Pre-print of "Fast rate allocation for JPEG2000 video transmission over time-varying channels / L. Jiménez-Rodríguez, et al." in IEEE Transactions on multimedia (IEEE), jan. 2013, vol. 15, issue 1, p. 15-26. DOI 10.1109/TMM.2012.2199973

FAST Rate Allocation for JPEG2000 Video Transmission over Time-Varying Channels

Leandro Jiménez-Rodríguez, Francesc Aulí-Llinàs, *Member, IEEE* and Michael W. Marcellin, *Fellow, IEEE*

***Abstract*—This work introduces a rate allocation method for the transmission of pre-encoded JPEG2000 video over time-varying channels, which vary their capacity during video transmission due to network congestion, hardware failures, or router saturation. Such variations occur often in networks and are commonly unpredictable in practice. The optimization problem is posed for such networks and a rate allocation method is formulated to handle such variations. The main insight of the proposed method is to extend the complexity scalability features of the FAst rate allocation through STeepest descent (FAST) algorithm. Extensive experimental results suggest that the proposed transmission scheme achieves near-optimal performance while expending few computational resources.**

***Index Terms*—Video transmission, time-varying channels, rate allocation, JPEG2000**

I. Introduction

Video transmission has been a prominent research topic for the last few decades. Its deployment in myriad applications, such as teleconferencing, video broadcasting, video-on-demand, and surveillance systems, manifests the consolidation of such technology in our every-day lives.

Three elements are key in the design of a video transmission scheme: the coding system, the network characteristics, and the requirements of the application. Two main families of *coding systems* are currently available for the coding and transmission of images and video: interframe and intraframe. H.264/AVC [1] is the most advanced interframe standard that exploits dependencies among frames to efficiently compress video. JPEG2000 [2] is the most advanced intraframe standard for the coding of images and video without considering frame dependencies. Both standards have been adopted in different scenarios, and both provide powerful tools for transmission of video over a network.

The *characteristics of the network* establish channel properties such as constant or variable channel capacity [3] and communication error rate [4], among others. The *application requirements* may introduce several demands on the transmission scheme: servers that deliver pre-encoded video [5] can use substantially different mechanisms than servers that encode and transmit video on-the-fly [6]; decoders with limited resources may raise challenging constraints [7]; and the use of smart proxies [8] or peer-to-peer (P2P) networks [9], [10] triggers new possibilities to efficiently transmit video.

Despite the large amalgam of scenarios created by the combination of these elements, all video transmission schemes pursue the same goal: to provide the best possible video quality to the end-user. When the distortion measure is mean squared error (MSE), one of two criteria is typically selected to optimize the quality of transmitted video [11]: 1) minimization of the average MSE (MMSE); and 2) provision of (pseudo-)constant quality, which is more commonly expressed as the minimization of the maximum MSE (MMAX) [12]. Of the two, subjective experiments suggest that MMAX may be more relevant perceptually [13]. Although this has been extensively discussed in the literature [14], [15], and even hybrid approaches have been proposed [16], video transmission schemes are generally focused on the optimization of one of these two criteria depending on the requirements of the application.

MMSE and MMAX are achieved by means of reducing/increasing the number of bytes transmitted for each frame, which is referred to as variable bitrate (VBR) video. Intuitively, VBR video delivers more bytes for those frames that are more difficult to compress (high spatial activity and/or motion) than for those frames that are easier to compress. The process that decides the number of bytes that are transmitted for each frame is called rate allocation, which is a key piece of video transmission schemes. Rate allocation methods must take into account the optimization criterion together with the coding system, the network characteristics, and the constraints imposed by the application.

This work considers a video-on-demand scenario that transmits pre-encoded JPEG2000 video to clients over a time-varying channel. To allow VBR video, the client has a limited buffer capacity to absorb irregularities in the sizes of compressed frames. We assume that the buffer size may vary from client to client, and that the channel capacity may vary over time in an unpredictable manner. We adopt JPEG2000 as the coding system since its fine grain quality scalability facilitates rate allocation of pre-encoded video. Furthermore, it is employed in many motion imagery applications, such as Digital Cinema distribution, television production, and surveillance.

Leandro Jiménez-Rodríguez and Dr. Francesc Aulí-Llinàs are with the Department of Information and Communications Engineering, Universitat Autònoma de Barcelona, Spain (phone: +34 935813575; fax: +34 935814477; e-mail: {ljimenez, fauli}@deic.uab.es). Prof. Michael W. Marcellin is with the Department of Electrical and Computer Engineering, University of Arizona, USA (e-mail: marcellin@ece.arizona.edu). This research was carried out when he was visiting professor and Marie Curie Fellow at the Department of Information and Communications Engineering, Universitat Autònoma de Barcelona, Spain. This work has been partially supported by the Universitat Autònoma de Barcelona, by the Spanish Government (MICINN), by the European Union, by FEDER, and by the Catalan Government, under Grants UAB-472-01-2/09, RYC-2010-05671, FP7-PEOPLE-2009-IIF-250420, TIN2009-14426-C02-01, and 2009-SGR-1224.

The rate allocation algorithm introduced in this paper builds on our previous approach FAst rate allocation through STeepest descent (FAST) [17]. The main shortcoming of FAST is that, as originally formulated, it can *not* absorb variations in channel capacity during transmission. Such variations occur frequently in real-world scenarios that transmit data over the Internet or wide area networks due to congestion, irregularities in network conditions, etc.

An important feature required in time-varying channels is that the rate allocation method absorbs channel irregularities while guaranteeing that the transmission does not violate any existing buffer limits at the client. Important aspects of the optimization problem are that the variations in channel capacity are not known a priori, and that the server cannot interrupt the video transmission to compute new frame rates when channel conditions vary. The method introduced in this work extends two particular features of FAST to deal with such variations: scalability in terms of complexity, and the roughly linear relation between computational load and the number of frames. This permits the introduction of efficient strategies that can handle variations in channel capacity without penalizing performance. Furthermore, the proposed method preserves interesting features of the original FAST algorithm such as optimization for MMSE or MMAX, and low memory requirements.

The paper is organized as follows. Section II describes the fundamentals of video transmission schemes. Section III establishes the optimization problem that arises in time-varying channels and describes the proposed algorithm. Section IV assesses the performance of the proposed algorithm through extensive experimental results. Section V provides concluding remarks.

II. Overview of video transmission schemes

The simplest scheme to transmit video is to use a Constant Bit Rate (CBR) policy that transmits the same rate (bits/frame) for all frames of the video sequence. Although CBR schemes maintain constant client buffer occupancy throughout the whole transmission, the video quality is not optimized.

Using Variable Bit Rate (VBR) policies can provide the opportunity to optimize video quality. Nonetheless, VBR schemes introduce constraints to the optimization problem that have to be addressed carefully. Specifically, let R^{total} be the total rate (bits/sequence) used to satisfy a client request for a sequence, and let N be the number of frames of the sequence. For now, we assume that the channel capacity is fixed at constant W bits per second and that the rendering pace is $\mathcal{F}$ frames per second (fps). Then, R^{total} is determined as $R^{total} = (W/\mathcal{F}) \cdot N$. Suppose that the codestream for the ith frame is scalable and can be truncated at points j corresponding to increasing bitrates r_{ij} (bits/frame) and decreasing distortions d_{ij}, with $1 \leq i \leq N$ and $1 \leq j \leq Q_i$, where Q_i is the number of truncation points available for frame i. When the optimization criterion is MMSE, the objective of the optimization problem is to find the truncation points $\mathbf{x} = \{x(1), x(2), ..., x(N)\}$ corresponding to bitrates $r_{ix(i)}$ and distortions $d_{ix(i)}$ that minimize the distortion, do not exceed the bit budget, and respect the client buffer size, i.e.,

$$\min_{\mathbf{x}} \sum_{i=1}^{N} d_{ix(i)} \tag{1}$$

such that

$$\sum_{i=1}^{N} r_{ix(i)} \leq R^{total} \tag{2}$$

and

$$B^{min} \;\leq\; \frac{R^{total}}{N} \cdot f - \sum_{i=1}^{f} r_{ix(i)} \;\leq\; B^{max} - \frac{R^{total}}{N} \tag{3}$$

$$\forall\ f,\ 1 \leq f \leq N\ ,$$

where B^{min} and B^{max} denote the minimum and maximum capacity of the client buffer, respectively, with $B^{min} < B^{max}$. The middle expression of inequality (3), $B(f) = \frac{R^{total}}{N} \cdot f - \sum_{i=1}^{f} r_{ix(i)}$, represents the buffer occupancy at the instant just after frame f is rendered. As discussed above, the total available transmission rate for the sequence is R^{total}, and the channel capacity W is assumed constant. So the rate transmitted per frame rendering period can be expressed as $W/\mathcal{F} = R^{total}/N$. It is worth noting that the frame period is constant (for example, $1/30$ second, corresponding to $\mathcal{F} = 30$ frames/second) even though the time to transmit the data for each frame is variable due to VBR encoding. Thus, data can be seen as entering the buffer at the constant rate of R^{total}/N bits per frame period. The total rate received up to frame f is therefore $\frac{R^{total}}{N} \cdot f$. On the other hand, each time a frame is rendered, its data are removed from the buffer, emptying $r_{ix(i)}$ bits from the buffer for frame i. Thus, the total rate emptied from the buffer up to frame f is expressed as $\sum_{i=1}^{f} r_{ix(i)}$. The difference between the filling and emptying corresponds to the buffer occupancy, $B(f)$ as expressed in the middle term of inequality (3). Recalling again that $B(f)$ is the buffer occupancy *just after* frame f is removed from the buffer, the right hand expression of (3) can be understood. During the frame period that occurs after frame f is removed, and before frame $f+1$ is removed, R^{total}/N bits will be added to the buffer as described above. There must be room in the buffer to accommodate these data, so the buffer occupancy just after frame f is rendered must be no greater than $B^{max} - R^{total}/N$.

