Rate-Distortion Optimized Streaming of Fine-Grained Scalable Video Sequences

MOHAMED HEFEEDA and CHENG-HSIN HSU
Simon Fraser University, Canada

We present optimal schemes for allocating bits of fine-grained scalable video sequences among multiple senders streaming to a single receiver. This allocation problem is critical in optimizing the perceived quality in peer-to-peer and distributed multi-server streaming environments. Senders in such environments are heterogeneous in their outgoing bandwidth and they hold different portions of the video stream. We first formulate and optimally solve the problem for individual frames, then we generalize to the multiple frame case. Specifically, we formulate the allocation problem as an optimization problem, which is nonlinear in general. We use rate-distortion models in the formulation to achieve the minimum distortion in the rendered video, constrained by the outgoing bandwidth of senders, availability of video data at senders, and incoming bandwidth of receiver. We show how the adopted rate-distortion models transform the nonlinear problem to an integer linear programming (ILP) problem. We then design a simple rounding scheme that transforms the ILP problem to a linear programming (LP) one, which can be solved efficiently using common optimization techniques such as the Simplex method. We prove that our rounding scheme always produces a feasible solution, and the solution is within a negligible margin from the optimal solution. We also propose a new algorithm (FGSAssign) for the single-frame allocation problem that runs in $O(n \log n)$ steps, where n is the number of senders. We prove that FGSAssign is optimal. Furthermore, we propose a heuristic algorithm (mFGSAssign) that produces near-optimal solutions for the multiple-frame case, and runs an order of magnitude faster than the optimal one. Because of its short running time, mFGSAssign can be used in real time. Our experimental study validates our analytical analysis and shows the effectiveness of our allocation algorithms in improving the video quality.

Categories and Subject Descriptors: H.1.1 [**Models and Principles**]: Systems and Information Theory

General Terms: Performance

Additional Key Words and Phrases: Fine-grained scalable streaming, FGS, rate-distortion optimized streaming, video streaming, peer-to-peer streaming, distributed streaming, rate-distortion models

ACM Reference Format:

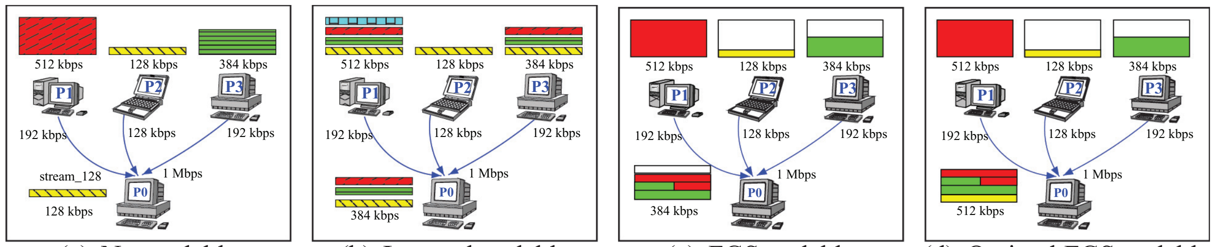
Hefeeda, M. and Hsu, C.-H. 2008. Rate-distortion optimized streaming of fine-grained scalable video sequences. ACM Trans. Multimedia Comput. Commun. Appl. 4, 1, Article 2 (January 2008), 28 pages. DOI = 10.1145/1324287.1324289 http://doi.acm.org/10.1145.1324287.1324289

1. INTRODUCTION

Video streaming over the Internet is increasingly getting very popular. In this article, we consider video streaming systems in which a streaming session has multiple senders and a single receiver. Multiple

This work was partially supported by the Natural Science and Engineering Research Council (NSERC) of Canada under Discovery Grant #313083 and RTI Grant #344619.

Authors' address: Simon Fraser University, 250-13450 102nd Ave, Surrey BC V3T AO3, Canada; email: {mhefeeda, cha16}@cs.sfu.ca.



(a) Nonscalable (b) Layered scalable (c) FGS scalable (d) Optimal FGS scalable

Fig. 1. A multi-sender streaming session using different allocation schemes.

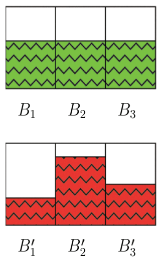
senders may be required in peer-to-peer streaming environments [Hefeeda et al. 2005; Magharei and Rejaie 2006; Cui and Nahrstedt 2003], because of the limited capacity and unreliability of peers. Multiple senders are also desired in distributed streaming systems [Nguyen and Zakhor 2002] to achieve disjoint network path streaming and hence better quality. We consider the general case when senders have heterogeneous outgoing bandwidth, and may store different portions of the requested stream. Our problem is to optimally allocate to each potential sender a transmission rate and range of bits to transmit such that the best video quality is achieved at the receiver.

We illustrate the importance of the bit allocation problem using the simple example shown in Figure 1. There are three senders P1, P2, and P3, and one receiver P0. P1, P2, and P3 have outgoing bandwidth of 192, 128, and 192 Kbps, respectively, and P0 has incoming bandwidth of 1 Mbps. Furthermore, senders are assumed to have downloaded different portions of the stream in the past. The size of the stored stream at each sender is denoted by the bit rate used in the download.

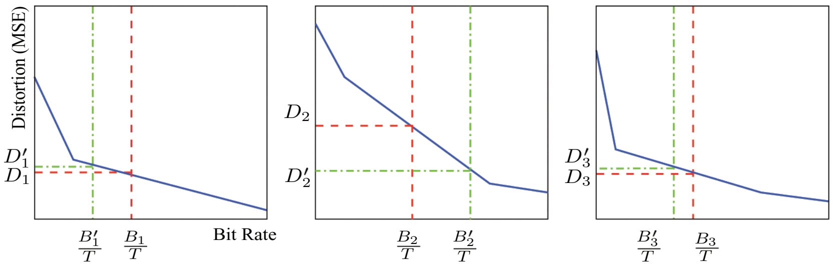
Figure 1 compares various possible solutions for the allocation problem, and the quality of the received stream in each case. In Figure 1(a), a nonscalable allocation scheme is considered, which assumes that the video stream is encoded using a nonscalable encoder. In nonscalable encoding, the stream has to be downloaded in its entirety, otherwise it is not decodable. In this case, senders are considered to have different *versions* of the same stream. Therefore, under the nonscalable allocation scheme, the receiver can only get the 128-Kbps stream version. Notice that collaboration among senders is not possible, because the versions they have are encoded differently. Nonscalable encoding is actually not uncommon in current Internet streaming systems: Many streaming servers post the same video clip in different rates to accommodate users with heterogeneous bandwidth.

To cope with the inflexibility of nonscalable encoding, some streaming systems encode a stream into multiple layers. Typically, there are a few number of layers, because of the layering overhead and the complexity of the coding/decoding processes. In layered scalable coding, partial layers are not decodable. Figure 1(b) shows a layered scalable allocation scheme, where the stream is encoded into four layers, each has a bit rate of 128 Kbps. In this case, the receiver can get up to three layers, one from each sender. Thus the received stream will be of rate 384 Kbps. Due to the coarse-grain nature of the layers, this allocation scheme cannot leverage the remaining 64-Kbps bandwidth at each of P1 and P3.

The fine-granularity scalability (FGS) encoding adds significant flexibility to the layered encoding: FGS-encoded streams provide bit-level scalability, which means that bit streams can be truncated at any bit location. In addition, FGS-encoded streams have the property that a low-quality stream is always a prefix of a higher-quality one. The FGS flexibility, however, complicates the problem of allocating bits to senders, because of the finer resolution and the too many allocation possibilities that should be considered to select the optimal allocation. For instance, a possible FGS allocation is shown in Figure 1(c), where the received stream has a bit rate of 384 Kbps. This scheme started by allocating the first 192 Kbps of the stream to P3. Thus it cannot use P2, because the stream at P2 is a prefix of



(a) Bits allocated to each frame in a block



(b) Rate-distortion curves for three consecutive frames in a block

Fig. 2. The importance of solving the bit allocation problem at the block level.

what would be transmitted by P3. The only option left for this allocation scheme is to allocate the second 192 Kbps of the stream to P1. A more careful—actually optimal—allocation is shown in Figure 1(d), where the receiver gets a much better quality stream of 512 Kbps bit rate.

To summarize, this example indicates that significant quality improvement could be achieved by adopting the fine-grained scalable encoding in streaming systems with multiple heterogeneous senders. It also highlights the importance of the optimal allocation of bits among senders.

The bit allocation problem described above can be solved on a frame-by-frame basis: For every frame, we optimally allocate the bits of that frame to senders. The optimal allocation ensures that the maximum number of bits in every frame is transmitted from senders. More bits of the same frame yield less reconstruction distortion, and hence better playout quality. The bit allocation problem could also be solved for a block of multiple frames at once. The rationale is that we may have an opportunity to further enhance quality by considering the relative importance of bits in frames belonging to the same block. Using rate-distortion curves is the standard method in the literature to quantify the relative contribution of bits towards the video reconstruction quality.

To illustrate the multiple-frame bit allocation problem, consider a block of three frames, as shown in Figure 2(a). Assume that the optimal solution of the single-frame allocation problem determined that senders can transmit up to $B_1 = B_2 = B_3$ bits for frames 1, 2, and 3. Further, suppose that the frames have the rate-distortion (R-D) curves shown in Figure 2(b). R-D curves map a given bit rate to the corresponding distortion level (detailed description of R-D curves is given in Section 3.2). Notice that the R-D curves may differ from one frame to another, because they depend on the temporal and spatial complexities of the frame. Figure 2(b) indicates that transmitting B_1, B_2, B_3 bits results in distortion levels D_1, D_2, D_3 for frames 1, 2, and 3, respectively. Notice that D_2 is much larger than D_1 and D_3, which implies large fluctuation in the playout quality of the consecutive frames. Quality fluctuations have a negative impact on the user-perceived quality [Zink et al. 2003]. Now, consider the same bit budget for the three-frame block, but distributed differently among the frames, as shown in the lower part of Figure 2(a). This slight change in bit distribution results in quality improvement in two ways: (i) smaller total distortion in the block; and, more importantly, (ii) much smaller quality fluctuation in successive frames. This is shown in Figure 2(b), where the new distortion levels are denoted by D_1', D_2', D_3'. Notice that the decrease in bit rates allocated to frames 1 and 2 result in a minor increase in their distortion levels, while the distortion of frame 2 decreases substantially due to the extra bit rate allocated to it.

In summary, solving the bit allocation problem at the block level optimizes the playout quality by getting the maximum number of bits from senders, and by carefully distributing these bits among frames using R-D curves. Notice that we only need R-D curves to solve the bit allocation problem at the block level to quantify the relative importance of bits belonging to different frames. R-D curves are not needed in solving the problem at the frame level, because more bits of the same frame always yield better quality.

1.1 Contributions

In this article, we formulate and optimally solve the single-frame and multiple-frame bit allocation problems, and we experimentally show the effectiveness of our solutions in improving the video playout quality. In particular, we make the following contributions.

—We formulate the single-frame allocation problem as an optimization problem with an objective function to minimize the distortion. By using the piece-wise linear rate-distortion model, we transform the general (nonlinear) optimization problem to an integer linear programming (ILP) problem. We design a simple rounding scheme that transforms the ILP problem to a linear programming (LP) one, which could be solved efficiently using common optimization techniques such as the Simplex method. We prove that our rounding scheme always produces a feasible solution, and the solution is within a negligible margin from the optimal one.

—We propose a new algorithm (FGSAssign) for the single-frame allocation problem that runs in $O(n \log n)$ steps, where n is the number of senders. We prove that FGSAssign is optimal.

—We generalize the LP formulation to the multiple-frame allocation problem and we solve it using the Simplex method.

—We propose a heuristic algorithm (mFGSAssign) that produces near-optimal solutions for the multiple-frame case, and runs an order of magnitude faster than the optimal one. Because of its short running time, the algorithm can be used in real time during the streaming session.

—As a side contribution, we rigorously analyze the accuracy of the piece-wise linear rate-distortion model, and we show how its parameters can be efficiently extracted, stored and used in streaming applications that seek to optimize distortion at different bit rates. This analysis is useful in its own right, and it was not done before in the literature.

1.2 Applications of Our Work

Our work is applicable to streaming systems that serve fine-grained scalable video sequences in rate-distortion optimized manner. That is, systems that minimize the distortion in the rendered video under various bit rates. We describe below three classes of systems that can benefit from our work.

—*Peer-to-Peer Streaming.* In this case, multiple peers collaborate in serving a video stream to a receiver. Multiple peers are required because of the limited bandwidth contribution from each peer. The receiver uses our allocation algorithm (mFGSAssign) to assign bit rates and data to senders. To run the allocation algorithm, the receiver requires the rate-distortion model, which is computed offline and stored in a meta file. The receiver gets this meta file from any sender in the beginning of the session. As described in Section 3, the size of the meta file is negligible compared to the size of the video file. The receiver sends assignments to senders in small control packets. Each assignment is computed for a block of frames, for example, 60 frames. A control packet contains assignments for a number of contiguous blocks, for example, 30 blocks. Assuming a frame rate of 30 fps and using the above example values, the receiver sends a single control packet every one minute to each sender, which is a very small overhead. Notice also that this overhead is imposed on the reverse channel (from receiver to sender), which is usually under utilized during streaming sessions.

