagrangian techniques for integer programming and Dr. D. LeGall from C-Cube Microsystems for helpful comments on activity-weighted allocation.

## REFERENCES

- [1] N. Farvardin and J. W. Modestino, "On the overflow and underflow problems in buffer-instrumented variable-length coding of fixed-rate memoryless sources," *IEEE Trans. Inform. Theory*, vol. IT-32, pp. 516-532, Nov. 1986.
- [2] G. K. Wallace, "The JPEG still picture compression standard," *Commun. ACM*, vol. 34, pp. 30-44, Apr. 1991.
- [3] D. LeGall, "MPEG: A video compression standard for multimedia applications," *Commun. ACM*, vol. 34, pp. 46-58, Apr. 1991.
- [4] B. G. Haskell and R. L. Schmidt, "A low bit-rate interframe coder for the videotelephone," *Bell Syst. Tech. J.*, vol. 54, pp. 1475-1495, Oct. 1975.
- [5] J. -P. Leduc and S. D'Agostino, "Universal VBR videocodecs for ATM networks in the Belgian broadband experiment," *Image Commun.*, vol. 3, pp. 157-165, June 1991.
- [6] J. Zdepsky, D. Raychaudhuri, and K. Joseph, "Statistically based buffer control policies for constant rate transmission of compressed digital video," *IEEE Trans. Commun.*, vol. 39, pp. 947-957, June 1991.
- [7] A. Ortega, J. I. Ronda, N. García, and P. J. Zufria, "A model based approach to buffer occupancy control for TV and HDTV coders," in *Proc. 4th Int. Workshop HDTV* (Turin), Sept. 1991.
- [8] J. I. Ronda, A. Ortega, A. Fernández, and N. García, "Predictive control of variable bit rate video coders," in *Proc. 6th Euro. Signal Processing Conf., EUSIPCO '92* (Brussels), Aug. 1992, pp. 1369-1372, vol. 3.
- [9] T. Hamada, M. Saito, and S. Matsumoto, "Optimal quantization by viterbi algorithm for HDTV coding," in *Proc. 4th Int. Workshop HDTV* (Turin), Sept. 1991 (Abstract).
- [10] A. Ortega, K. Ramchandran, and M. Vetterli, "Optimal buffer-constrained source quantization and fast approximations," in *Proc. IEEE Int. Symp. Circ. Syst. ISCAS '92* (San Diego), pp. 192-195, May 1992.
- [11] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," *IEEE Trans. Signal Processing*, vol. 36, pp. 1445-1453, Sept. 1988.
- [12] S.-W. Wu and A. Gersho, "Rate-constrained optimal block-adaptive coding for digital tape recording of HDTV," *IEEE Trans. Circuits Syst. Video Tech.*, vol. 1, pp. 100-112, Mar. 1991.
- [13] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Optimal pruning with applications to tree-structured source coding and modeling," *IEEE Trans. Inform. Theory*, vol. IT-35, pp. 299-315, Mar. 1986.
- [14] G. L. Nemhauser and L. A. Wolsey, *Integer and combinatorial optimization*. New York: Wiley, 1988.
- [15] M. Minoux, *Mathematical Programming: Theory and Algorithms*. New York: Wiley, 1986.
- [16] M. L. Fisher, "The Lagrangian relaxation method for solving integer programming problems," *Management Sci.*, vol. 27, pp. 1-18, Jan. 1981.
- [17] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding*. New York: McGraw-Hill, 1979.
- [18] G. D. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268-278, Mar. 1973.
- [19] J. G. Proakis, *Digital Communications*. New York: McGraw-Hill, 1989.
- [20] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. on Inform. Theory*, vol. IT-28, pp. 55-67, Jan. 1982.
- [21] M. W. Marcellin and T. R. Fischer, "Trellis coded quantization of memoryless and gauss-markov sources," *IEEE Trans. Commun.*, vol. 38, pp. 82-93, Jan. 1990.
- [22] H. Everett, "Generalized Lagrange multiplier method for solving problems of optimum allocation of resources," *Oper. Res.*, vol. 11, pp. 399-417, 1963.
- [23] K. Ramchandran and M. Vetterli, "Best wavelet packet bases in a rate-distortion sense," *IEEE Trans. Image Processing*, Apr. 1993.
- [24] E. P. Rathgeb, "Modeling and performance comparison of policing mechanisms for ATM networks," *IEEE J. Selected Areas Commun.*, vol. 9, pp. 325-334, Apr. 1991.
- [25] B. Maglaris, D. Anastassiou, P. Sen, G. Karlsson and J. Robbins, "Performance models of statistical multiplexing in packet video communications," *IEEE Trans. Commun.*, vol. 36, pp. 834-843, July 1988.
- [26] W. Verbiest, L. Pinnoo, and B. Voeten, "The impact of the ATM concept on video coding," *IEEE J. Selected Areas Commun.*, vol. 6, pp. 1623-1632, Dec. 1988.
- [27] M. W. Garrett and M. Vetterli, "Joint source/channel coding of statistically multiplexed real time services on packet networks," *IEEE/ACM Trans. Networking*, vol. 1, pp. 71-80, Feb. 1993.
- [28] A. Ortega, M. W. Garrett and M. Vetterli, "Toward joint optimization of VBR video coding and packet network traffic control," in *Proc. 5th Packet Video Workshop* (Berlin), Mar. 1993.