In practice, a certain amount of buffering delay is required to partially fill the buffer prior to any frames being rendered. In order to avoid cluttering the notation by including this delay and the resulting initial data in the buffer, we take $t = 0$ be the instant just before the first frame is rendered. Furthermore, we

take $B(f)$ to be the buffer occupancy *relative* to the amount of data initially buffered, say B^0. The amount of data actually in the buffer just after frame f is rendered is then $B^0 + B(f)$. In our experiments, we fill the buffer half way prior to rendering the first frame. Thus, for a buffer size of S, we set $B^0 = S/2$, $B^{max} = S/2$ and $B^{min} = -S/2$.

With respect to the MMAX criterion, the formulation of the optimization problem is the same except that the objective function (1) is replaced by

$$\min_{\mathbf{x}} \left(\max_{i=1}^{N} \; d_{ix(i)} \right) \; . \tag{4}$$

It has been shown that both optimization problems (i.e., that of (1) (2) (3), and that of (4) (2) (3)) can be solved within the same optimization framework [14], so some methods proposed in the literature are able to address both MMSE and MMAX.

Many schemes for video transmission have been explored since the mid-90s [18], [19]. Three main approaches have proven effective to tackle the optimization problem above: dynamic programming techniques [7], Lagrange relaxation methods [20], [21], and steepest descent algorithms [8], [22], [23]. Commonly, dynamic programming techniques construct a trellis structure that contains all solutions to the problem. The application of the Viterbi algorithm over the trellis reaches the optimal solution. The main disadvantage of this approach is that it requires high memory resources to build the trellis, and high computational load to search the trellis. Lagrange relaxation methods reduce computational requirements by relaxing the constraints of the optimization problem.

The use of steepest descent techniques leads to more efficient rate allocation methods. The steepest descent algorithm employed by our previous work FAST [17] selects a trivial valid solution to the problem (potentially poor), and then iteratively makes small changes to the solution following some heuristic. The heuristic for the steepest descent when the optimization criterion is MMSE is the Lagrange cost [24]. Generally speaking, the Lagrange cost measures the compression efficiency achieved at different truncation points of the compressed codestream. In the JPEG2000 framework [2], the Lagrange cost is embodied in the distortion-rate slope. If r_{ij} and d_{ij} respectively denote the rate and distortion at the jth truncation point for frame i, the distortion-rate slope at this point is defined as

$$S_{ij} = \frac{d_{i(j-1)} - d_{ij}}{r_{ij} - r_{i(j-1)}} \; . \tag{5}$$

Truncation points are represented as quality layers within the JPEG2000 codestream. The distortion-rate slope of each layer can be recorded within the codestream. If layer fragmentation is desired, distortion-rate slopes at intra-layer fragmentation points can be estimated using a linear form as described in [17]. Accordingly, more truncation points for frame i can be added. The use of S_{ij} allows FAST to exclude codestream segments with low distortion-rate slopes (less valuable segments in terms of rate-distortion performance), leaving room for those segments with higher distortion-rate slopes. If heuristic S_{ij} is replaced by d_{ij}, the objective of the algorithm is altered so that it seeks the solution that has the lowest maximum distortion (MMAX) [17].

III. JPEG2000 VIDEO TRANSMISSION OVER TIME-VARYING CHANNELS

A. Optimization problem

We now address the optimization problem that arises in time-varying channels. The capacity of a TCP/IP communication link is commonly determined using the amount of data accepted by the receiving node divided by the round trip time, i.e., $W = \frac{RWIN}{RTT}$, where $RWIN$ is the receive window and RTT denotes the round-trip time [25, Ch. 3.7]. In general, this provides a reliable enough estimate of the channel capacity (or TCP/IP throughput), which can be used by applications such as the proposed rate allocation algorithm. Evidently, each implementation may use different low-level routines to determine the channel capacity, although most are based on the aforementioned principle. FAST-TVC then considers the capacity as a parameter given by the network layer. Our experience indicates that most low-level network routines provide estimates of the channel capacity that are reliable enough to be used by applications. Although it may depend on each implementation, in general these routines detect variations on the channel bandwidth in fractions of a second, so the impact on the convergence and performance of the proposed algorithm is negligible.

For simplicity, we assume piecewise constant capacity. To this end, let C be the number of transmission intervals, each with constant channel capacity, that occur during the transmission of a video sequence. Let $W_c, 1 \leq c \leq C$, be the channel capacity during transmission interval c in bits/second. The total rate available to satisfy the client request is then $R'^{total} = \sum_{c=1}^{C} R_c^{total}$, where R_c^{total} is the total rate in transmission interval c, i.e., $R_c^{total} = W_c \cdot (\mathcal{T}_{c+1} - \mathcal{T}_c)$, with $\mathcal{T}_c$ representing the instant in time at which the channel capacity changes to W_c. For simplicity, we assume that the channel capacity can only change the instant just after a frame is rendered. We then seek the frame truncation points $\mathbf{x} = \{x(1), x(2), ..., x(N)\}$ to achieve

$$\min_{\mathbf{x}} \sum_{i=1}^{N} d_{ix(i)} \tag{6}$$

subject to

$$\sum_{i=1}^{N} r_{ix(i)} \leq R'^{total} \tag{7}$$

and

$$
\begin{aligned}
B^{min} \;&\leq\; B_c + \frac{W_c}{\mathcal{F}} \cdot (f - f_c + 1) - \sum_{i=f_c}^{f} r_{ix(i)} \\
&\leq\; B^{max} - \frac{W_c}{\mathcal{F}} \\
&\forall\ f,\ f_c \leq f < f_{c+1} \text{ and } \forall\ c,\ 1 \leq c \leq C\ ,
\end{aligned}
\tag{8}
$$

where f_c is the first frame rendered after $\mathcal{T}_c$ and B_c is the initial buffer occupancy for the cth transmission interval determined as

$$
B_c = \sum_{k=1}^{c-1} W_k \cdot (\mathcal{T}_{k+1} - \mathcal{T}_k) - \sum_{i=1}^{f_c-1} r_{ix(i)}\ . \tag{9}
$$

It is worth emphasizing that time $\mathcal{T}_c$ falls just after frame f_c-1 is rendered. As in (3), the middle expression of the inequality in (8) represents the buffer occupancy the instant just after frame f is rendered. As in the previous section, replacing the objective function (6) that seeks MMSE by that of (4), the optimization criterion becomes MMAX.

B. Proposed approach

The method proposed below to tackle the optimization problem of (6) (7) (8) is named FAST for time-varying channels (FAST-TVC). The main idea behind FAST-TVC is to use a greedy approach that assumes that the channel capacity will remain fixed throughout the entire transmission of the video. This approach is reasonable, since in a real-time scenario channel capacity changes are not known a priori. When a change does occur, the rates of all non-transmitted frames are recomputed, taking into account the buffer occupancy at the time of the change, but assuming again that there will be no further capacity changes. Assuming infinite computational resources, this would mean simply executing, at each bandwidth change, an instance of FAST with appropriate parameter settings. The main difficulty of this approach is that, absent infinite computational resources, the time required to compute a new solution may be non-negligible and/or unpredictable.