In addition, as we show in Section 6, the running time of our algorithm is in the order of msec on a commodity PC. This means that the algorithm can easily be run in real time during the streaming session. Furthermore, the algorithm can handle dynamic conditions such as bandwidth changes and sender failures. In such conditions, a re-assignment is quickly computed and communicated to senders in exceptional control packets.

—*Multi-Server Distributed Streaming.* Streaming servers are sometimes replicated at different locations in the Internet. A client may receive different portions of the stream from two or more servers. This achieves network path diversity and has shown to enhance quality [Nguyen and Zakhor 2002]. Unlike P2P streaming, in which senders have limited streaming capacity and are the bottleneck, the receiver bandwidth in this case is usually the bottleneck. To maximize the use of the receiver's bandwidth, the available bandwidth values along network paths from different servers to receiver are estimated using, for example, techniques in Hefeeda et al. [2005]. Then using our allocation algorithm, and the available bandwidth to receiver, each server is assigned the optimal sending rate and the range of bits that minimize distortion.

—*Client-Server Streaming.* Our algorithms are also useful in the common client-server streaming systems. In this case, the streaming server uses our allocation algorithm to decide on range of bits to transmit to client such that distortion is minimized at a given client bandwidth.

Finally, we should mention that although FGS coding enables senders to better utilize available bandwidth, this flexibility comes at an expense of coding inefficiency. However, this coding inefficiency can be reduced by utilizing more elaborate coding techniques in scalable coders. For example, more recent H.264/SVC coders produce FGS streams with less than 1 dB quality loss compared to nonscalable streams [Schwarz et al. 2006], which is significantly smaller than that produced by previous standards. This diminishing coding inefficiency indicates that FGS coders are getting more mature, and thus we expect to see more systems based on FGS coders in the near future. Our algorithms can be used in these systems to maximize their performance. In addition, adopting FGS coding requires FGS-enabled decoders with a small computing overhead as indicated in Radha et al. [2001] and Li [2001].

1.3 Organization of the Article

The rest of this article is organized as follows: We discuss related work in Section 2. Section 3 presents a brief overview of FGS coding and validates the rate-distortion model adopted in this paper. In Section 4, we formulate the single- and multiple-frame allocation problems, and we present our rounding scheme. In Section 5, we present the new FGSAssign and mFGSAssign algorithms, and we prove the optimality of FGSAssign. We evaluate our algorithms and compare them against the optimal ones in Section 6. Section 7 concludes this article.

2. RELATED WORK

Distributed and peer-to-peer streaming has recently received significant research attention. For example, the distributed video streaming framework [Nguyen and Zakhor 2002] shows the feasibility and benefits of streaming from multiple servers to a single receiver. The receiver uses a rate allocation algorithm to determine the sending rate for each server to minimize the total packet loss. A tomography-based sender selection protocol is proposed in Hefeeda et al. [2005] to optimize quality at the receiver. Both work do not consider scalable-coded streams. Layered-scalable streams are considered in Cui and Nahrstedt [2003] and Magharei and Rejaie [2006] to cope with the bandwidth heterogeneity. In Cui and Nahrstedt [2003], the multi-sender streaming problem is formulated to maximize the streaming quality of all peers and minimize the load on the originating media distributor. Magharei and Rejaie [2006] designed practical algorithms to adapt to bandwidth dynamics. Different from our work, these two studies employ coarse-grained scalability, and they are not R-D optimized.

Su and Wang [2006] consider a minimization problem for the transmission time of FGS-encoded images. They formulate the problem as a nonlinear program, and then they transform it to a series of LP subproblems. Each LP subproblem determines a peer allocation to maximize the number of received bits in a given delay bound. Unlike our work, the work in Su and Wang [2006] does not consider the characteristics of the R-D curves of video sequences. A framework to solve the problem of streaming interdependent packetized video data units over lossy networks in R-D optimized fashion is proposed in Chou and Miao [2006]. This framework has been generalized by many researchers from various perspectives. For instance, Chakareski and Girod [2003] and Begen et al. [2003] consider the multiple server video streaming problem in R-D optimized way. However, they concentrate on nonscalable and layered coded streams, and do not explicitly specify any R-D models.

The bit distribution problem at different aggregation levels for FGS-encoded sequences is studied in de Cuetos et al. [2005]. This work considers a single bandwidth-constrained streaming server that serves multiple video sequences to clients. The authors formulate a minimization problem for the total distortion for a block of frames of different sequences, and solve it using dynamic programming technique. They show frame aggregations reduce the complexity of the dynamic programming algorithm without compromising the quality. The algorithms in de Cuetos et al. [2005] are not applicable to our problem, because we consider multiple senders serving one receiver, and each sender may not have the complete video stream.

Although Largrangian method and dynamic programming are the most common techniques to solve R-D optimized problems for video compression, two MPEG-2 related R-D optimized problems are solved using linear programming in Sermadevi and Hemami [2003]. To formulate a linear programming problem, they approximate the actual MPEG-2 R-D curves with piecewise linear and convex functions. While this approximation may not be very accurate in nonscalable coders, piecewise linear R-D curves have been observed in FGS-coded sequences (enhancement layer streams) throughout our experimental studies and in Zhang et al. [2003].

3. FGS AND R-D MODELS

In this section, we first provide an overview of fine granularity scalability coding that highlights the important issues to our problem. Then, we present and validate the rate-distortion model used in this article.

3.1 Fine Granularity Scalability Coding

Fine granularity scalability (FGS) has been proposed as part of the MPEG-4 standard to improve rate scalability and error resiliency [Li 2001; Radha et al. 2001]. An FGS-encoded sequence consists of two streams: a nonscalable base layer which provides basic quality, and an enhancement layer that adds incremental quality refinements proportional to the number of bits received. The base layer is typically encoded at a low bit rate to accommodate a wide range of heterogeneous receivers. Due to its small size and its critical role, the base layer is assumed to be delivered over a reliable channel, for example, a channel that uses forward error correction or retransmission if the round trip time is relatively small. The full enhancement layer (i.e., the one that provides full quality) is usually very large and requires high streaming bit rates. In this article, we seek to optimize the quality by controlling the bit rate of the enhancement layer, given that the base layer is encoded at a fixed bit rate and it will be reliably transported to the receiver.

FGS encoding provides great flexibility and efficiency to a server streaming to heterogeneous clients: The server encodes the video only once with the highest quality, and it can easily control the bit rate of a streaming session by truncating the bitstream at appropriate locations. Truncation is a simple operation which can be done in real time.

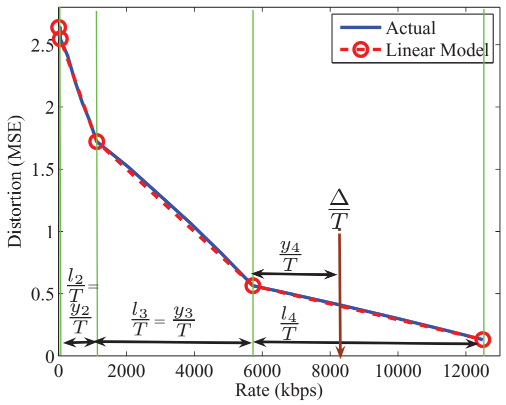


Fig. 3. The piecewise linear R-D model for frame 150 of Akiyo sequence.

Due to the space limitation, we refer readers to Hsu and Hefeeda [2006b] for a brief discussion, and to Li [2001] and Radha et al. [2001] for a detailed treatment on FGS coding.

3.2 Rate-Distortion Models

Rate-distortion (R-D) models are functions that map bit rates to expected distortion—and hence perceived quality—level. Since we are interested in optimizing quality at various rates, we need accurate R-D models. Several analytic R-D models for FGS-encoded sequences have been proposed, for example, the square-root model [Dai and Loguinov 2003], the logarithm model [Dai et al. 2004], and the Generalized Gaussian model [Sun et al. 2005]. The accuracy and complexity of various R-D models are studied in Hsu and Hefeeda [2006a]. In each one of these analytic models, the distortion is related to the rate with a *nonlinear* function. Nonlinear R-D functions make our bit allocation problem (described in Section 4) a nonlinear optimization problem, which is hard (if at all feasible) to solve.

Another approach for obtaining R-D models is by empirically measuring the distortion at various bit rates. Since the range of possible bit rates for the enhancement layer is typically large (order of Mbps), too many sample bit rates are needed to obtain accurate R-D models. This requires decoding the video sequence many times, which is computationally expensive [Wang et al. 2002, page 302]. A third approach for constructing R-D models is to empirically measure distortion only at a few carefully chosen bit rates and interpolate the R-D curve between these points based on some inherent characteristics of the enhancement layer. The piecewise linear R-D model [Zhang et al. 2003], which we adopt in this paper, falls into this category.

In the following sections, we first present the piece-wise linear R-D model and how we extract its parameters from video sequences. Then, we experimentally show its accuracy. Finally, we discuss important properties of this model which we will employ in the formulation and solution of our bit allocation problem.

3.2.1 *The Piecewise Linear R-D Model.* The key idea of the piece-wise linear model [Zhang et al. 2003] (we refer to it simply as the linear model) is that within each bitplane the R-D curve can be approximated by a line segment. Line segments of different bitplanes have different slopes. Figure 3 shows an example. The problem of finding the R-D function is now reduced to measuring the distortion at bit rates that correspond to bitplane boundaries (up to 8 samples in most cases), and computing the slopes of the different line segments.

Table I. The R-D Model Parameters for the First 12 Frames of Foreman Sequence

		bp 1		bp 2		bp 3		bp 4		bp 5		bp 6	
Frame	d	l_1	g_1	l_2	g_2	l_3	g_3	l_4	g_4	l_5	g_5	l_6	g_6
1	5.30	114	−0.92	1132	−0.74	7772	−0.27	19580	−0.08	28087	−0.02	NA	NA
2	11.79	61	−0.15	565	−1.80	3765	−1.11	11722	−0.36	20205	−0.08	27244	−0.02
3	8.64	148	−1.10	2581	−0.92	11139	−0.33	20584	−0.09	27520	−0.02	NA	NA
4	9.28	62	−0.26	400	−1.10	2616	−0.90	11951	−0.34	20852	−0.09	27374	−0.02
5	10.28	203	−1.47	2971	−1.02	12616	−0.36	20807	−0.09	27278	−0.02	NA	NA
6	10.55	205	−1.32	3495	−1.02	11987	−0.36	20717	−0.09	27304	−0.02	NA	NA
7	7.52	80	−0.34	1721	−0.73	11698	−0.31	21630	−0.09	27743	−0.02	NA	NA
8	9.16	169	−1.21	2693	−0.94	11700	−0.33	21463	−0.09	27392	−0.02	NA	NA
9	7.89	105	−0.82	2148	−0.83	11118	−0.31	21471	−0.09	27625	−0.02	NA	NA
10	7.26	78	−0.35	1700	−0.71	11414	−0.29	22241	−0.09	27732	−0.02	NA	NA
11	9.00	145	−1.07	2425	−0.92	12131	−0.34	21076	−0.09	27743	−0.02	NA	NA
12	9.15	148	−1.73	2476	−0.92	12138	−0.34	21209	−0.09	27728	−0.02	NA	NA

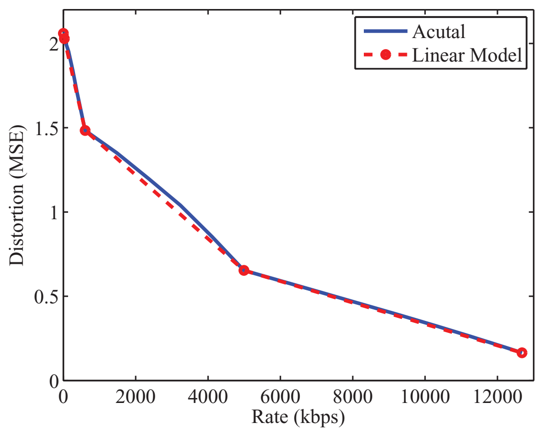
—Bitplane sizes (l) are in bytes, while slopes (g) are 10^{-3} scaled to save page space.
—Non-existing bitplanes are annotated with NA.

Building and storing the linear R-D model for a given sequence is quite efficient. We first identify the bitplane boundaries from the header inserted in each bitplane during the FGS encoding process. We compute the size of each bitplane h, and denote it by l_h. If there are z bitplanes, we decode the sequence $z + 1$ times but in each time we *truncate* the enhancement layer at a different bitplane boundary. After decoding we compute the distortion—in mean-square error (MSE)—between the original and reconstructed sequences. Then we compute the slope g_h for each bitplane h. The R-D model is extracted only once from the video sequence and stored in a meta file. We need to store only the size of each bitplane and the slope of the line segment in that bitplane. The meta file adds a negligible storage overhead, up to 64 bytes per frame. Frames sizes are usually in order of tens of kilo bytes. Table I shows the linear R-D model parameters for the first twelve frames of Foreman sequence.