**Antonio Ortega** (S'92) was born in Madrid, Spain, in 1965. He received the Telecommunication Engineering degree from the Universidad Politécnica de Madrid (UPM), Madrid, Spain, in 1989 and is currently working toward the Ph.D. degree in electrical engineering at Columbia University, New York, NY. In 1986 and 1987, he worked during the summer for Gibbs & Hill, Inc., Washington, DC. In 1988, he was a summer intern at the Optoelectronics Lab, ENST, Paris, France. During 1990, he was a research assistant at the Image Processing group at UPM. Since January 1991, he has been a Research Assistant at the Center for Telecommunications Research, Columbia University. His work is being supported by a scholarship from the Fulbright Commission and the Spanish Ministry of Education. His research interests include digital image and video processing and communications.

**Kannan Ramchandran** (S'92) was born in Madras, India, in 1961. He received the B.S. in E.E. degrees from the City College of New York in 1982 and the M.S. and M. Phil. degrees in electrical engineering from Columbia University in 1984 and 1992, respectively.

From 1982 to 1984, he worked for IBM as a microprocessor systems designer. In 1984, he joined AT&T Bell Labs as a Member of the Technical Staff, where he designed digital line multiplexers, phase-locked loops, synchronization circuitry, and custom transmission IC's in the optical fiber loop carrier system and the fiber-to-the-home development system. He is currently on educational leave from AT&T to finish his Ph.D. dissertation at the Center for Telecommunications Research at Columbia. His research interests include multiresolution signal processing, adaptive quantization, fast algorithms, and architectures for image and video coding, digital telecommunications, wavelets, time-frequency analysis, and packet video.

**Martin Vetterli** (S'86-M'86-SM'90) was born in Switzerland in 1957. He received the Dipl. El.-Ing. degree from the Eidgenössische Technische Hochschule Zürich, Switzerland, in 1981, the Master of Science degree from Stanford University, Stanford, CA, in 1982, and the Doctorat ès Science degree from the Ecole Polytechnique Fédérale de Lausanne, Switzerland, in 1986.

In 1982, he was a Research Assistant at Stanford University, and from 1983 to 1986, he was a researcher at the Ecole Polytechnique. He has worked for Siemens and AT&T Bell Laboratories. In 1986, he joined Columbia University in New York, where he is currently Associate Professor of Electrical Engineering, member of the Center for Telecommunications Research, and codirector of the Image and Advanced Television Laboratory. Since 1992, he has been an Acting Associate Professor of Electrical Engineering and Computer Science at the University of California at Berkeley. His research interests include wavelets, multirate signal processing, computational complexity, signal processing for telecommunications, and digital video processing and compression.

He is a member of SIAM and ACM, a member of the MDSP committee of the IEEE Signal Processing Society, and he is on the editorial boards of *Signal Processing*, *Image Communication*, *Annals of Telecommunications Applied and Computational Harmonic Analysis*, and *The Journal of Fourier Analysis and Applications*. He received the Best Paper Award of EURASIP in 1984 for his paper on multidimensional subband coding, the Research Prize of the Brown Boveri Corporation (Switzerland) in 1986 for his thesis, and the IEEE Signal Processing Society's 1991 Senior Award (DSP Technical Area) for a 1989 Transactions paper with D. LeGall on filter banks for subband coding. He was a plenary speaker at the 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing and is the co-author, with J. Kovacevic, of the forthcoming book *Wavelets and Subband Coding* (Prentice-Hall, 1994).