To this end, let t_c denote the time –not known a priori– required by the rate allocation algorithm to reach a solution. When a variation on the channel occurs at $\mathcal{T}_c$, the server continues the transmission of video from $\mathcal{T}_c$ to $\mathcal{T}_c + t_c$ employing frame rates as computed at the beginning of the previous transmission interval $c-1$. This could violate the limits of the client buffer. In practice, if t_c is sufficiently small, the limits of the buffer are not trespassed except in rare occasions. In such cases, if t_c were known, the server could compute a CBR strategy for use during this period that would avoid buffer violations. This calculation could be performed in negligible time. Instead of using the previous solution, a CBR strategy might always be employed from $\mathcal{T}_c$ to $\mathcal{T}_c + t_c$, though the result achieved in both cases is similar since few frames are transmitted during this period. More important is the fact that t_c determines the range of frames that will be

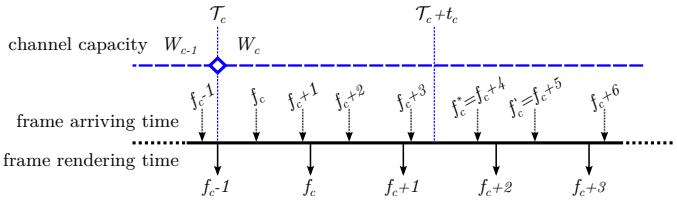


Fig. 1: Example time line of frame arrival and frame rendering times.

re-optimized in response to the bandwidth change, denoted by $[f'_c, N]$. If t_c were known, f'_c could be determined as follows: Let f^*_c be the smallest f such that

$$
\sum_{k=1}^{c-1} W_k \cdot (\mathcal{T}_{k+1} - \mathcal{T}_k) + W_c \cdot t_c < \sum_{i=1}^{f} r_{ix(i)}\ . \tag{10}
$$

Then

$$
f'_c = f^*_c + 1\ . \tag{11}
$$

The left side of Inequality (10) is the total number of bits received at the client up to time $\mathcal{T}_c + t_c$. The right side is the total number of bits received at the client up to and including frame f. Therefore, frame f^*_c is the last frame that begins to be received prior to time $\mathcal{T}_c + t_c$, and so is the first frame to finish being received at the client after $\mathcal{T}_c + t_c$. Thus, f^*_c can *not* be considered by the rate allocation algorithm because at the moment the algorithm finishes execution f^*_c is already partially delivered. The frame after f^*_c (i.e., f'_c) is the first considered by the algorithm since its transmission begins after $\mathcal{T}_c + t_c$. Figure 1 shows an example that illustrates these quantities. In this figure, the arrows above the horizontal line indicate the moment at which the *final bit* of a frame is deposited into the buffer. The arrows below the line indicate the moment at which a frame is removed from the buffer to be rendered.

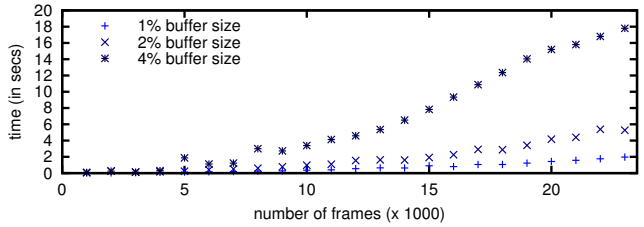
Considering the quantities discussed above, the optimization problem of (6)-(8) can be re-formulated to take into account the time required by the rate allocation algorithm to reach a solution as

$$
\min_{\mathbf{x}} \sum_{i=f'_c}^{N} d_{ix(i)} \tag{12}
$$

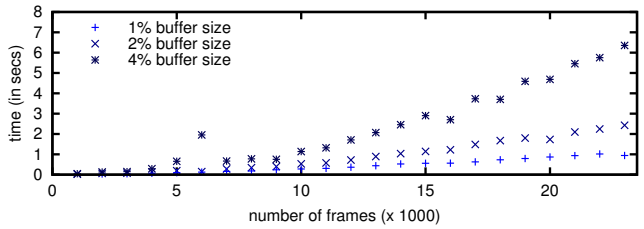
subject to

$$
\sum_{i=f'_c}^{N} r_{ix(i)} \leq R'^{total} - \sum_{i=1}^{f'_c-1} r_{ix(i)} \tag{13}
$$

and



(a) MMSE optimization for "Batman."



(b) MMAX optimization for "Willow."

Fig. 2: Evaluation of the computational load *vs.* the number of frames for different video sequences, buffer sizes, and optimization criteria.

$$B^{min} \;\leq\; B_c + \frac{W_c}{\mathcal{F}} \cdot (f - f_c + 1) - \sum_{i=f_c}^{f} r_{ix(i)} \qquad (14)$$

$$\leq\; B^{max} - \frac{W_c}{\mathcal{F}} \qquad \forall\; f,\; f_c \leq f \leq N\;.$$

Inequality (13) represents the rate constraint for the frames in $[f'_c, N]$. The left side of this inequality is the number of bits to be transmitted for these frames. The right side is the remaining bit budget. Expression (14) is the buffer constraint, which is repeated from Expression (8).

Key to tackling the optimization problem is then to determine t_c *before* the algorithm is actually executed. We propose three approaches to do so. The first approach uses a novel feature of the original FAST algorithm: scalability in terms of complexity. This type of scalability refers to the ability of the algorithm to provide successively improved solutions (in terms of the chosen optimization criterion) as more time is spent in its execution. Complexity scalability allows the server to stop the rate allocation procedure after a predetermined period of time. Thus, t_c can be set by the server arbitrarily, depending on the system load, or by using any other indicator of the operating system. This approach is referred to as "constant t_c."

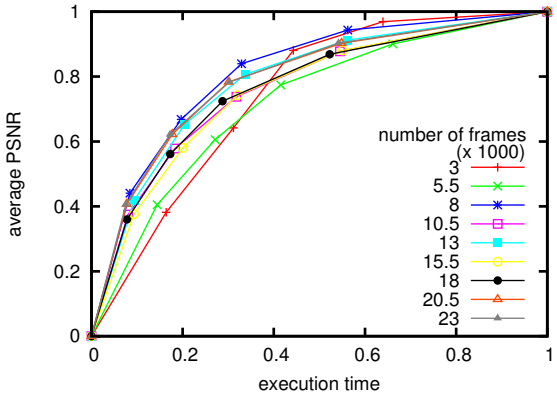
As shown in the next section, the complexity of determining frame rates varies significantly depending on the video sequence, the client buffer size, and the number of frames to be optimized. Hence, the "constant t_c" strategy may lead to significant suboptimalities. The second approach is to estimate the time that the rate allocation algorithm will need to finish the execution as a function of the number of frames to be optimized. The estimation is based on the roughly linear relation between the computational load required by FAST and the number of frames that are optimized. This can be seen from Figure 2(a), which depicts the time spent by FAST when optimizing subsequences having different numbers of frames. Each subsequence is a clip from the "Batman" movie[1]. This figure reports computational load for three different buffer sizes, which are expressed as a percentage of R^{total}. The optimization criterion is MMSE. The time spent by the algorithm increases roughly linearly with the number of frames. This observation also holds for other video sequences and the MMAX optimization criteria (see Figure 2(b)). Let T denote the time spent by the algorithm when all N frames of the sequence are initially optimized at the beginning of transmission interval $c = 1$. The algorithm execution time can then be approximated as $t'_c = N' \cdot \frac{T}{N}$, with N' denoting the number of non transmitted frames at time $\mathcal{T}_c$. This approach is referred to as "estimated t_c." We note that when employing this strategy, the algorithm is terminated at time t'_c even if it has not yet converged. Suboptimality due to this is typically negligible.

Although the "estimated t_c" strategy allows enough time for the algorithm to reach the optimal solution, it does not provide any mechanism to regulate the time spent by the rate allocation procedure. This may be critical when, for instance, the system load is high and resources have to be distributed among different processes. Furthermore, the value of t'_c may be too large, jeopardizing the client buffer as described above. Our third approach combines both the "constant t_c" and "estimated t_c" strategies to allow the server to regulate the time spent by the algorithm without sacrificing performance significantly. The main insight behind this approach comes from the observation that the performance metric improves more rapidly at the beginning of execution than when the algorithm is near convergence.

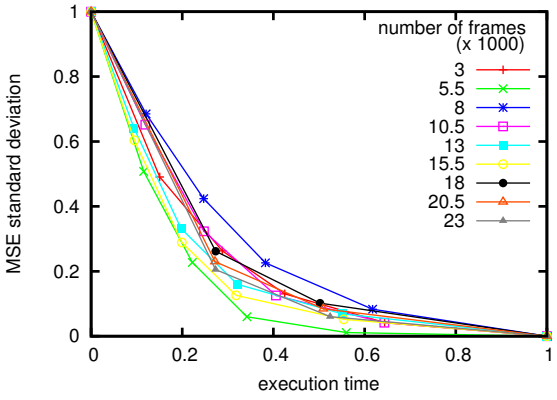
Figure 3(a) depicts the MSE performance metric for solutions provided by the FAST procedure as a function of the time spent by the algorithm. This figure depicts results for a variety of subsequence lengths when the buffer size is 1% of R^{total}. Similar results hold for other buffer sizes and sequences. Both axes of the figure are normalized to allow comparison among different plots. Note that the average PSNR increases very rapidly at the beginning of execution, reaching near-optimal performance in half the time required by the algorithm to converge. Results are similar for the MMAX criterion, and are reported in Figure 3(b) as the MSE standard deviation as a function of algorithm execution time.