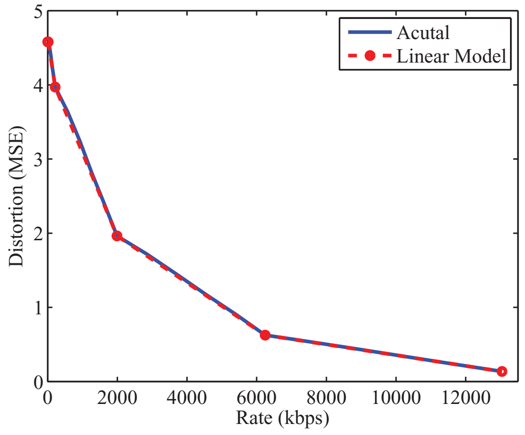
3.2.2 *Accuracy of the Linear R-D Model.* Since the accuracy of the adopted R-D model is crucial to our problem, we validate the accuracy of the linear R-D model. This accuracy validation was not done before in the literature, the work in Zhang et al. [2003] conducted a limited validation analysis only at a few sampling rates. We compare the accuracy of the linear model against the actual R-D curves, which are obtained as follows. We select six equally spaced sampling rates on each bitplane. Then, we compute the actual distortion by decoding and comparing the original and reconstructed sequences at each sampling rate. We carry out the comparison over several video sequences of different temporal and spatial complexities.

For visual validation, we *randomly* choose a few frames from different sequences and plot the actual R-D curves and the ones estimated by the linear model. Figures 3 and 4 illustrate that the linear model approximates the actual R-D curves quite well. For more rigorous validation, we compute the absolute error between the actual and linear model curves at all sampling rates. We compute the average error per frame, and we repeat that for all frames in the considered sequences. The results, plots are given in Hsu and Hefeeda [2006b], confirm the accuracy of the linear R-D model: the average error is less than 2% in almost all cases.

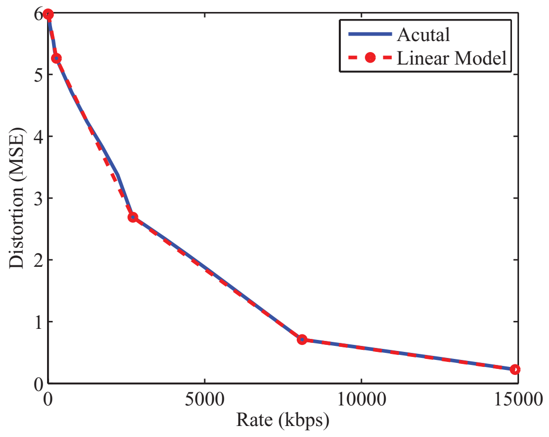
3.2.3 *Properties of the Linear R-D Model.* By examining a large set of R-D parameters for many frames, we found that curves produced by the linear model has the following property: The slopes of the line segments form a monotonically increasing series, that is, $g_1 < g_2 < \cdots < g_z < 0$, where g_h is the slope of line segment in bitplane h. Figure 5 demonstrates this property. This is intuitive because, more significant bitplanes carry higher significant bits, and thus the per-bit reduction in distortion is



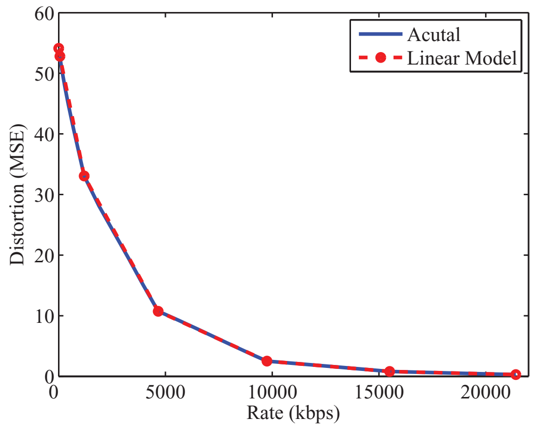
(a) Akiyo sequence, frame 125



(b) Mother sequence, frame 110



(c) Foreman sequence, frame 100



(d) Mobile sequence, frame 90

Fig. 4. The accuracy of the linear R-D model for randomly chosen frames from different sequences.

higher. This means that the most significant bitplanes will have steeper and more negative slopes (refer to Figure 4).

While the above explanation of slope monotonicity makes sense, we have seen a few anomalies. For instance, frame 2 of Foreman (see Table I) has $g_1 = -0.15 > -1.8 = g_2$. Almost all anomalies occurred between bitplanes 1 and 2. From further examination, we found that the anomalies occur when the size of the first bitplane is very small, around 100 bytes or less as shown in Table I. At that small size, the overhead of the bitplane header becomes significant, which negatively impacts the *effective* per-bit reduction in distortion (i.e., the slope) for bits in bitplane 1. Notice from Table I that the smaller the size of bitplane 1, the higher the negative impact.

Because some of the proofs of our optimal bit allocation problem require the monotonicity of the slopes, we propose a simple processing of linear R-D curves to maintain monotonicity: We aggregate any two neighboring bitplanes into a single bitplane whenever there is a violation in monotonicity. For

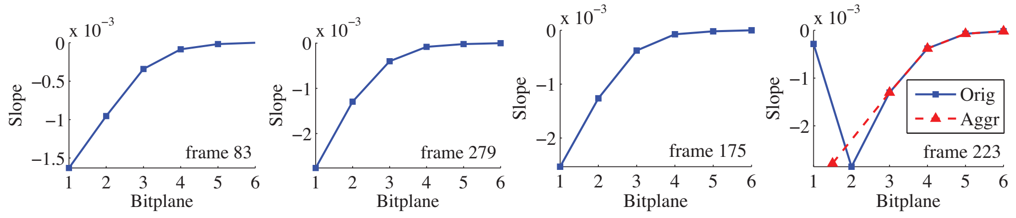


Fig. 5. The monotonicity property of the linear R-D model: $g_1 < g_2 < \cdots < g_z < 0$. The figure shows the slope parameters g of two frames from Foreman (left) and two frames from Mobile (right) sequences. Note that, only the rightmost frame requires aggregation.

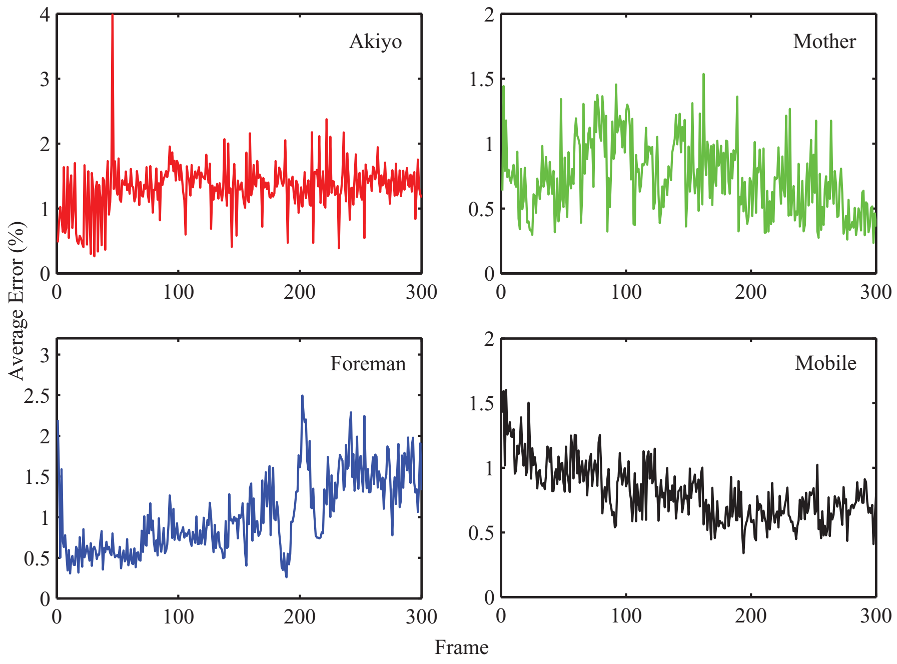


Fig. 6. The accuracy of the linear R-D model after bitplane aggregation. The figure shows the average error between curves estimated by the (modified) linear model and the actual R-D curves.

instance, the rightmost subfigure in Figure 5 shows the slopes before and after bitplane aggregation. Notice that if bitplane aggregation is needed, it is almost always sufficient to aggregate the first two bitplanes and in very rare cases between the second and third bitplanes. To verify this, we examined slopes of all bitplanes of all frames in four video sequences: Akiyo, Mother, Foreman, and Mobile. We observed only two pairs of consecutive bitplanes out of 5734 total bitplanes violated the monotonicity property. Furthermore, these two violations can be eliminated by two more bitplane aggregations.

Finally, we examined the impact of the bitplane aggregation on the accuracy of the *transformed* linear R-D curves. We repeat the accuracy assessment experiments in the previous subsection, but using linear R-D curves after bitplane aggregation is performed on them. Figure 6 shows that the average error between the modified linear R-D curves and the actual ones is less than 2% in almost all cases, the same accuracy for the untransformed R-D curves.

Table II. Notations Used in this Article

Notation	Definition	Notation	Definition
n	Number of potential senders.	Δ_i	Number of bits allocated to sender i.
M	Number of frames in a sequence.	r_i	Sending rate allocated to sender i.
m	Number of frames in a block.	z	Number of bitplanes in the enhancement layer.
T	Frame period.	d	Distortion when only base layer bits are sent.
s_i	Size of the bit stream stored at sender i.	l_h	Size of bitplane h.
b_i	Outgoing bandwidth of sender i.	g_h	Slope of the line segment in bitplane h.
b_I	Incoming bandwidth of the receiver.	y_h	Number of bits transmitted from bitplane h.

4. THE BIT ALLOCATION PROBLEM: FORMULATION AND SOLUTION

In this section, we consider the bit allocation problem for multi-sender video streaming systems that use FGS encoded streams. We assume that the base layer is coded at a reasonably low bit rate and can be transmitted through a reliable channel, for example, a forward error correction (FEC) protected channel. Our goal is to develop a practical algorithm for many-to-one streaming sessions such that the available resources are intelligently allocated to achieve the maximal perceptual streaming quality.

4.1 Problem Formulation

The bit allocation problem we address in this article can be stated as follows: Consider multiple senders that can potentially serve an FGS-encoded video sequence to a receiver, senders have different portions of the sequence and have different outgoing bandwidth, and the receiver has a limit on the incoming bandwidth. Determine the streaming rate and the range of bits allocated to each sender to achieve the best possible video quality at the receiver. It is assumed that the base layer is coded at a low, and fixed bit rate and can be reliably delivered by any sender. We seek to optimize the streaming of the enhancement layer because it contributes significantly larger bit rates than the base layer, and unlike the base layer, its bit rate can be controlled.

We address this problem in two steps. First, we optimize the quality of individual frames of the sequence. That is, we formulate and solve the problem for each frame. In the second step, we divide the sequence into blocks of frames, each has a fixed number of frames. Then, we formulate and solve the optimization problem for each block. Table II summarizes notations used in this article.

We now formulate the allocation problem for individual frames. Let T denote the frame period in seconds, which is the multiplicative inverse of the frame rate. T is fixed for all frames in the considered sequence. Assume there are n potential senders, each with outgoing bandwidth b_i ($i = 1, 2, \ldots, n$). Each sender i holds s_i contiguous bits, which is a portion of the enhancement layer bit stream. Without loss of generality, we assume that $s_1 \leq s_2 \leq \cdots \leq s_n$, otherwise, we re-label senders to achieve that. Note that the nature of FGS-encoded sequences implies that a lower quality bit stream is always a subset of a higher quality one. That is, s_i is always a *prefix* of s_{i+1} for all $i = 1, 2, \ldots, n-1$. The receiver incoming bandwidth is denoted by b_I.

Solving the allocation problem should yield an allocation policy $A = \{(\Delta_i, r_i) | i = 1, 2, \ldots, n\}$, where Δ_i is the number of bits allocated to peer i and $r_i \leq b_i$ is its streaming rate. Bits are allocated to peers as follows. Peer 1 transmits the range from 0 to $\Delta_1 - 1$, peer 2 transmits from Δ_1 to $\Delta_1 + \Delta_2 - 1$, and in general peer i transmits from $\sum_{t=1}^{i-1} \Delta_t$ to $\sum_{t=1}^{i} \Delta_t - 1$.

Mathematically, the bit allocation problem for a given frame can be formulated as:

$$\min_{A} \quad D\left(\sum_{t=1}^{n} \Delta_t \right) \tag{1a}$$

$$\text{s.t.} \;\; \Delta_i - r_i T \leq 0 \tag{1b}$$

$$\sum_{t=1}^{i} \Delta_t \leq s_i \tag{1c}$$

$$r_i \leq b_i \tag{1d}$$

$$\sum_{t=1}^{n} r_t \leq b_I \tag{1e}$$

$$\Delta_i, r_i \in \mathbb{N}; \quad i = 1, 2, \ldots, n. \tag{1f}$$

The objective function in (1a) is to find the optimal allocation A^* that minimizes the reconstruction distortion, and hence maximizes the rendered quality. The constraints can be explained as follows: (1b) ensures that each sender has enough time to transmit all bits allocated to it ($\Delta_i - r_i T \leq 0 \implies \Delta_i / r_i \leq T$). (1c) ensures that the allocated bits to each sender are within the portion of bits stored at that sender, while (1d) and (1e) ensure that the limits on the incoming and outgoing bandwidth of the receiver and senders are not exceeded.

In order to solve this optimization problem, we need the mapping between the distortion and total number of bits received. Note that dividing the total number of bits by the (fixed) frame period T yields the bit rate. As we discussed in Section 3.2, we adopt the linear R-D model for this mapping because: (i) it is fairly accurate, (ii) its parameters can be efficiently extracted from the video sequence, and more importantly, (iii) it results in a linear objective function and hence efficient solution of the optimization problem. Recall that the linear R-D model divides the R-D curve into several line segments, each corresponds to a bitplane h and has a different slope g_h (see Figure 3). The slope g_h represents the per-bit reduction in the distortion in bitplane h. Therefore, in order to compute the total distortion for a possible bit allocation, we need to find how many bits are transmitted from each bitplane. To do that, we introduce a new variable y_h ($h = 1, 2, \ldots, z$) to represent the number of bits transmitted from bitplane h. z is the number of bitplanes in the enhancement layer of the considered frame. Now the distortion can be computed as $d + \sum_{h=1}^{z} g_h \; y_h$, where d is the distortion when only the base layer is transmitted. The optimization problem in (1) can be rewritten as:

$$\min_{A} \; D\left(\sum_{t=1}^{n} \Delta_t\right) = d + \sum_{v=1}^{z} g_v \; y_v \tag{2a}$$

$$\text{s.t.} \qquad \Delta_i - r_i T \leq 0 \tag{2b}$$

$$\sum_{t=1}^{i} \Delta_t \leq s_i \tag{2c}$$

$$r_i \leq b_i \tag{2d}$$

$$\sum_{t=1}^{n} r_t \leq b_I \tag{2e}$$

$$y_h \leq l_h \tag{2f}$$

$$\sum_{t=1}^{n} \Delta_t = \sum_{v=1}^{z} y_v \tag{2g}$$

$$\Delta_i, r_i, y_h \in \mathbb{N}; \quad i = 1, 2, \ldots, n; \quad h = 1, 2, \ldots, z. \tag{2h}$$

Notice the two new constrains in (2f) and (2g). (2f) makes sure that the number of bits transmitted from a bitplane does not exceed the size of that bitplane, whereas (2g) ensures that the total number of bits transmitted from different bitplanes is exactly the same as the number of bits allocated to senders.

Unlike (1), (2) is an integer linear programming (ILP) problem, because the objective function as well as all constraints are linear. While ILP problems are less complex than nonlinear problems, they are still NP-hard [Cormen et al. 2001, page 777]. In the next section, we present a rounding scheme that transforms the ILP problem in (2) to a linear programming (LP) problem, which can be solved using the Simplex method or other efficient LP solvers. But before doing so, we need to make sure that the optimal solution for the ILP in (2) will produce a *valid* FGS-encoded bit stream. An FGS-encoded bit stream is valid if it has a contiguous stream of bits with no gaps between them. If there is a gap in the bit stream, the FGS decoder will ignore the rest of the bit stream beyond the gap, which will reduce the quality. The following lemma proves the validity of the optimal solution of (2).

LEMMA 1. *An optimal solution for* (2) *produces a contiguous FGS-encoded bit stream with no bit gaps.*

PROOF. To prove this lemma, we need to show that no bits from a bitplane j will be transmitted before all bits of bitplanes $h < j$ are transmitted, where $h, j = 1, 2, \ldots, z$ and z is the number of bitplanes. That is, we need to show that if an optimal allocation results in $y_j^* \neq 0$ for any $j = 2, 3, \ldots, z$, then it must be the case that $y_h^* = l_h$, where $h = 1, 2, \ldots, j-1$, and l_h is the size of bitplane h. We prove this by contradiction.

Assume that the optimal allocation A^* produced $y_j^* \neq 0$ and there exists $y_h^* < l_h$ for some $h < j$. We construct another allocation $\widehat{A}$ which is exactly the same as A^*, except we shift $q = \min(y_j^*, l_h - y_h^*) > 0$ bits from bitplane j to bitplane h. $\widehat{A}$ is indeed a feasible allocation because it satisfies all the constraints in (2). Furthermore, the distortion associated with $\widehat{A}$ is given by $\widehat{D} = D^* + (g_h - g_j)q < D^*$, since g_h is always less than g_j because slopes are monotonically increasing, as discussed in Section 3.1. This is a contradiction because D^* is supposed to be the minimum distortion. □

4.2 Rounding Scheme and Linear Programming Solution

In this section, we present a rounding scheme that transforms the ILP problem in (2) to a linear programming (LP) problem, which can be solved using the Simplex method. We show that our rounding scheme results in a solution that is within a negligible gap from the optimal.

Our LP formulation of the bit allocation problem is exactly the same as (2), except the constraint in (2h) is now relaxed to be:

$$\Delta_i, r_i, y_h \in \mathbb{R}^+ \cup \{0\}; \quad i = 1, 2, \ldots, n; \quad h = 1, 2, \ldots, z. \tag{3}$$

Let $\widehat{A} = \{(\widehat{\Delta}_i, \widehat{r}_i) | i = 1, 2, \ldots, n\}$ be the optimal allocation produced by solving the LP problem. $\widehat{\Delta}_i, \widehat{r}_i$ are in general non-negative real numbers. To obtain integer solutions for the original ILP problem, we propose the following rounding scheme:

$$\overline{r}_i = \lfloor \widehat{r}_i \rfloor, \quad \text{and } \overline{\Delta}_i = \begin{cases} \lfloor \widehat{\Delta}_i \frac{\widehat{r}_i - 1}{\widehat{r}_i} \rfloor, & \widehat{r}_i > 1 \\ 0, & 0 \leq \widehat{r}_i \leq 1. \end{cases} \tag{4}$$

In the following two lemmas, we prove that this rounding scheme indeed produces a feasible solution, and that solution is very close to the optimal one.

LEMMA 2. *Rounding of the optimal solution of the relaxed linear programming problem using the rounding scheme in* (4) *always produces a feasible solution for the integer linear programming problem defined in* (2).

PROOF. We only need to prove that constraint (2b) is satisfied, that is, we need to show that $\overline{\Delta}_i - \overline{r}_i T \leq 0$. Notice that all other constraints are automatically satisfied because we round down both of $\widehat{r}_i$ and $\widehat{\Delta}_i$.

We first consider the case when $\widehat{r_i} > 1$. For any $i = 1, 2, \ldots, n$, we have:

$$\frac{\overline{\Delta_i}}{\overline{r_i}} = \frac{\lfloor \widehat{\Delta_i} \frac{\widehat{r_i}-1}{\widehat{r_i}} \rfloor}{\lfloor \widehat{r_i} \rfloor} \leq \frac{\widehat{\Delta_i} \frac{\widehat{r_i}-1}{\widehat{r_i}}}{\lfloor \widehat{r_i} \rfloor} < \frac{\widehat{\Delta_i} \frac{\widehat{r_i}-1}{\widehat{r_i}}}{\widehat{r_i} - 1} = \frac{\widehat{\Delta_i}}{\widehat{r_i}} \leq T,$$

where the first inequality (from the left) is from the definition of the floor function, and the second inequality is due to the fact that $\lfloor \widehat{r_i} \rfloor > \widehat{r_i} - 1$. Next we consider the case when $0 \leq \widehat{r_i} \leq 1$. In this case, $\overline{\Delta_i} = 0$. Therefore, $\overline{\Delta_i} - \overline{r_i}T = -\overline{r_i}T \leq 0$, because $\overline{r_i} \geq 0$, and the constraint is satisfied. □

Now we bound the gap between the rounded solution and the optimal one, and show that this gap is negligible.

LEMMA 3. *The rounding scheme in* (4) *results in a total number of bits that is smaller than the optimal number of bits by at most $nT + n$, where n is the number of senders and T is the frame period.*

PROOF. We first compute the gap between the optimal and rounded solutions for an arbitrary sender i, where $i = 1, 2, \ldots, n$. If $\widehat{r_i} > 1$, we have:

$$\widehat{\Delta_i} - \overline{\Delta_i} = \widehat{\Delta_i} - \left\lfloor \widehat{\Delta_i} \frac{\widehat{r_i} - 1}{\widehat{r_i}} \right\rfloor < \widehat{\Delta_i} - \widehat{\Delta_i} \frac{\widehat{r_i} - 1}{\widehat{r_i}} + 1 = \frac{\widehat{\Delta_i}}{\widehat{r_i}} + 1 \leq T + 1.$$

If $0 \leq \widehat{r_i} \leq 1$, we have

$$\widehat{\Delta_i} - \overline{\Delta_i} = \widehat{\Delta_i} \leq \widehat{r_i}T \leq T < T + 1.$$

Therefore, the gap for any sender is bounded by $T + 1$. Thus, for all n senders, the gap is at most $nT + n$. □

Lemma 3 shows that the gap between the solution resulted from our rounding scheme and the optimal one is negligible. To put this gap into perspective, consider an extreme-case streaming session in which 30 senders are concurrently streaming to a single receiver. Assume that the frame rate is 30 frames per second. Then, the gap is at most $30/30 + 30 = 32$ bits. That is, the rounded solution may transmit at most 2 bytes less than the optimal one from any given frame, which is indeed negligible given that frame sizes are in the order of kilo bytes.

4.3 Multiple Frame Formulation

In this section, we formulate and solve the bit allocation problem for blocks of frames, each block has a fixed number of frames m. The objective is to minimize the total distortion across all frames of a given block. As discussed in Section 1, solving the bit allocation problem at the block level is expected to produce better quality than solving it at the frame level. This is because of the flexibility of distributing bits among frames of the same block, which is leveraged by the optimal solution to allocate bits where they can reduce the distortion the most.

The generalization of the bit allocation problem is given in (5). Notice the addition of the suffix f to the variables to denote a frame within the block.

$$\min_{A_m} \sum_{u=1}^{m} D\Big(\sum_{t=1}^{n} \Delta_{tu}\Big) = \sum_{u=1}^{m} d_u + \sum_{u=1}^{m}\sum_{v=1}^{z_u} g_{vu}\; y_{vu} \tag{5a}$$

$$\text{subject to} \sum_{u=1}^{m} \Delta_{iu} - m\, r_i\, T \leq 0 \tag{5b}$$

$$\sum_{t=1}^{i} \Delta_{tf} \leq s_{if} \tag{5c}$$

$$r_{if} \leq b_i \tag{5d}$$

$$\sum_{t=1}^{n} r_{tf} \leq b_I \tag{5e}$$

$$y_{hf} \leq l_{hf} \tag{5f}$$

$$\sum_{t=1}^{n} \Delta_{tf} = \sum_{v=1}^{z_f} y_{vf} \tag{5g}$$

$$\Delta_{if}, r_{if}, y_{hf} \in \mathbb{N};\;\; i = 1, 2, \ldots, n;\;\; f = 1, 2, \ldots, m;\;\; h = 1, 2, \ldots, z_f. \tag{5h}$$

Since this formulation is a straightforward generalization of the single frame case, the rounding scheme in (4) is also applicable to the solution of its LP relaxation. However, the generalization increases the number of variables and constraints from $O(n)$ to $O(mn)$. To illustrate, for $m = 100$ and $n = 16$, this formulation has $3,700$ variables and $3,916$ constraints, if we assume every frame consists of 5 bitplanes. Despite the efficiency of modern LP solvers, solving this problem for every block requires significant computation resources, and more importantly, may not be feasible in real time. Using the Optimization Toolbox of Matlab and our rounding scheme, we have implemented an algorithm to find the optimal solution for the multiple-frame problem. The algorithm is denoted as mOPT in Section 6.

5. EFFICIENT ALLOCATION ALGORITHMS

The LP formulation of the bit allocation problems, presented in the previous section, can be solved by the Simplex method. On average, the Simplex method is efficient, but it has a worst-case exponential running time [Papadimitriou and Steiglitz 1998, Sect. 8.6]. While there exist worst-case polynomial time methods (e.g., Karmarkar's interior-point algorithm) for solving general LP optimization problems, they are quite complex to implement, and they have very large average running times, which in many cases exceed the average running time of the Simplex method [Goldfarb and Todd 1989].

In this section, we propose two efficient allocation algorithms. The first one optimally solves the single-frame bit allocation problem in $O(n \log n)$ time, where n is number of senders. The second algorithm is a heuristic that produces near-optimal results for the multiple-frame allocation problem, it runs much faster than the optimal algorithm, and can be used in real time.