These figures and our experience with other sequences indicate that 60% and 80% of the total time is enough to reach a solution very close to the optimal one, respectively for MMSE and MMAX. This can be exploited by the server to set $t_c = \min(t''_c\;, P \cdot t'_c)$, where the first term t''_c is the maximum time allowed by the server (which may depend on the system load or other indicators). The second term $P \cdot t'_c$, with $P = 0.6$ for MMSE and $P = 0.8$ for MMAX, is set to allow the algorithm to reach a near-optimal solution without spending computational resources unnecessarily. This third strategy is referred to as "weighted t_c."

[1]See Section IV for a description of the video sequences and the experimental setup employed herein.



(a) MMSE optimization for "Batman."



(b) MMAX optimization for "Willow."

Fig. 3: Evaluation of the complexity scalability for the MMSE and MMAX criteria.

C. Algorithm

The optimization procedure that seeks the solution to (12) (13) (14) is embodied in Algorithm 1. The algorithm assumes that there is no significant delay between the change in the channel capacity and its detection. As stated previously, when the server first receives a request from a client, it computes frame rates for all frames of the sequence. In the algorithm this is carried out by the procedure "computeFrameRates" (line 7), which receives the first and last frame numbers for the subsequence to be optimized, the channel capacity, buffer limits, current buffer occupancy, and maximum execution time. The procedure "computeFrameRates" is an implementation of the original FAST algorithm as described in [17], which returns solution $\mathbf{x}$ and execution time T. Frames are then transmitted according to the current solution until the end of the sequence is reached or a change in the channel capacity occurs. When the channel capacity changes, the algorithm sets $t_{c^\star}$ in line 17 using one of the three strategies described above. While frame rates for the new channel capacity are being computed in line 20 using the maximum execution time $t_{c^\star}$, frames until $f'_{c^\star}$ are transmitted in the loop of lines 21-24. This process is carried out until all frames are transmitted.

Algorithm 1 FAST-TVC

1: receive client request
2: $c^\star \leftarrow 1$ */* current transmission interval */*
3: $f'_{c^\star} \leftarrow 1$ */* first frame transmitted in interval $c^\star$ */*
4: $B_{c^\star} \leftarrow 0$ */* buffer occupancy at the beginning of interval $c^\star$ */*
5: $i^\star \leftarrow 1$ */* currently transmitted frame */*
6: $W_{c^\star} \leftarrow$ currentChannelCapacity
7: $\mathbf{x}, T \leftarrow$ computeFrameRates$(f'_{c^\star}, N, W_{c^\star}, B^{min}, B^{max}, B_{c^\star}, \infty)$
8: **repeat**
9: **while** the channel capacity remains constant AND $i^\star \leq N$ **do**
10: transmit frame $i^\star$ using $r_{i^\star x(i^\star)}$
11: $i^\star \leftarrow i^\star + 1$
12: **end while**
13: **if** $i^\star \leq N$ **then**
14: $c^\star \leftarrow c^\star + 1$
15: $W_{c^\star} \leftarrow$ currentChannelCapacity
16: $N' \leftarrow N - i^\star$
17: $t_{c^\star} \leftarrow$ estimateAlgorithmTime(T, N') */* using "constant t_c," "estimated t_c," or "weighted t_c" */*
18: $f'_{c^\star} \leftarrow$ according to Equation (11) using $t_{c^\star}$
19: $B_{c^\star} \leftarrow$ according to Equation (9)
20: $\mathbf{x} \leftarrow$ computeFrameRates$(f'_{c^\star}, N, W_{c^\star}, B^{min}, B^{max}, B_{c^\star}, t_{c^\star})$
21: **while** (in parallel with line 20) $i^\star < f'_{c^\star}$ **do**
22: transmit frame $i^\star$ using $r_{i^\star x(i^\star)}$
23: $i^\star \leftarrow i^\star + 1$
24: **end while**
25: **end if**
26: **until** $i^\star > N$

IV. Experimental results

A. Coding performance

FAST-TVC is assessed in terms of coding performance on five different sequences. Each frame of each sequence is compressed with 24 quality layers obtained by using the same 24 distortion-rate slope thresholds for each frame. Coding parameters are: 5 levels of $9/7$ wavelet transform, with codeblocks of size 64×64. Table I describes the characteristics of the five video sequences employed in the experiments, as well as the transmitted range of frames, the rendering pace, and the transmission intervals. We first focus on the transmission of 2000 frames[2] of the "StEM" sequence over a channel that changes its capacity twice. The purpose of this first experiment is to appraise the coding performance of FAST-TVC compared to other strategies that obtain optimal performance, namely, the Viterbi algorithm [7], and the Lagrange method [17]. As described in Section II, the Viterbi algorithm is not practical since it requires enormous computational resources. Nonetheless, it provides optimal performance and provides a good reference to assess the performance of FAST-TVC. The Lagrange method is also impractical since it does not consider the restriction on the buffer size (Expressions (3), (8), and (14)). But, it yields the maximum performance that could be achieved if there were *no* buffer limits. The performance of the CBR strategy is also reported for comparison purposes.

To provide an upper bound on the performance that can be obtained, in this first experiment, the execution time required by the algorithms is not considered (i.e., t_c is set to 0). The

[2]This experiment uses only 2000 frames to allow the use of our Viterbi implementation. The execution time and memory requirements of longer sequences exceed the resources of our servers.

TABLE I: Characteristics of the video sequences employed in the experiments, and conditions of the channel in each transmission interval. For simplicity, only the luminance component is employed (images are 8-bit, gray scale).

sequence	frame size	subsequence rendering pace	transmission intervals $\mathcal{T}_c$ in seconds, W_c in Mbps						total
"StEM"	2048×857	[1150, 3149] 10 fps	$\mathcal{T}_1 = 0$ $W_1 = 3.52$		$\mathcal{T}_2 = 65$ $W_2 = 4.8$		$\mathcal{T}_3 = 130$ $W_3 = 4$		200 secs. 102.6 MB
"Batman"	590×325	[1, 24000] 10 fps	$\mathcal{T}_1 = 0$ $W_1 = 3.04$	$\mathcal{T}_2 = 300$ $W_2 = 2.8$	$\mathcal{T}_3 = 1150$ $W_3 = 2.48$	$\mathcal{T}_4 = 1400$ $W_4 = 2.64$	$\mathcal{T}_5 = 1950$ $W_5 = 2.72$	$\mathcal{T}_6 = 2150$ $W_6 = 3.12$	2400 secs. 836 MB
"Willow"	720×432	[290, 40290] 10 fps	$\mathcal{T}_1 = 0$ $W_1 = 0.8$	$\mathcal{T}_2 = 500$ $W_2 = 0.72$	$\mathcal{T}_3 = 1000$ $W_3 = 0.68$	$\mathcal{T}_4 = 2000$ $W_4 = 0.64$	$\mathcal{T}_5 = 2500$ $W_5 = 0.76$	$\mathcal{T}_6 = 3000$ $W_6 = 0.84$	4000 secs. 372.5 MB
"Giants of Africa"	720×432	[1, 53580]	$\mathcal{T}_c = \{0, 100, 200, 400, 550, 650, 750, 1000, 1150, 1300, 1500, 1850, 1900, 2000, 2200, 2400, 2750, 2850, 3000, 3200, 3500, 3750, 3900, 4050, 4200, 4500, 4650, 4800, 4950, 5200\}$						5357.9 secs.
		10 fps	$W_c = \{0.5, 0.51, 0.53, 0.54, 0.53, 0.55, 0.56, 0.58, 0.54, 0.53, 0.49, 0.51, 0.5, 0.49, 0.48, 0.55, 0.58, 0.56, 0.55, 0.54, 0.53, 0.49, 0.46, 0.48, 0.5, 0.51, 0.5, 0.49, 0.48, 0.5\}$						221.02 MB
"Toy Story"	720×432	[1, 113168]	$\mathcal{T}_c = \{0, 400, 500, 1000, 1500, 1700, 2200, 3000, 3300, 3700, 3900, 4500, 4800, 5200, 5700, 6000, 6200, 6500, 6800, 7300, 7700, 8200, 8500, 8850, 9200, 9450, 9700, 10100, 10750, 10950\}$						11316.7 secs.
		10 fps	$W_c = \{0.63, 0.65, 0.69, 0.68, 0.7, 0.69, 0.66, 0.64, 0.63, 0.6, 0.61, 0.6, 0.59, 0.58, 0.6, 0.56, 0.59, 0.55, 0.54, 0.58, 0.6, 0.61, 0.64, 0.66, 0.69, 0.68, 0.64, 0.63, 0.6, 0.59\}$						564 MB

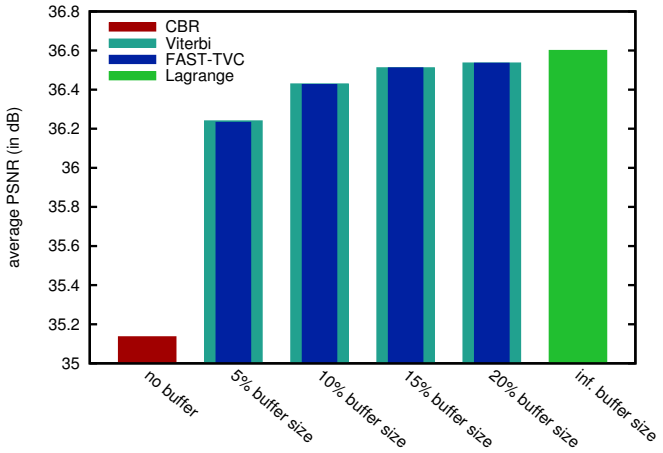


Fig. 4: Average PSNR achieved by FAST-TVC, CBR, Viterbi and the Lagrange method when transmitting 2000 frames of the "StEM" sequence over a time-varying channel. The optimization criterion is MMSE.