5.1 FGSAssign: An Optimal Algorithm for the Single-Frame Bit Allocation Problem

We propose a greedy algorithm, called FGSAssign, to efficiently solve the single frame bit allocation problem. The pseudo code for the algorithm is given Figure 7. The basic idea of the algorithm is to transmit the maximum possible number of bits from each sender. FGSAssign does so by first sorting all senders based on the portion of the stream stored at each of them, such that $s_1 \leq s_2 \leq \cdots \leq s_n$. Then,

FGSAssign

```
/* Inputs:
$n$: number of senders.
$b_i$: outgoing bandwidth for every sender $i$ ($i = 1, 2, \ldots, n$).
$b_I$: incoming bandwidth for the receiver.
$s_i$: portion of the stream cached by every sender $i$.
*/
/* Outputs:
$r_i$: allocated sending rate for every sender $i$.
$\Delta_i$: assigned stream length for every sender $i$.
*/
1.  Sort all senders based on $s_i$, where $s_1 \leq s_2 \leq \cdots \leq s_n$;
2.  $x_0 = \cdots = x_n = 0$;  $\Delta_1 = \cdots = \Delta_n = 0$;  $r_{agg} = 0$;
3.  for $i = 1$ to $n$ do
4.      $x_i = \min(x_{i-1} + b_i T, s_i)$;
5.      $r_i = (x_i - x_{i-1})/T$;
6.      if $(r_{agg} + r_i < b_I)$ then
7.          $r_{agg} = r_{agg} + r_i$;
8.          $\Delta_i = x_i - x_{i-1}$;
9.      else
10.         $r_i = b_I - r_{agg}$;
11.         $\Delta_i = T \times r_i$;
12.         return ;
13. endfor
```

Fig. 7. Pseudo code for an optimal and efficient algorithm for the single-frame bit allocation problem.

it sequentially allocates to sender i ($i = 1, 2, \ldots, n$) the maximum number of bits which sender i can transmit within the frame period T. The allocated bits must be available at sender i, that is, they are a subset of s_i, and they do not overlap with bits allocated to j ($1 \leq j < i$). Thus FGSAssign avoids holes as well as overlapping of bits among senders. Therefore, the allocation will result in a contiguous bit stream, which is necessary for decoding FGS-encoded enhancement layers.

The following theorem proves that FGSAssign is optimal and efficient.

THEOREM 1. *The FGSAssign algorithm terminates in $O(n \log n)$ steps, where n is the number of senders, and it produces an allocation that minimizes the distortion of individual frames.*

PROOF. The termination and time complexity part is straightforward: sorting of s_i's takes $O(n \log n)$ steps and the for-loop iterates up to n times, each taking a constant number of operations.

For the optimality part, we first notice that for individual frames, sending more bits always results in smaller distortion. Therefore, sending the maximum number of bits corresponds to the minimum distortion. The maximum number of bits is determined based on either: (i) the receiver incoming bandwidth b_I, or (ii) senders outgoing bandwidth b_i and portions of stream stored s_i ($1 \leq i \leq n$). In the first case, because s_i's are sorted, the algorithm fills up the entire receiver's bandwidth with non overlapping bits and returns from the for-loop in line 12. Thus the maximum number of bits will be transmitted.

In the second case, the algorithm terminates after finishing n iterations of the for-loop. In every iteration i of the for-loop, the algorithm assigns to sender i the maximum number of bits that this sender can transmit, constrained only by the outgoing bandwidth b_i and the length of the stored portion of the stream s_i. In other words, the algorithm makes a greedy decision (Δ_i, r_i), and solves a subproblem in the $i + 1$ iteration. To prove that this greedy approach is optimal, we show that the problem has two properties: greedy-choice and optimal substructure [Cormen et al. 2001, Chapter 16].

The greedy-choice property guarantees that a globally optimal solution can be arrived at by a greedy choice. Suppose $\{(\Delta^*_u, r^*_u)|u = i, i+1, \ldots, n\}$ is an optimal solution to the original problem in iteration i. Since (Δ_i, r_i) is a greedy choice, we have $\Delta_i \geq \Delta^*_i$ (and $r_i \geq r^*_i$). Now, we construct a new solution from the optimal one by shifting bits from other peers to i, such that the new solution contains the greedy decision. Obviously, the number of bits remains the same, thus, we have an optimal solution consists of the greedy choice.

The optimal substructure property ensures that combining a greedy decision with an optimal solution to the subproblem results in an optimal solution to the original problem. We prove this property by contradiction. Consider iteration i. Suppose the combination of the greedy decision (Δ_i, r_i) and a subproblem optimal solution $\{(\Delta_v, r_v)|v = i+1, i+2, \ldots, n\}$ is not an optimal solution to the original problem. Thus, we can find an optimal solution $\{(\Delta^*_u, r^*_u)|u = i, i+1, \ldots, n\}$ such that $\sum_{u=i}^{n} \Delta^*_u > \sum_{v=i}^{n} \Delta_v$. Since $\Delta_i \geq \Delta^*_i$ by the greedy decision, we must have $\sum_{u=i+1}^{n} \Delta^*_i > \sum_{v=i+1}^{n} \Delta_v$, which is a contradiction because $\sum_{v=i+1}^{n} \Delta_v$ is assumed to be an optimal solution to the subproblem. □

5.2 Multiple-frame Allocation Algorithm

In this section, we present a heuristic algorithm, called mFGSAssign, for the multiple-frame bit allocation problem. mFGSAssign strives to achieve two goals: (i) minimize the total distortion in a block, and (ii) reduce the quality variation among successive frames. The first goal is the same as the objective function of the LP formulation for the multiple-frame bit allocation problem (Section 4.3).[1] The second goal is important to render *consistent* playout quality perceived by users, because it tries to reduce quality fluctuations among frames. Previous studies have shown that quality fluctuations have a negative impact on users [Zink et al. 2003]. Our experimental results (Section 6), show that mFGSAssign achieves both objectives: It results in total distortion values close to the optimal values, and it renders a fairly smooth quality of successive frames. Furthermore, its running time is significantly shorter than that of the optimal algorithm.

The pseudo code of the mFGSAssign algorithm is given in Figure 8. The video sequence is assumed to be divided into blocks, each of size m frames, and the algorithm solves the allocation problem for a given block. The inputs to the mFGSAssing algorithm are: (i) block size m; (ii) a set of n potential senders, each sender i has an outgoing bandwidth b_i and stores s_{if} bits of each frame f $(f = 1, 2, \ldots, m)$ in the block; and (iii) a table rd_tbl that provides mapping between bit rate and distortion (the table is built using the linear R-D model discussed in Section 3.2). The algorithm has two phases. The first phase (lines 1–16) estimates a target distortion D that is feasible and achieves the two objectives above. It also computes, for every frame f in the block, the bit budget B_f required to achieve D. Then, the second phase (lines 17–19) calls FGSAssign for each frame to optimally allocate B_f among the n senders.

The idea of estimating the target distortion is depicted in Figure 9, and can be summarized as follows. The algorithm first computes the maximum allowed bit budget B^{max} for the block. It also computes, using the rd_tbl, the upper and lower bounds for distortion in this block. The maximum distortion in a frame occurs when no bits from the enhancement layer of that frame are received. Thus, the upper bound D_u is set as the highest value of the maximum distortion across all frames in the block. On the other hand, the minimum distortion occurs when all bits of the enhancement layer are received. Thus, the lower bound D_l is set as the lowest value of the minimum distortion across all frames in the block. For example, the four frames in Figure 9 have maximum distortion values of 7.98, 4.52, 8.51, and 8.02,

[1]Formulating and solving optimization problems with multiple objective functions are difficult and tricky, because we need to quantify the relative importance of the objective functions. In general, these problems are more complex than single objective ones unless the preferences are irrelevant or fully understood [Marler and Arora 2004].

```
mFGSAssign

/* Inputs:
$n$: number of senders.
$m$: number of frames in the block.
$b_i$: outgoing bandwidth for every sender $i$ ($i = 1, 2, \ldots, n$).
$b_I$: incoming bandwidth for the receiver.
$s_{if}$: portion of the stream cached by every sender $i$ for each frame $f$ ($f = 1, 2, \ldots, m$) in the block.
$rd\_tbl$: rate-distortion parameters of the linear R-D model.
*/
/* Outputs:
$r_{if}$: allocated sending rate for every sender $i$ for each frame $f$.
$\Delta_{if}$: assigned stream length for every sender $i$ for each frame $f$.
*/
1. $B^{max} = \min(b_I, \sum_{k=1}^{n} b_k) \times T \times m$; // max bit budget for the block
2. $D_u$ = getMaxDistortion($rd\_tbl$);   $D_l$ = getMinDistortion($rd\_tbl$);
3. while ($D_l < D_u$) do
4.       $B = 0$; // estimated bit budget for the block
5.       $D = (D_u + D_l)/2$; // estimated target distortion for each frame in the block
6.       for $f = 1$ to $m$ do
7.             $B_f = T \times$ getRate($rd\_tbl, f, D$); //bit budget for frame $f$ to achieve $D$
8.             $B = B + B_f$;
9.       endfor
10.      if ($0 \leq B^{max} - B \leq B^{th}$) then
11.            break; //max and estimated bit budgets are close enough
12.      if ($B < B_{max}$) then
13.            $D_u = D$;
14.      else
15.            $D_l = D$;
16. end while
17. for $f = 1$ to $m$ do
18.      Call FGSAssign to divide up to $B_f$ bits among the $n$ senders, based on their $b_i$'s and $s_{if}$'s;
19. endfor
```

Fig. 8. Pseudo code for a heuristic algorithm for the multiple-frame bit allocation problem.

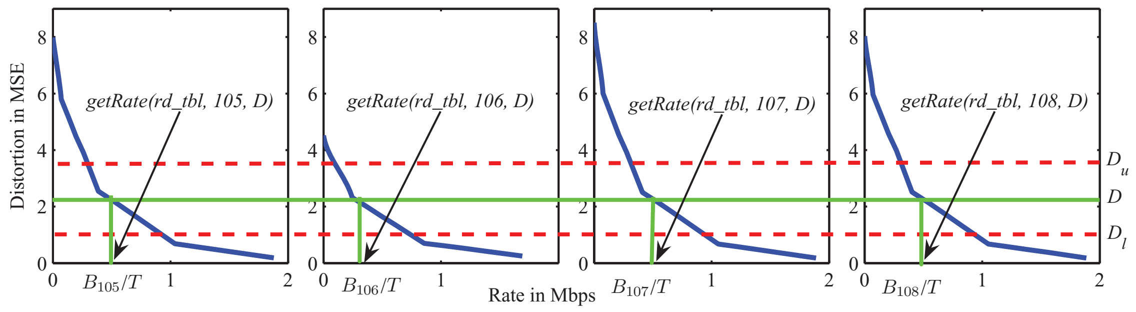


Fig. 9. An example showing the estimation of the target distortion and the associated bit budget for a block of four frames from the Foreman sequence. B_{105}–B_{108} are the required bit budgets for frames 105–108 to achieve the same distortion level D in all of them.

therefore, initially $D_u = 8.51$; and minimum distortion values of 0.19, 0.25, 0.18, and 0.18, therefore, initially $D_l = 0.18$.

Using the distortion bounds, the algorithm uses binary search to get the closest feasible bit budget to B^{max}. It first guesses a target distortion as $D = (D_u + D_l)/2$ and computes the needed bit budget B, by summing up the bit budget required for each frame to achieve D (see Figure 9). $B < B^{max}$ indicates that we can potentially transmit more bits and reduce the distortion. Thus, we set $D_u = D$, reducing the search space by half. Similarly, we set $D_l = D$ if $B > B^{max}$. We terminate the search for the target distortion when the needed bit budget is within a predefined threshold ($B^{\text{th}} \geq 0$) from B^{max}. We use this threshold B^{th} because there may not be a target distortion that produces exactly B^{max}. A threshold of at most 100 bytes would guarantee a successful search for all practical situations. In our experiments, we used a much smaller threshold $B^{\text{th}} = 1$ byte, and we did not encounter any failed search.

As a final note on the mFGSAssign algorithm, we should mention that the distortion variables D, D_u, D_l are treated as integer variables to guarantee termination and prove the time complexity given below. But, in fact, they are small real numbers (almost always less than 65). In our implementation, we multiply the original distortion values by 10^c then round to the nearest integers. c is a constant that could be set to achieve the desired precision. Increasing c does not significantly increase the running time, because the algorithm does binary search, and thus the number of iterations in the loop will be up to the logarithm of the distortion range. For all practical purposes, c between 10 and 20 should be more than enough.

Finally, the following lemma provides the time complexity of the mFGSAssign algorithm.

LEMMA 4. *The mFGSAssign terminates in $O(m\,n\,\log n)$ steps.*

PROOF. The while loop has a constant maximum number of iterations: $O(\log(D_u - D_l))$. Each iteration takes up to $O(m)$ steps, because the getRate() function checks a fixed number of biplanes (MPEG-4 standard [ISO/IEC 14496-2:2004] reserves 5 bits for *max_bitplanes* field, which supports up to 31 bitplanes). Thus, the cost for the while loop is $O(m)$. Furthermore, from Lemma 1, the m calls to FGSAssign cost $O(m\,n\,\log n)$ steps, and the lemma follows. □

6. EVALUATION

In this section, we evaluate the allocation algorithms presented in the previous sections. We address several issues: (i) the importance of the scalable allocation problem, (ii) the optimality of the FGSAssign algorithm, (iii) the performance of the mFGSAssign algorithm versus the optimal one, (iv) the average running times of our algorithms, and (v) the effect of block size on the video quality and running times. We first describe our experimental setup, and then present the results.

6.1 Experimental Setup

6.1.1 *Software Used and Developed.* In our experiments, we use the MPEG-4 Reference Software Version 2.5 [ISO/IEC 14496-5:2004] developed by Microsoft as an experimental package for the MPEG-4 standard. It is implemented in C++ and contains three major executables: *encoder*, *decoder*, and *fgs_server*. The *encoder* is a configurable MPEG-4 encoder that can compress a raw video file into a base layer and an enhancement layer bitstream. Each bitstream is stored in a separate file. The *decoder* is FGS-enabled, that is, it can process an incomplete enhancement layer and produces a raw video file with proportional quality improvements. The *fgs_server* is a utility to trim the enhancement layer according to a given target bit rate. It does so by calculating the bit budget of each frame at the target bit rate. The truncated bitstream is then saved in a new file. We instrument the reference software to extract various statistics of a video sequence. For instance, we collect the transform coefficients, number

Table III. Detailed Parameters of the Four Considered Streaming Scenarios

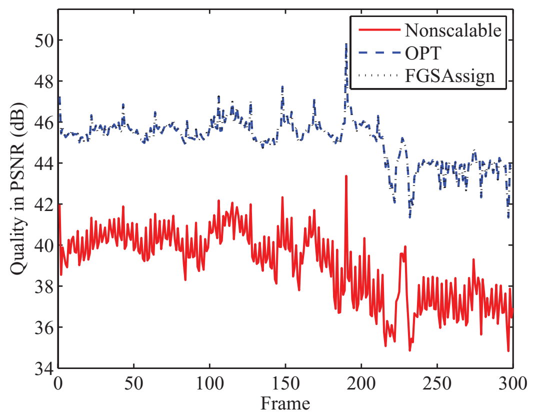
Scenario	Represented environment	Receiver (incoming bw)	Senders (outgoing bw, bitstream ver)
I	Internet streaming session among residential and corporate participants	(10 Mbps)	(512 kbps, 512 kbps), (256 kbps, 4 Mbps), (1.5 Mbps, 8 Mbps), (3 Mbps, 10 Mbps)
II	Internet streaming session among residential participants with basic service	(1 Mbps)	(128 kbps, 1 Mbps), (128 kbps, 1 Mbps), (128 kbps, 1 Mbps), (128 kbps, 512 kbps), (64 kbps, 128 kbps), (64 kbps, 128 kbps)
III	Intranet broadcast session within corporate networks	(3 Mbps)	(1.5 Mbps, 256 kbps), (1.5 Mbps, .512 kbps), (1.5 Mbps, 4 Mbps), (1.5 Mbps, 4 Mbps)
IV	Internet streaming session among residential participants with higher grade of service	(1.5 Mbps)	(256 kbps, 1.5 Mbps), (256 kbps, 1.5 Mbps), (256 kbps, 1.5 Mbps), (256 kbps, 1.5 Mbps), (512 kbps, 512 kbps)

of bitplanes, and size of each bitplane in the enhancement layer. This information is used to estimate the parameters of the linear R-D model.

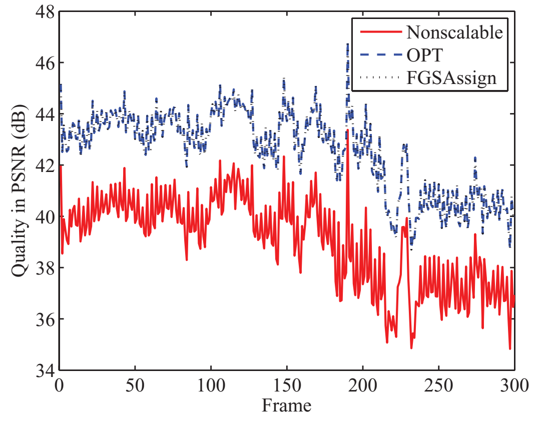
For the single-frame bit allocation problem, we have implemented the FGSAssign algorithm (Figure 7) and the optimal algorithm (referred to as OPT) using the Simplex method. For the multiple-frame case, we have implemented the mFGSAssign algorithm (Figure 8) and the optimal algorithm (referred to as mOPT) using the Simplex method as well. For baseline comparisons, we have also implemented a nonscalable allocation algorithm, in which only senders storing the same quality stream can contribute to a streaming session. In addition, the nonscalable algorithm does not allow for partial quality, and therefore, the receiver incoming bandwidth as well as the aggregate rate from senders should be large enough to carry out the streaming session. All allocation algorithms are implemented in Matlab, and the code is available to the research community [Web Page of Network Systems Lab 2006]. We run the experiments on a 3.0-GHz Pentium 4 workstation running Windows XP.

6.1.2 *Streaming Scenarios and Video Test Sequences.* To compare the performance of our allocation algorithms, we design four representative streaming scenarios, which we believe capture various Internet and Intranet streaming settings. Table III summarizes these scenarios. In the first scenario, we consider a receiver using a high-speed connection with enough incoming bandwidth to receive full quality stream, and four senders with 512-kbps, 256-kbps, 1.5-Mbps, and 3-Mbps outgoing bandwidth. The two senders with lower outgoing bandwidth represent peers with cable modem or ADSL connections, while the other two could be office or campus workstations. Due to the asymmetry between incoming and outgoing bandwidth, peers typically receive video streams with higher bit rates (quality) than they could serve to others. Therefore, we assume that the four senders store different version of the FGS-encoded stream: 512 kbps, 4 Mbps, 8 Mbps, and 10 Mbps, respectively.

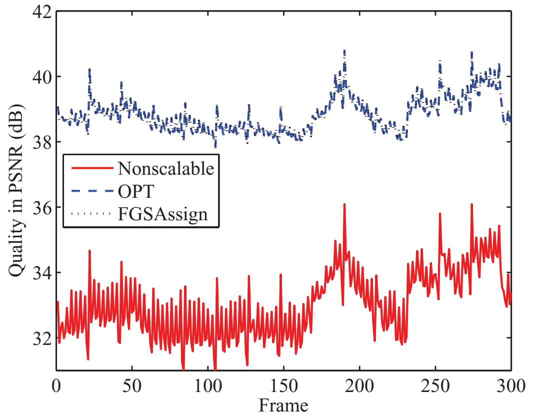
The second scenario assumes that the receiver has 1 Mbps incoming bandwidth, and there are six senders. Four of them subscribe to 128-kbps uplink access service while the other two have 64-kbps uplinks. Among those four 128-kbps senders, three of them store 1-Mbps version of the coded stream, and the last one stores a 512-kbps version. All others have a 128-kbps version of the stream. The third scenario simulates an Intranet video casting that is very common nowadays. Suppose there are four senders at different facilities, each of them has 1.5-Mbps connection. The receiver, located at a regional center, has larger bandwidth of 3 Mbps. Senders store four different versions of the stream: 256 kbps, 512 kbps, 4 Mbps, and 4 Mbps, respectively. The forth scenario has five senders: four of them have 256 kbps outgoing bandwidth and store a 1.5-Mbps coded stream. The last sender has 512 kbps outgoing bandwidth and holds a 512-kbps coded version of the stream. The receiver has 1.5-Mbps incoming bandwidth.



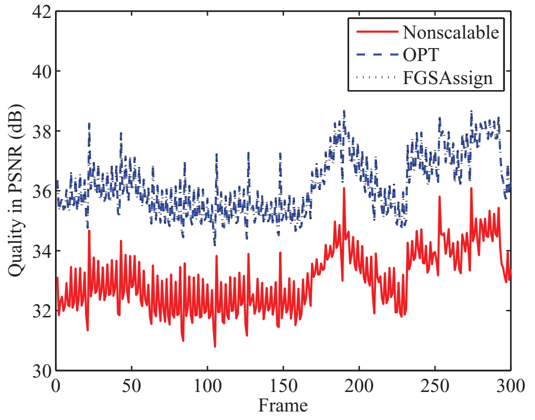
(a) Foreman sequence, scenario I



(b) Foreman sequence, scenario III



(c) Mobile sequence, scenario I



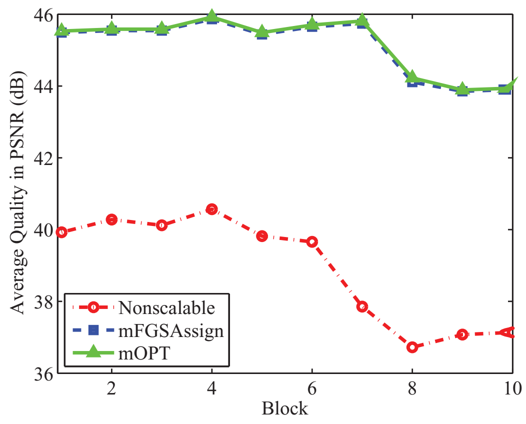
(d) Mobile sequence, scenario III

Fig. 10. Potential quality improvement by using scalable (OPT and FGSAssign) over nonscalable algorithms. Data shown for streaming Foreman sequence in scenarios I, III (a, b), and for streaming Mobile sequence in scenarios I, III (c, d).

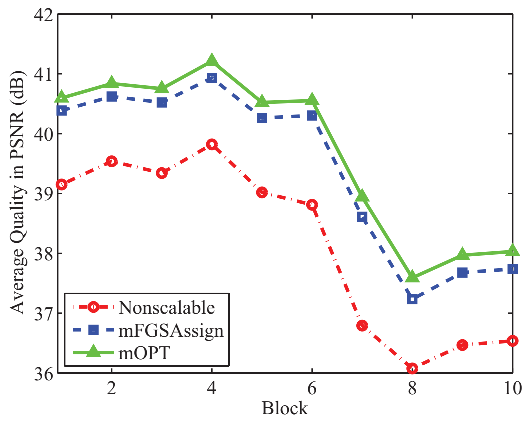
In all scenarios, we stream a set of video sequences with different characteristics. To form this set of test sequences, we have analyzed twenty representative video sequences from various sources. We categorize sequences into three complexity classes based on the average spatial and temporal complexities of each sequence [Hsu and Hefeeda 2006a]. We present results for two sequences: Foreman and Mobile. Both are coded at 30 fps, and have CIF (352x288) resolution. Foreman sequence is a low-complexity sequence and has a scene cut, where the second scene has more details than the first one. Mobile contains saturated colors and several moving objects, and therefore is a high-complexity sequence.

6.2 Results

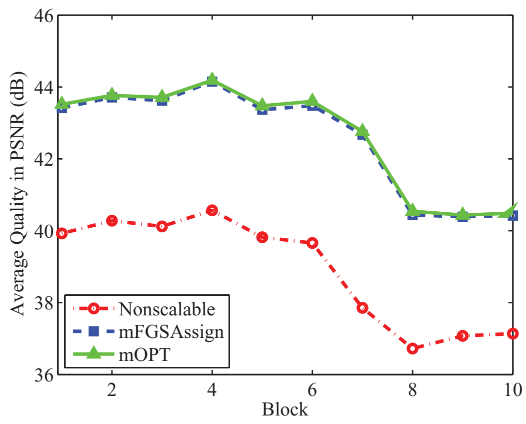
6.2.1 *Scalable Versus Nonscalable Allocation.* Our results clearly confirm the potential quality improvement from using scalable allocation algorithms over the nonscalable ones: In all considered streaming scenarios and with all test sequences, there was at least 1 dB and up to 8 dB improvement in quality. Figure 10 shows sample plots for streaming of different sequences in various scenarios, where we see an average quality improvement of at least 4 dB.