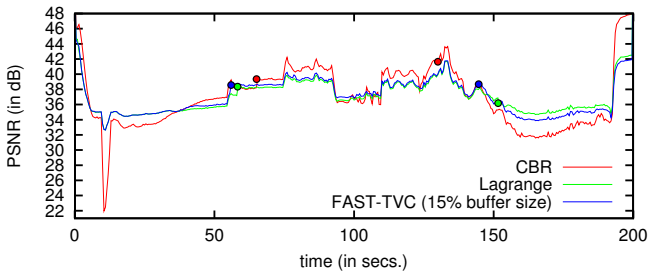


Fig. 5: Frame-by-frame PSNR achieved by FAST-TVC, CBR, and the Lagrange method for the "StEM" sequence. The optimization criterion is MMSE. There are three transmission intervals. For each method, the first frame transmitted after reoptimization (i.e., f_2' and f_3') is marked by a dot.

resulting performance cannot be obtained in practice without essentially infinite computational resources. Figure 4 reports the average MSE of all frames of the sequence for the aforementioned methods when the optimization criterion is MMSE. Viterbi and FAST-TVC obtain virtually the same coding performance regardless of the client buffer size. As expected, the larger the buffer, the closer the performance of Viterbi and FAST-TVC is to that of the Lagrange method. Similar results hold for other video sequences and for the MMAX criterion. These experiments suggest that, under these circumstances, FAST-TVC achieves near-optimal performance.

Figure 5 reports, for the same conditions as above and a buffer size of 15%, the PSNR achieved for each frame. The first frame transmitted in the second and third transmission intervals (i.e., f_2' and f_3') is marked with a dot in this figure. The quality of frames within each transmission interval can be seen to depend on its corresponding channel capacity. It is worth noting that, for the buffer size shown, frames transmitted with FAST-TVC have quality very similar to those transmitted with the Lagrange method. Contrarily, the quality of the simple CBR strategy often varies significantly from that of the Lagrange strategy.

As mentioned above, the first experiment reports results when algorithm execution time is ignored (i.e., $t_c = 0$). The aim of the next experiment is to assess the coding performance of FAST-TVC in a more realistic scenario. This test transmits 24000, 40290, and 53580 frames, respectively, from the "Batman", "Willow", and "Giants of Africa" video sequences. The channel changes capacity 5, 5, and 29 times after the start of transmission, respectively for "Batman", "Willow", and "Giants of Africa" (see Table I for more details). The three strategies described in Section III to determine t_c, namely "estimated t_c," "constant t_c" and "weighted t_c," are put into practice in this experiment. Additionally, performance for the CBR and Lagrange methods are also reported for comparison

purposes.

The results of this experiment are reported in Table II for both the MMSE and MMAX criteria. The buffer sizes chosen for MMAX are generally larger than those for MMSE since MMAX commonly requires more buffer space to provide better pseudo-constant quality [17]. The corresponding values of T and t_c are reported in Table IV. For the "constant t_c" strategy, t_c is chosen so that the total time spent by the algorithm is lower than that spent by the other two strategies. This choice of t_c illustrates the degradation on performance that the "constant t_c" strategy might produce when small values for t_c are used. The fixed part of the "weighted t_c" strategy is chosen to be larger than the variable part (i.e., $t''_c > P \cdot t'_c$) to let this strategy achieve near-optimal performance. Evidently, when $t''_c < P \cdot t'_c$, the "weighted t_c" strategy becomes equivalent to the "constant t_c" strategy. Results for MMSE are reported as the average MSE achieved for all frames of the sequence, while results for MMAX are reported via the MSE standard deviation of frames. Table II presents the results for both criteria. For completeness, results achieved for transmission of the "StEM" sequence are also given in this table.

The results reported in Table II suggest that the three strategies proposed to control the time spent by FAST-TVC achieve significantly better results than the CBR strategy. The "estimated t_c" strategy achieves the best results, and "weighted t_c" is only 2% worse than "estimated t_c," on average. Results indicate that the larger the buffer size, the lower the average MSE, or MSE standard deviation achieved for the MMSE and MMAX criteria, respectively. This suggests that the use of FAST-TVC (either "weighted t_c" or "estimated t_c") with large enough buffers would achieve virtually the same performance as that achieved by the Lagrange method. On the other hand, the "constant t_c" strategy leads to lower performance improvements. This is because t_c is not selected considering the characteristics of the video sequence, the number of frames to be optimized or the channel conditions, which may give too little time to the allocation algorithm to optimize the sequence.

The third test transmits the "Toy Story" video sequence over a channel that changes capacity 29 times. This test employs three buffer sizes and the MMSE criterion. The last column of Table III reports the achieved results, in terms of average MSE. These results correspond with previous experiments, suggesting that the "weighted t_c" strategy achieves virtually same performance as that of the "estimated t_c" strategy, while the larger the buffer size the closer the solution to the Lagrange method. Figures 6(a) and 6(b) report, respectively, the PSNR and the transmitted rate achieved by FAST-TVC "weighted t_c" and the CBR policy for the same conditions as before with buffer sizes 0.5% and 1%. The PSNR achieved by CBR is irregular, producing quick quality changes among consecutive frames. The use of a buffer and FAST-TVC obtains more regular PSNR. The larger the buffer size, the fewer abrupt quality changes. The Lagrange method (not depicted in the figure to avoid cluttering) achieves only a slightly more regular PSNR than FAST-TVC with buffer size 1%. The achievement of regular quality comes at the expense of more variable transmitted rate. Note in Figure 6(b) that the strategy with the largest variations on the transmitted frame rate is FAST-TVC with buffer size 1%.

B. Computational load

The proposed FAST-TVC algorithm is implemented in Java and executed on a Java Virtual Machine v1.6 using GNU/Linux v2.6. The server is an Intel Xeon E5520 CPU at 2.3 GHz. Time results are reported as CPU processing time, in seconds. Table IV reports the execution time spent by the three strategies of FAST-TVC, for transmission of the video sequences "StEM", "Batman", "Willow", and "Giants of Africa" under the same conditions as described above. Table III reports results for "Toy Story". The first column for each strategy reports the time spent when the client request is received and all frames of the sequence are optimized, i.e., T. The following columns report the execution time spent by the algorithm when a variation on the channel occurs (i.e., t_c). The last column reports the sum of all execution times excepting T. Recall that the percentage of time given to the "weighted t_c" strategy is 60% and 80% for the MMSE and MMAX strategies, respectively.

Experimental results suggest that, even though the "weighted t_c" strategy spends 40% and 20% less time than the "estimated t_c" strategy, respectively for MMSE and MMAX, its coding performance is almost unaffected compared to "estimated t_c." As stated previously, these savings on computational load are achieved due to the fast convergence of the rate allocation algorithm.

Figure 7 depicts the computational time spent by the three strategies of FAST-TVC when transmitting "Toy Story" with the same buffer sizes and channel conditions as used before. Note that the larger the buffer, the more time required by the strategies "weighted t_c" and "estimated t_c". The "constant t_c" strategy spends the same computational time regardless of the buffer size. It is worth noting that, under these conditions, "constant t_c" spends more time on average than "weighted t_c" when the buffer size is 0.5% although the solution achieved by "weighted t_c" is better than that of "constant t_c" (see Table III). This is because "weighted t_c" spends a variable amount of time depending on the number of frames to be optimized. In particular, less computational time is used by "weighted t_c" for capacity variations that occur near the end of the sequence due to the smaller number of frames to be considered. This indicates that distribution of the computational time carried out by "weighted t_c" is adequately balanced considering the conditions of the channel and video sequence at the instant the channel variation occurs.

V. Conclusions

Rate allocation is of paramount importance in video transmission schemes to optimize video quality. Applications that transmit video over local area networks, Internet, or dedicated networks, may experience variations on channel conditions due to network saturation, TCP congestion, or router failures. This work proposes a rate allocation algorithm for the transmission of JPEG2000 video named FAST for time-varying channels (FAST-TVC). The proposed method is built on our previous

TABLE II: Coding performance evaluation for FAST-TVC, CBR, and the Lagrange method. Three different strategies to compute t_c are employed by FAST-TVC.

			"StEM"		"Batman"		"Willow"		"Giants of Africa"		*average*
		buffer size:	5%	7%	1%	2%	4%	5%	5%	6%	*4.38%*
MMSE av. MSE	CBR		19.42		3.32		8.20		35.75		*16.67*
	FAST-TVC	"constant t_c"	16.60	15.40	3.19	3.17	6.98	6.88	31.91	31.74	*14.48*
		"weighted t_c"	15.55	14.94	3.10	3.07	6.80	6.60	27.42	27.35	*13.10*
		"estimated t_c"	15.26	14.89	3.09	3.07	6.78	6.60	27.42	27.33	*13.06*
	Lagrange		13.88		3.01		6.29		26.97		*12.54*

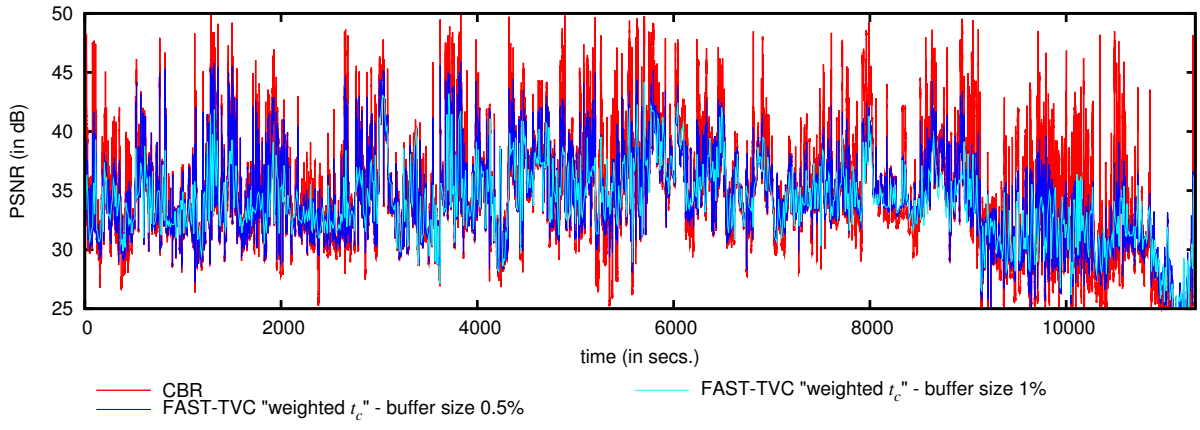
			buffer size:	15%	20%	5%	7%	12%	15%	16%	17%	*13.38%*
MMAX MSE st. dev.	CBR			21.92		1.50		7.61		33.27		*16.08*
	FAST-TVC	"constant t_c"		6.43	1.02	0.84	0.77	8.66	5.68	27.95	26.25	*9.7*
		"weighted t_c"		3.09	0.99	0.57	0.47	3.83	3.16	5.99	5.04	*2.89*
		"estimated t_c"		3.08	0.99	0.57	0.47	3.83	3.11	5.97	5.03	*2.88*
	Lagrange			0		0		0		0		*0*

TABLE III: Coding performance and computational time evaluation for optimizing MMSE with three different buffer sizes. Results are reported as average MSE and seconds for "Toy Story" transmitted over a channel that changes capacity 30 times.

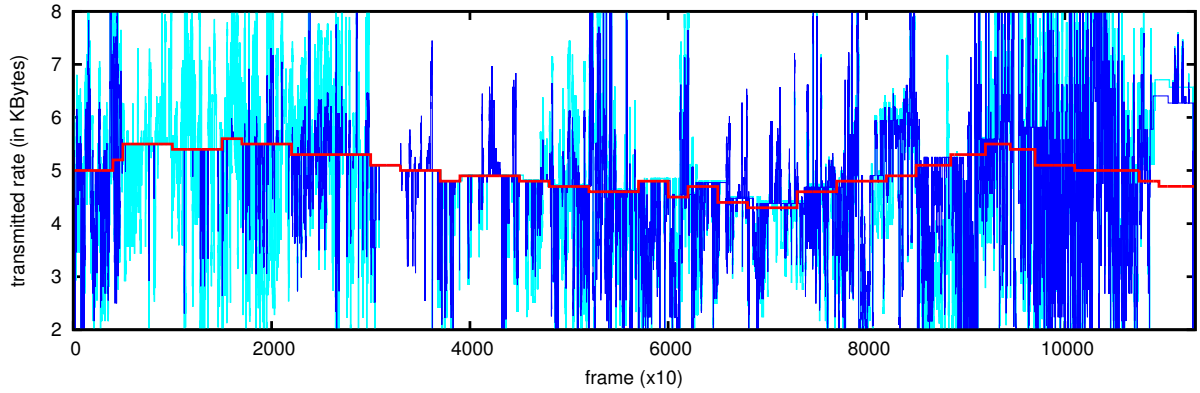
buffer size	policy		T	t_c	$\sum t_c$	av. MSE
0.5%	CBR		0	0	*0*	40.8
	FAST-TVC	"constant t_c"	5.6	2.5	*72.5*	36.9
		"weighted t_c"	5.6	3.2, 3.2, 3.1, 3.0, 2.9, 2.8, 2.6, 2.5, 2.4, 2.3 2.1, 2.0, 1.9, 1.7, 1.6, 1.6, 1.5, 1.4, 1.2, 1.1 0.9, 0.8, 0.7, 0.6, 0.5, 0.5, 0.3, 0.1, 0.1	*48.6*	35.9
		"estimated t_c"	5.9	5.7, 5.7, 5.4, 5.4, 5.2, 4.9, 4.6, 4.3, 4.2, 4.1 3.7, 3.6, 3.3, 3.1, 2.9, 2.8, 2.6, 2.4, 2.1, 1.9 1.6, 1.5, 1.2, 1.1, 0.9, 0.8, 0.6, 0.3, 0.2	*86.1*	35.2
	Lagrange		0	0	*0*	31.0
0.8%	CBR		0	0	*0*	40.7
	FAST-TVC	"constant t_c"	10.1	2.5	*72.5*	35.0
		"weighted t_c"	10.2	5.9, 5.9, 5.6, 5.5, 5.4, 5.1, 4.7, 4.5, 4.3, 4.2 3.8, 3.7, 3.4, 3.2, 2.9, 2.8, 2.7, 2.5, 2.1, 1.9 1.7, 1.5, 1.2, 1.0, 0.9, 0.8, 0.6, 0.2, 0.1	*88.1*	34.2
		"estimated t_c"	10.2	9.9, 9.7, 9.3, 9.2, 9.1, 8.7, 7.9, 7.6, 7.2, 7.0 6.4, 6.1, 5.8, 5.3, 4.9, 4.7, 4.4, 4.1, 3.7, 3.3 2.8, 2.5, 2.0, 1.7, 1.5, 1.2, 0.9, 0.3, 0.2	*147.4*	33.8
	Lagrange		0	0	*0*	31.0
1%	CBR		0	0	*0*	40.8
	FAST-TVC	"constant t_c"	17.8	2.5	*72.5*	34.6
		"weighted t_c"	18.1	10.5, 10.4, 9.9, 9.8, 9.6, 9.1, 8.3, 7.9, 7.6, 7.4 6.8, 6.5, 6.0, 5.5, 5.1, 4.9, 4.6, 4.2, 3.8, 3.4 2.9, 2.5, 2.1, 1.8, 1.5, 1.3, 0.9, 0.3, 0.2	*154.8*	34.2
		"estimated t_c"	17.8	17.1, 16.9, 16.2, 16.0, 15.8, 14.9, 13.7, 13.1, 12.5, 12.1, 11.1, 10.6, 9.9, 9.1, 8.5, 8.2, 7.7, 7.2, 6.3 5.7, 4.8, 4.2, 3.5, 2.9, 2.6, 2.2, 1.6, 0.7, 0.4	*255.5*	33.1
	Lagrange		0	0	*0*	31.0

FAst rate allocation through STeepest descent (FAST) algorithm, extending and exploiting some of its features. The main insight behind FAST-TVC is to employ complexity scalability and the roughly linear relation between computational load and number of frames to re-compute frame rates once a variation on the channel capacity takes place.