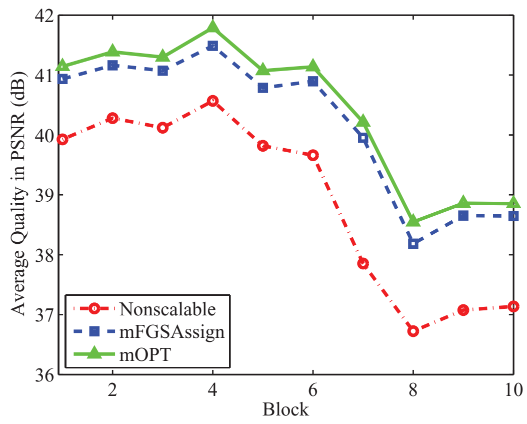
(a) Foreman sequence, scenario I



(b) Foreman sequence, scenario II



(c) Foreman sequence, scenario III

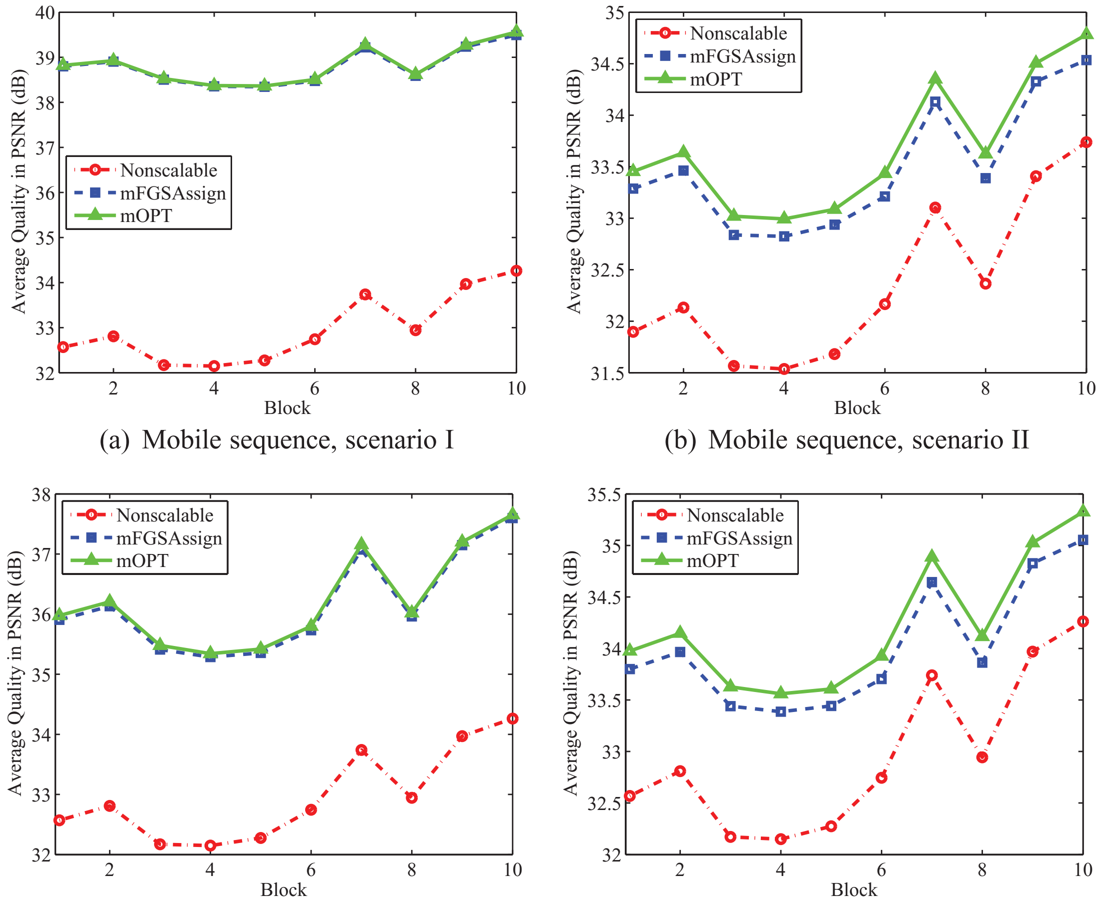


(d) Foreman sequence, scenario IV

Fig. 11. Potential quality improvement achieved by scalable (mOPT and mFGSAssign) and nonscalable algorithms for the multiple-frame bit allocation problem. Data shown for streaming Foreman sequence in different scenarios.

We use scenario I with Foreman sequence (Figure 10(a)), as an example to illustrate the reasons behind the poor performance of the nonscalable allocation. In this scenario, the allocation solution is not bounded by the receiver incoming bandwidth. We also notice that the aggregate bandwidth from the senders is more than 5 Mbps. Nevertheless, the nonscalable allocation chooses the 512-kbps coded stream, because it is the only version that can be streamed in nonscalable manner and meets all the constraints in scenario I. Notice that the nonscalable algorithm could not leverage collaboration from other senders, because they carry different versions of the stream. The results for other scenarios with different sequences are given in Hsu and Hefeeda [2006b].

6.2.2 *Optimality of FGSAssign.* To verify our analytic results regarding the optimality of the FGSAssign, we compare the distortion achieved by solving the linear programming problem using the Simplex method versus the distortion achieved by the FGSAssign algorithm. As shown in Figure 10, FGSAssign produced exactly the same distortion values as the optimal solution in all cases.

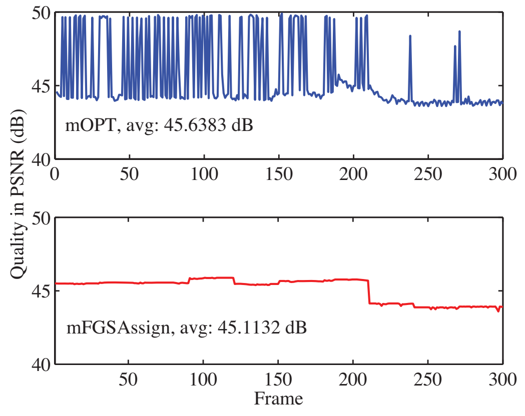


(a) Mobile sequence, scenario I
(b) Mobile sequence, scenario II
(c) Mobile sequence, scenario III
(d) Mobile sequence, scenario IV

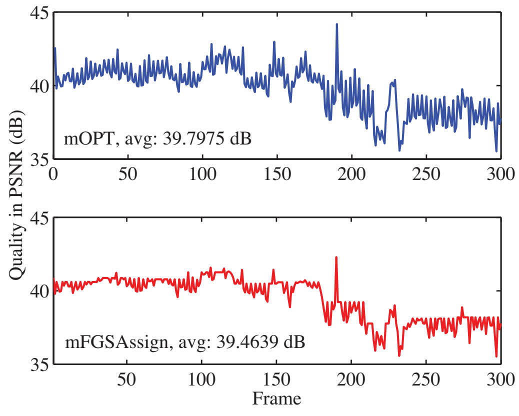
Fig. 12. Potential quality improvement achieved by scalable (mOPT and mFGSAssign) over nonscalable algorithms for the multiple-frame bit allocation problem. Data shown for streaming Mobile sequence in different scenarios.

6.2.3 *Performance of mFGSAssign versus mOPT.* Next we consider the potential quality improvement when solving the bit allocation problem for blocks of multiple frames. In Figure 11, we plot the average quality achieved by mOPT, mFGSAssign and Nonscalable algorithms for streaming Foreman sequence in all considered scenarios. We give similar plots in Figure 12 for streaming Mobile sequence. A few observations can be made from these figures. First, the quality improvement of scalable (mOPT and mFGSAssign) algorithms over the nonscalable one is clear, and, on average, it is larger than the case for the single frame allocation. This is because of the flexibility of allocating bits among frames of the same block. The second observation is that mFGSAssign achieves total distortion values that are very close to those achieved by mOPT: In all cases that we tested, the difference between mFGSAssign and mOPT was less than 1%.

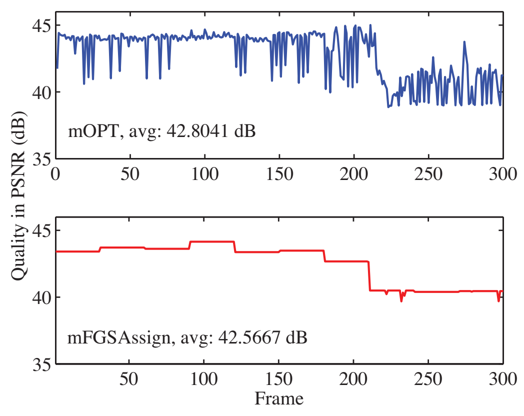
As discussed in Section 5.2, our mFGSAssign strives to achieve constant quality among consecutive frames, in addition to minimizing the total distortion. This is in contrast to mOPT which has the sole objective of minimizing the total distortion. We plot in Figures 13 and 14 the quality (in PSNR) of individual frames for mOPT (upper subfigures) and mFGSAssign (lower subfigures) for streaming



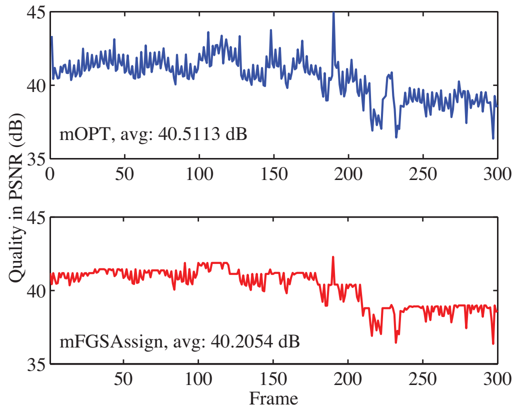
(a) Foreman sequence, scenario I



(b) Foreman sequence, scenario II



(c) Foreman sequence, scenario III

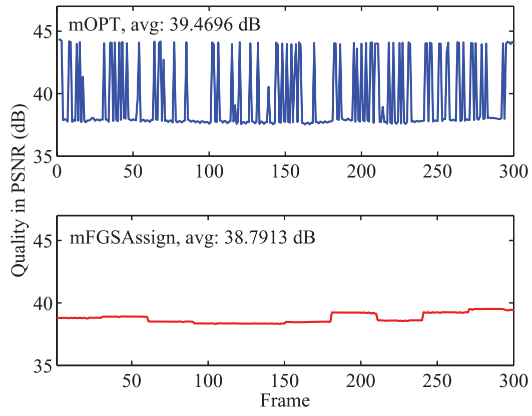


(d) Foreman sequence, scenario IV

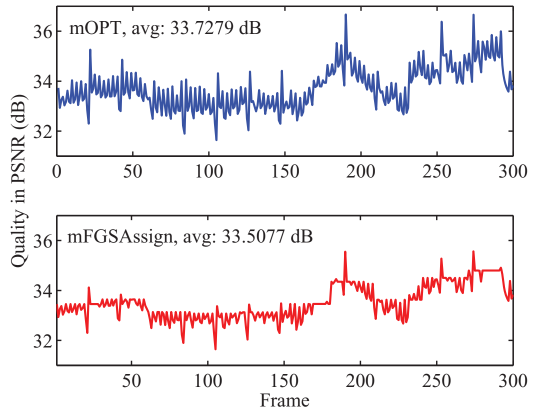
Fig. 13. Quality fluctuations between successive frames for the mOPT (top) and mFGSAssign (bottom) algorithms. Smoother quality is achieved by mFGSAssign. Data shown for streaming Foreman sequence in different scenarios. Block size $m = 30$ frames.

Foreman and Mobile sequence, respectively. These figures show that mFGSAssign achieves a much smaller quality fluctuation than mOPT. Notice also that mFGSAssign does not sacrifice the total distortion: the average quality values (shown in the figures) achieved by mFGSAssign are very close to those of the mOPT.

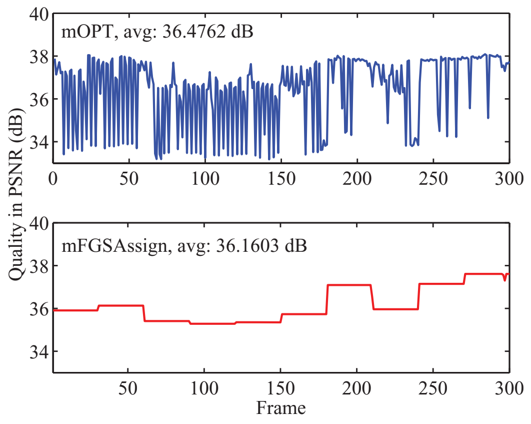
6.2.4 *Running Times.* We plot the minimum, average, and maximum per-frame running times for executing mFGSAssign and mOPT for different block sizes in Figure 15 for Foreman sequence in two scenarios. Similar results for other sequences and scenarios are given in Hsu and Hefeeda [2006b]. The figure illustrates the efficiency of the mFGSAssign algorithm versus the mOPT algorithm, which uses the highly-optimized Matlab implementation for solving LP problems. For instance, on average, mFGSAssign optimizes a block size of 100 frames in 0.3 seconds (i.e., 3 ms per frame). The mOPT algorithm takes more than 2.5 seconds (i.e., 25 ms per frame) for the same allocation problem, which is an order of magnitude slower than mFGSAssign. We also observe that the average running time of mFGSAssign remains almost constant as the block size m increases, which indicates better scalability. This is in contrast to the running time of mOPT, which increases dramatically as m increases. This is



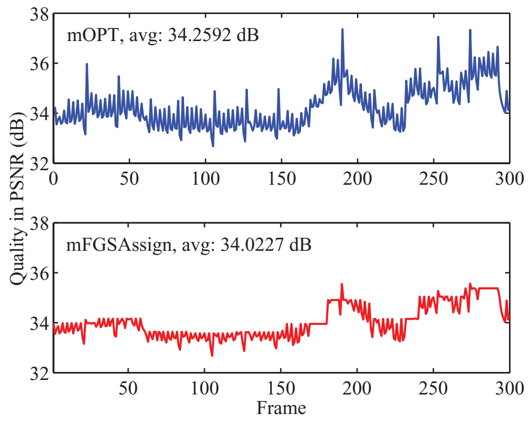
(a) Mobile sequence, scenario I



(b) Mobile sequence, scenario II



(c) Mobile sequence, scenario III



(d) Mobile sequence, scenario IV

Fig. 14. Quality fluctuations between successive frames for the mOPT (top) and mFGSAssign (bottom) algorithms. Smoother quality is achieved by mFGSAssign. Data shown for streaming Mobile sequence in different scenarios. Block size $m = 30$ frames.

because, as m increases, the number of constraints in the LP problem increases, and the efficiency of the Simplex method starts to degrade.