Experimental results indicate that FAST-TVC achieves virtually the same coding performance as that of the optimal Viterbi algorithm (when the Viterbi algorithm is computationally feasible). When the server needs to control the resources dedicated to the rate allocation algorithm depending on system load or other indicators, FAST-TVC can use one of three

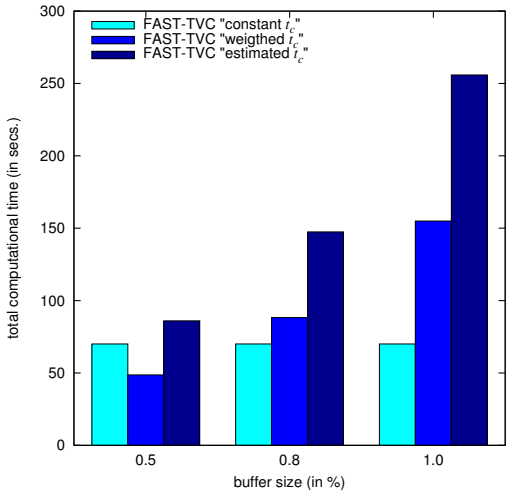


(a)



(b)

Fig. 6: (a) Frame-by-frame PSNR and, (b) transmitted rate achieved by FAST-TVC and CBR method for the "Toy Story" sequence. The optimization criterion is MMSE.



(a)

Fig. 7: Computational time spent by the three strategies of FAST-TVC in response to channel capacity variations (i.e., $\sum_c t_c$) when optimizing the "Toy Story" sequence for MMSE with three different buffer sizes.

proposed strategies. The first strategy is named "constant t_c" and provides a constant execution time to the algorithm. Although this strategy achieves a non-negligible gain in coding performance with respect to a constant-rate strategy, results vary significantly depending on the video sequence, buffer size, and channel conditions. The second strategy is named "estimated t_c" referring to its ability to estimate the total time that FAST-TVC requires to finish its execution. This allows FAST-TVC to achieve more consistent results, but does not supply any mechanism to reduce computational time when the server is busy. The "weighted t_c" strategy is a compromise between the previous two: it achieves virtually same results as "estimated t_c," and reduces computational load significantly. Experimental results evaluating the computational costs of FAST-TVC indicate that very few computational resources are expended. These characteristics makes FAST-TVC a suitable method for the transmission of pre-encoded JPEG2000 video in real-world applications.

REFERENCES

[1] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with H.264/AVC: Tools, performance, and complexity," *IEEE Circuits Syst. Mag.*, vol. 4, no. 1, pp. 7–28, Jan. 2004.

[2] D. S. Taubman and M. W. Marcellin, *JPEG2000 Image compression fundamentals, standards and practice*. Norwell, Massachusetts 02061 USA: Kluwer Academic Publishers, 2002.

[3] C.-Y. Hsu, A. Ortega, and A. R. Reibman, "Joint selection of source and channel rate for VBR video transmission under ATM policing constraints," *IEEE J. Sel. Areas Commun.*, vol. 15, no. 6, pp. 1016–1028, Aug. 1997.

TABLE IV: Computational time evaluation for FAST-TVC. The first column of each criteria reports the computational time (in seconds) spent to optimize all frames of the sequence (i.e., T). The following columns report t_c. The last column reports the total time spent by the algorithm to re-compute frame rates in response to channel capacity variations (i.e., $\sum_c t_c$).

		MMSE						*total*
"StEM" (buffer 5%)	"constant t_c"	2.089	0.075					*0.150*
	"weighted t_c"	2.088	0.881	0.406				*1.287*
	"estimated t_c"	2.220	1.562	0.722				*2.284*
"StEM" (buffer 7%)	"constant t_c"	3.502	0.075					*0.150*
	"weighted t_c"	3.478	1.599	0.719				*2.318*
	"estimated t_c"	3.664	2.665	1.185				*3.850*
"Batman" (buffer 1%)	"constant t_c"	2.542	0.250					*1.250*
	"weighted t_c"	2.527	1.330	0.794	0.639	0.283	0.160	*3.206*
	"estimated t_c"	2.545	2.232	1.334	1.072	0.474	0.270	*5.382*
"Batman" (buffer 2%)	"constant t_c"	4.759	0.250					*1.250*
	"weighted t_c"	4.773	2.516	1.506	1.213	0.529	0.308	*6.072*
	"estimated t_c"	4.753	4.175	2.499	2.013	0.878	0.511	*10.076*
"Willow" (buffer 4%)	"constant t_c"	4.691	0.500					*2.500*
	"weighted t_c"	4.711	2.053	1.528	1.022	0.729	0.324	*5.656*
	"estimated t_c"	4.721	3.430	2.552	1.704	1.217	0.539	*9.442*
"Willow" (buffer 5%)	"constant t_c"	8.062	0.500					*2.500*
	"weighted t_c"	8.096	3.522	2.618	1.729	1.288	0.597	*9.754*
	"estimated t_c"	8.088	5.872	4.355	2.882	2.110	0.996	*16.215*
"Giants of Africa" (buffer 5%)	"constant t_c"	127.835	1.750					*50.75*
	"weighted t_c"	127.862	74.949, 73.616, 70.884, 71.393, 70.011, 68.576, 64.879, 63.312, 61.365, 59.019, 54.122, 53.389, 51.763, 47.910, 44.402, 39.505, 37.993, 36.107, 33.166, 27.793, 24.870, 22.772, 20.524, 18.365, 11.299, 9.201, 5.088, 2.046					*1218.319*
	"estimated t_c"	127.863	124.916, 122.694, 118.141, 118.990, 116.687, 114.294, 108.132, 105.520, 102.275, 98.366, 90.203, 88.982, 86.272, 79.853, 74.006, 65.841, 63.322, 60.174, 55.277, 46.322, 41.450, 37.953, 34.208, 30.609, 18.831, 15.334, 8.480, 3.409					*2030.541*
"Giants of Africa" (buffer 6%)	"constant t_c"	130.647	1.750					*50.75*
	"weighted t_c"	130.664	76.414, 75.065, 72.274, 72.799, 71.385, 69.939, 66.153, 64.582, 62.582, 60.202, 55.292, 54.636, 53.036, 49.001, 45.319, 40.401, 38.794, 36.863, 33.906, 28.349, 25.363, 23.238, 20.931, 18.718, 11.501, 9.374, 6.487, 5.182, 2.084					*1249.87*
	"estimated t_c"	130.681	127.428, 125.179, 120.524, 121.400, 119.042, 116.629, 110.317, 107.697, 104.346, 100.372, 92.141, 91.058, 88.357, 81.654, 75.544, 67.156, 64.554, 61.292, 56.136, 47.109, 42.052, 38.478, 34.506, 30.887, 18.720, 15.380, 10.663, 8.517, 3.421					*2080.559*

		MMAX						*total*
"StEM" (buffer 15%)	"constant t_c"	3.507	0.050					*0.100*
	"weighted t_c"	3.533	2.094	0.681				*2.775*
	"estimated t_c"	3.512	2.602	0.846				*3.448*
"StEM" (buffer 20%)	"constant t_c"	8.854	0.050					*0.100*
	"weighted t_c"	8.858	4.022	1.218				*5.240*
	"estimated t_c"	8.820	6.704	2.014				*8.718*
"Batman" (buffer 5%)	"constant t_c"	22.087	0.500					*2.500*
	"weighted t_c"	21.928	15.683	9.653	7.931	3.577	1.998	*38.842*
	"estimated t_c"	22.045	19.708	12.131	9.966	4.495	2.510	*48.810*
"Batman" (buffer 7%)	"constant t_c"	15.781	0.500					*2.500*
	"weighted t_c"	15.650	11.123	6.766	5.495	2.486	1.396	*27.266*
	"estimated t_c"	15.778	14.018	8.527	6.925	3.133	1.758	*34.361*
"Willow" (buffer 12%)	"constant t_c"	13.176	1.000					*5.000*
	"weighted t_c"	13.115	7.956	5.645	3.174	2.168	1.079	*20.022*
	"estimated t_c"	13.143	9.967	7.071	3.976	2.714	1.346	*25.074*
"Willow" (buffer 15%)	"constant t_c"	13.693	1.000					*5.000*
	"weighted t_c"	13.677	8.272	5.918	3.540	2.566	1.196	*21.492*
	"estimated t_c"	13.685	10.347	7.401	4.430	3.219	1.506	*26.903*
"Giants of Africa" (buffer 16%)	"constant t_c"	9.529	2.800					*81.2*
	"weighted t_c"	9.521	7.423, 7.271, 7.040, 7.122, 6.989, 6.858, 6.505, 6.370, 6.251, 6.061, 5.663, 5.616, 5.424, 5.114, 4.746, 4.177, 4.059, 3.761, 3.397, 2.796, 2.523, 2.437, 2.243, 1.966, 1.240, 0.953, 0.622, 0.494, 0.173					*125.294*
	"estimated t_c"	9.529	9.287, 9.097, 8.807, 8.910, 8.744, 8.580, 8.138, 7.970, 7.820, 7.582, 7.085, 7.026, 6.785, 6.397, 5.938, 5.225, 5.078, 4.705, 4.250, 3.498, 3.157, 3.047, 2.807, 2.459, 1.552, 1.192, 0.778, 0.617, 0.214					*156.745*
"Giants of Africa" (buffer 17%)	"constant t_c"	10.089	2.800					*81.2*
	"weighted t_c"	10.089	7.870, 7.716, 7.461, 7.548, 7.399, 7.266, 6.913, 6.757, 6.621, 6.436, 6.003, 5.957, 5.759, 5.500, 5.050, 4.427, 4.321, 4.000, 3.628, 2.966, 2.675, 2.595, 2.374, 2.084, 1.323, 1.009, 0.658, 0.480, 0.048					*132.844*
	"estimated t_c"	10.092	9.840, 9.648, 9.329, 9.438, 9.252, 9.085, 8.644, 8.445, 8.279, 8.047, 7.506, 7.448, 7.201, 6.877, 6.315, 5.536, 5.403, 5.002, 4.537, 3.709, 3.345, 3.244, 2.968, 2.606, 1.654, 1.262, 0.823, 0.600, 0.061					*166.104*

[4] C.-Y. Hsu, A. Ortega, and M. Khansari, "Rate control for robust video transmission over burst-error wireless channels," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 5, pp. 756–773, May 1999.

[5] R. Rejaie, M. Handley, and D. Estrin, "Layered quality adaptation for internet video streaming," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 12, pp. 2530–2543, Dec. 2000.

[6] J. C. Dagher, A. Bilgin, and M. W. Marcellin, "Resource-constrained rate control for motion JPEG2000," *IEEE Trans. Image Process.*, vol. 12, no. 12, pp. 1522–1529, Dec. 2003.

[7] A. Ortega, K. Ramchandran, and M. Vetterli, "Optimal trellis-based buffered compression and fast approximations," *IEEE Trans. Image Process.*, vol. 3, no. 1, pp. 26–40, Jan. 1994.

[8] Z. Miao and A. Ortega, "Scalable proxy caching of video under storage constraints," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 7, pp. 1315–1327, Sep. 2002.

[9] D. Jurca, J. Chakareski, J.-P. Wagner, and P. Frossard, "Enabling adaptive video streaming in P2P systems," *IEEE Commun. Mag.*, vol. 45, no. 6, pp. 108–114, Jun. 2007.

[10] E. Setton and B. Girod, *Peer-to-Peer Video Streaming*. New York, NY 10013 USA: Springer, 2007.

[11] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Process. Mag.*, vol. 15, no. 6, pp. 23–50, Nov. 1998.

[12] D. T. Hoang, "Fast and efficient algorithms for text and video compression," Ph.D. dissertation, Brown University, Providence, Rhode Island, May 1997.

[13] T. V. Lakshman, A. Ortega, and A. R. Reibman, "VBR video: Tradeoffs and potentials," *Proc. IEEE*, vol. 86, no. 5, pp. 952–973, May 1998.

[14] G. M. Schuster, G. Melnikov, and A. K. Katsaggelos, "A review of the minimum maximum criterion for optimal bit allocation among dependent quantizers," *IEEE Trans. Multimedia*, vol. 1, no. 1, pp. 3–17, Mar. 1999.

[15] K.-L. Huang and H.-M. Hang, "Consistent picture quality control strategy for dependent video coding," *IEEE Trans. Image Process.*, vol. 18, no. 5, pp. 1004–1014, May 2009.

[16] S.-Y. Lee and A. Ortega, "Optimal rate control for video transmission over VBR channels based on a hybrid MMAX/MMSE criterion," in *Proc. IEEE International Conference on Multimedia and Expo*, vol. 2, Aug. 2002, pp. 93–96.

[17] F. Auli-Llinas, A. Bilgin, and M. W. Marcellin, "FAST rate allocation through steepest descent for JPEG2000 video transmission," *IEEE Trans. Image Process.*, vol. 20, no. 4, pp. 1166–1173, Apr. 2011.

[18] A. R. Reibman and B. G. Haskell, "Constraints on variable bit-rate video for ATM networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, no. 4, pp. 361–372, Dec. 1992.

[19] C.-T. Chen and A. Wong, "A self-governing rate buffer control strategy for pseudoconstant bit rate video coding," *IEEE Trans. Image Process.*, vol. 2, no. 1, pp. 50–59, Jan. 1993.

[20] S.-W. Wu and A. Gersho, "Rate-constrained optimal block-adaptive coding for digital tape recording of HDTV," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 1, no. 1, pp. 100–112, Mar. 1991.

[21] J. Choi and D. Park, "A stable feedback control of the buffer state using the controlled Lagrange multiplier method," *IEEE Trans. Image Process.*, vol. 3, no. 5, pp. 546–558, Sep. 1994.

[22] Z. Miao and A. Ortega, "Optimal scheduling for streaming of scalable media," in *Proc. IEEE Asilomar Conference on Signals, Systems, and Computers*, vol. 2, Nov. 2000, pp. 1357–1362.

[23] Y. Sermadevi and S. S. Hemami, "Efficient bit allocation for dependent video coding," in *Proc. IEEE Data Compression Conference*, Mar. 2004, pp. 232–241.

[24] H. Everett, "Generalized Lagrange multiplier method for solving problems of optimum allocation of resources," *Oper. Res.*, vol. 11, pp. 399–417, 1963.

[25] J. F. Kurose and K. W. Ross, *Computer Networking. A top-down approach.* Addison Wesley, 2008.

Leandro Jiménez-Rodríguez received the B.E. and M.S. degrees in 2008 and 2010, respectively, both in Computer Engineering from the Universitat Autònoma de Barcelona. In 2009 he was awarded with a doctoral fellowship from the Universitat Autònoma de Barcelona that funds his current doctoral studies. His research interests include scalable video coding systems and video transmission.

Francesc Aulí-Llinàs (S'06-M'08) is a Ramón y Cajal Fellow at the Department of Information and Communications Engineering, Universitat Autònoma de Barcelona (Spain). He received the B.Sc. and B.E. degrees in Computer Management Engineering and Computer Engineering in 2000 and 2002, respectively, both from the Universitat Autònoma de Barcelona (Spain), and for which he was granted with two extraordinary awards of Bachelor. In 2004 and 2006 he respectively received the M.S. degree and the Ph.D. degree (with honors), both in Computer Science from the Universitat Autònoma de Barcelona. Since 2002 he has been consecutively awarded with doctoral and postdoctoral fellowships in competitive calls. From 2007 to 2009 he carried out two research stages of one year each with the group of David Taubman, at the University of New South Wales (Australia), and with the group of Michael Marcellin, at the University of Arizona (USA). He is the main developer of BOI, a JPEG2000 Part 1 implementation that was awarded with a free software mention from the Catalan Government in April 2006. His research interests include a wide range of image coding topics, including highly scalable image and video coding systems, rate-distortion optimization, distortion estimation, and interactive transmission, among others.

Michael W. Marcellin (S'81-M'87-SM'93-F'02) graduated *summa cum laude* with the B.S. degree in Electrical Engineering from San Diego State University in 1983, where he was named the most outstanding student in the College of Engineering. He received the M.S. and Ph.D. degrees in Electrical Engineering from Texas A&M University in 1985 and 1987, respectively. Since 1988, Dr. Marcellin has been with the University of Arizona, where he holds the title of Regents' Professor of Electrical and Computer Engineering, and of Optical Sciences. Dr. Marcellin is a major contributor to JPEG2000, the emerging second-generation standard for image compression. He is coauthor of the book, *JPEG2000: Image compression fundamentals, standards and practice,* Kluwer Academic Publishers, 2002. Dr. Professor Marcellin is a Fellow of the IEEE, and is a member of Tau Beta Pi, Eta Kappa Nu, and Phi Kappa Phi. He is a 1992 recipient of the National Science Foundation Young Investigator Award, and a corecipient of the 1993 IEEE Signal Processing Society Senior (Best Paper) Award. He has received teaching awards from NTU (1990, 2001), IEEE/Eta Kappa Nu student sections (1997), and the University of Arizona College of Engineering (2000). In 2003, he was named the San Diego State University Distinguished Engineering Alumnus. Professor Marcellin is the recipient of the 2006 University of Arizona Technology Innovation Award. From 2001 to 2006, Dr. Marcellin was the Litton Industries John M. Leonis Professor of Engineering. He is currently the International Foundation for Telemetering Professor of Electrical and Computer Engineering at the University of Arizona.