In addition, the variance in running times of our mFGSAssign algorithm is very small, while it is huge for mOPT. This variance is depicted by the range between minimum and maximum values in Figure 15. Stable and small running times are important for real-time applications. The results in Figure 15 confirm the applicability of our mFGSAssign algorithm in real time environments.

6.2.5 *Effect of Block Size on Video Quality.* We vary the block size m from 1 to 150 frames, and run the mFGSAssign and mOPT algorithm. We present a sample plot for streaming of Foreman sequence under scenario III in Figure 16. This figure shows that larger m values provide mFGSAssign more opportunities to approach constant quality: An almost constant quality is achieved for a reasonable block size of $m = 30$ frames. On the other hand, larger m values enable mOPT to reduce the total distortion. However, because mOPT does not consider quality fluctuations, it may actually yield wild fluctuations as m increases, which are not desirable in streaming systems. Similar results obtained for other sequences and scenarios are shown in Hsu and Hefeeda [2006b].

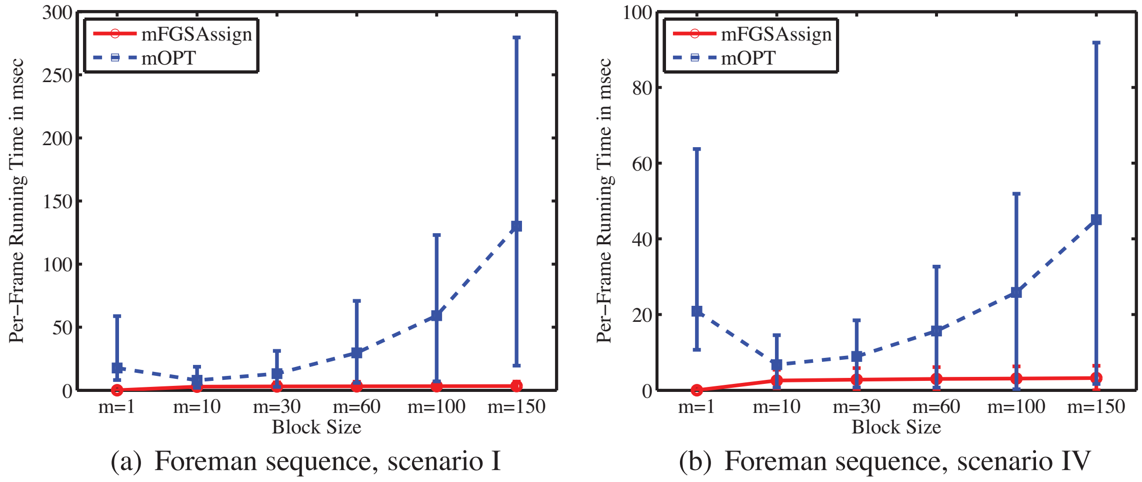


Fig. 15. The minimum, average, and maximum running times for the mOPT and mFGSAssign algorithms. Our mFGSAssign algorithm achieves much smaller and more stable running times than those of mOPT, hence it is more suitable for real-time environments. Data shown for streaming Foreman sequence in scenario I (a), and in scenario IV (b).

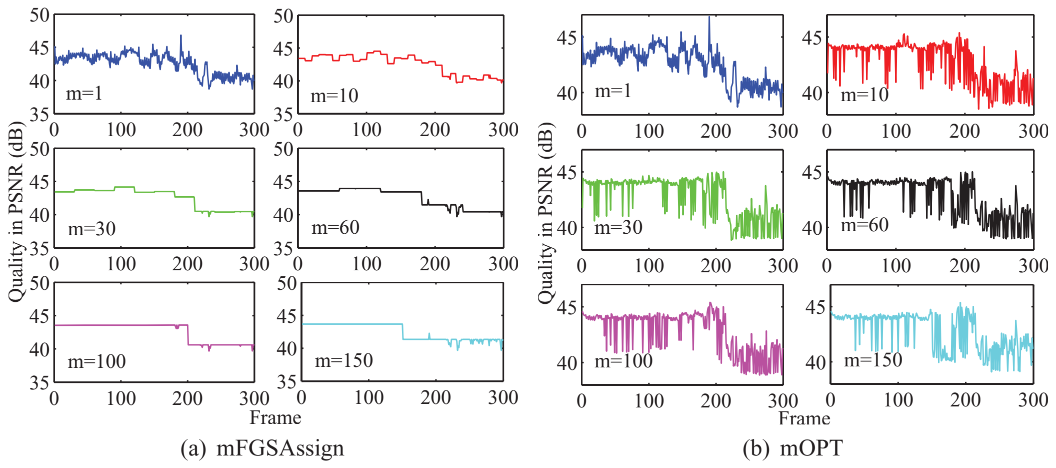


Fig. 16. Effect of block size m on quality fluctuations between successive frames. Data shown for streaming Foreman sequence in scenario III using our algorithm (a), and mOPT (b).

7. CONCLUSION

In this article, we formulated and solved the bit allocation problem for FGS-encoded video sequences streamed in distributed and heterogeneous environments. The formulation was done in a number of steps. First a general optimization problem was formed, which is transformed to an integer linear programming (ILP) one. Then, using a simple rounding scheme, the ILP problem is transformed to a linear programming problem. We proposed an optimal allocation algorithm (FGSAssign) for the single-frame case, and an efficient heuristic algorithm (mFGSAssign) for the multiple-frame case.

Our experimental results show that solving the bit allocation problem using our algorithms could provide significant quality improvement over solving it using the nonscalable algorithm: An improvement of up to 8 dB could be achieved in some cases. The quality improvement is even higher when allocating bits to blocks of multiple frames. This is because multiple frame allocation provides more chances for optimization. Furthermore, our experiments indicate that increasing the block size enables our mFGSAssign algorithm to reduce quality fluctuations in successive frames, while at the same time it decreases the total distortion. Finally, our experiments confirm that the mFGSAssign algorithm achieves its three design goals: (i) constant quality in consecutive frames, (ii) near-optimal total distortion values, and (iii) small running times.

At a high level, the results presented in this article advocate the use of fine-grained scalable encoding in streaming systems with heterogeneous receivers for at least three reasons. First, customization of an FGS-encoded sequence for different clients is easy, a mere bit truncation process, which results in reduction in CPU load for busy servers. Second, streaming servers can maintain only a single copy of the stream for all possible clients, resulting in significant storage savings for large-scale servers. Third, using our algorithms with FGS-encoded sequences, streaming systems can efficiently utilize the entire available resources (e.g., bandwidth) to yield the best possible video quality for heterogeneous receivers.

REFERENCES

BEGEN, A., ALTUNBASAK, Y., AND BEGEN, M. 2003. Rate-distortion optimized on-demand media streaming with server diversity. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'03)* (Barcelona, Spain). IEEE Computer Society Press, Los Alamitos, CA.

CHAKARESKI, J. AND GIROD, B. 2003. Rate-distortion optimized packet scheduling and routing for media streaming with path diversity. In *Proceedings of the Data Compression Conference (DCC'03)* (Snowbird, UT).

CHOU, P. AND MIAO, Z. 2006. Rate-distortion optimized streaming of packetized media. *IEEE Trans. Multimed. 8,* 2 (Apr.), 390–404.

CORMEN, T., LEISERSON, C., RIVEST, R., AND STEIN, C. 2001. *Introduction to Algorithms*, 2nd ed. MIT Press, Cambridge, MA.

CUI, Y. AND NAHRSTEDT, K. 2003. Layered peer-to-peer streaming. In *Proceedings of the ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'03)* (Monterey, CA). ACM, New York.

DAI, M. AND LOGUINOV, D. 2003. Analysis of rate-distortion functions and congestion control in scalable Internet video streaming. In *Proceedings of the ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'03)*. (Monterey, CA). ACM, New York.

DAI, M., LOGUINOV, D., AND RADHA, H. 2004. Rate-distortion modeling of scalable video coders. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'04)* (Singapore). IEEE Computer Society Press, Los Alamitos, CA.

DE CUETOS, P., SEELING, P., REISSLEIN, M., AND ROSS, K. 2005. Comparing the streaming of FGS encoded video at different aggregation levels: frame, GoP, and scene. *Int. J. Commun. Syst. 18,* 5 (June), 449–464.

GOLDFARB, D. AND TODD, M. 1989. Linear programming. *Handbook Oper. Res. Manage. Sci. Optimiz. 1*, 73–170.

HEFEEDA, M., HABIB, A., XU, D., BHARGAVA, B., AND BOTEV, B. 2005. CollectCast: A peer-to-peer service for media streaming. *ACM/Springer Multimed. Syst. J. 11,* 1 (Nov.), 68–81.

HSU, C. AND HEFEEDA, M. 2006a. On the accuracy and complexity of rate-distortion models for fine-grained scalable video sequences. Tech. Rep. TR 2006-12, Simon Fraser University. (Aug.). (Available online at http://nsl.cs.surrey.sfu.ca/projects/fgs/.)

HSU, C. AND HEFEEDA, M. 2006b. Optimal bit allocation for fine-grained scalable video sequences in distributed streaming environments. Tech. Rep. TR 2006-20, Simon Fraser University. (July). (Available online at http://nsl.cs.surrey.sfu.ca/projects/fgs/.)

ISO/IEC 14496-2. 2004. *Coding of audio-visual objects - Part 2: Visual*.

ISO/IEC 14496-5. 2004. *MPEG-4 Visual reference software*.

LI, W. 2001. Overview of fine granularity scalability in MPEG-4 video standard. *IEEE Trans. Circ. Syst. Video Tech. 11,* 3 (Mar.), 301–317.

MAGHAREI, N. AND REJAIE, R. 2006. Adaptive receiver-driven streaming from multiple senders. *ACM/Springer Multimed. Syst. J. 11,* 6 (Apr.), 1–18.

MARLER, R. AND ARORA, J. 2004. Survey of multi-objective optimization methods for engineering. *Struct. Multidisc. Optimiz. 26,* 6 (Apr.), 369–395.

NGUYEN, T. AND ZAKHOR, A. 2002. Distributed video streaming over Internet. In *Proceedings of the Multimedia Computing and Networking (MMCN'02)* (San Jose, CA).

PAPADIMITRIOU, C. AND STEIGLITZ, K. 1998. *Combinatorial Optimization: Algorithms and Complexity*, 1st ed. Dover.

RADHA, H., VAN DER SCHAAR, M., AND CHEN, Y. 2001. The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP. *IEEE Trans. Multimed. 3,* 1 (Mar.), 53–68.

SCHWARZ, H., MARPE, D., AND WIEGAND, T. 2006. The scalable H.264/MPEG4-AVC extension: Technology and applications. In *Proceedings of the European Symposium on Mobile Media Delivery (EuMob'06)* (Sardinia, Italy).

SERMADEVI, Y. AND HEMAMI, S. 2003. Linear programming optimization for video coding under multiple constraints. In *Proceedings of the Data Compression Conference (DCC'03)* (Snowbird, UT).

SU, X. AND WANG, T. 2006. Sequence of linear programming for transmission of fine-scalable coded content in bandwidth-limited environments. *ACM/Springer Multimed. Syst. J. 11,* 5 (June), 455–466.

SUN, J., GAO, W., ZHAO, D., AND HUANG, Q. 2005. Statistical model, analysis and approximation of rate-distortion function in MPEG-4 FGS videos. In *Proceedings of the SPIE International Conference on Visual Communication and Image Processing (VCIP'05)* (Beijing, China).

WANG, Y., OSTERMANN, J., AND ZHANG, Y. 2002. *Video Processing and Communications*. Prentice Hall, Englewood Cliffs, NJ.

WEB PAGE OF NETWORK SYSTEMS LAB. 2006. http://nsl.cs.surrey.sfu.ca/projects/fgs/.

ZHANG, X., VETRO, A., SHI, Y., AND SUN, H. 2003. Constant quality constrained rate allocation for FGS-coded video. *IEEE Trans. Circ. Syst. Video Tech. 13,* 2 (Feb.), 121–130.

ZINK, M., KÜNZEL, O., SCHMITT, J., AND STEINMETZ, R. 2003. Subjective impression of variations in layer encoded videos. In *Proceedings of the the IEEE International Workshop on Quality of Service (IWQoS'03)* (Monterey, CA).

Received November 2006; revised February 2007 and May 2007; accepted June 2007