Optimal Bit Allocation for Fine-Grained Scalable Video Sequences in Distributed Streaming Environments

ChengHsin Hsu and Mohamed Hefeeda

School of Computing Science

Simon Fraser University

Surrey, Canada

{cha16, mhefeeda}@cs.sfu.ca

Technical Report: TR 2006-20

Abstract

We present optimal schemes for allocating bits of fine-grained scalable video sequences among multiple senders streaming to a single receiver. The senders 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 with an objective function to minimize the distortion. By using the piece-wise linear rate-distortion model, we transform the general (nonlinear) optimization problem into an integer linear programming (ILP) problem. We further design a simple rounding scheme that transforms the ILP problem into 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 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. Our extensive experimental study shows the effectiveness of our allocation algorithms in improving the video playout quality.

I. INTRODUCTION

Video streaming over the Internet is increasingly getting very popular. In this paper, we consider on-demand video streaming systems in which a streaming session has multiple senders and a single receiver. Multiple senders may be required in peer-to-peer streaming environments [1]–[3], because of the limited capacity and unreliability of peers. Multiple senders are also desired in distributed streaming systems [4] 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 number 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:

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

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. We provide an overview of FGS encoding in Section II-A.

The FGS flexibility 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 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. Finally, 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.

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 II-B). Notice that 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 [5]. 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. (Figure 2(b) shows that $\sum_{i=1}^3 D_i > \sum_{i=1}^3 D'_i$, while $|D_i - D_{i+1}| > |D'_i - D'_{i+1}|$ for both $i = 1, 2$). Therefore, solving the bit allocation problem at the block level optimizes the playout quality by: (i) getting the maximum number of bits from senders, and then (ii) carefully distributing these bits among frames using rate-distortion curves.

In this paper, 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. First, 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 into an integer linear programming (ILP) problem. We further design a simple rounding scheme that transforms the ILP problem into 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. Second, 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. Third, we generalize the LP formulation to the multiple-frame allocation problem and we solve it using the Simplex method. Finally, 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.

The rest of this paper is organized as follows. Section II presents an overview of FGS coding and validates the rate-distortion model adopted in this paper. In Section III, we formulate the single- and multiple-frame allocation problems, and we present our rounding scheme. In Section IV, we present the

(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.

Fig. 3. The block diagram of an FGS encoder (enhancement layer only).

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 V. We discuss related work in Section VI, while Section VII concludes the paper.

II. FGS AND R-D MODELS

In this section, we first provide an overview of fine granularity scalability coding. Then, we present and validate the rate-distortion model used in this paper.

A. Fine Granularity Scalability Coding

Fine granularity scalability (FGS) has been proposed as part of the MPEG-4 standard to improve rate scalability and error resilience [6], [7]. An FGS-encoded sequence consists of two streams: a non-scalable base layer which provides basic quality, and an enhancement layer that adds incremental quality refinements proportional to the number of bits received. In scalable streaming systems, the base layer is typically encoded at a low bit rate to accommodate a wide range of heterogeneous receivers. The full enhancement layer (i.e., the one that provides full quality) is usually very large and requires high streaming bit rates. Furthermore, due to its small size and its critical role, the base layer is assumed to

TABLE I

BITPLANE CODING OF THE FIRST 16 DCT COEFFICIENTS OF THE FIRST BLOCK OF FRAME 1 IN AKIYO SEQUENCE.

DCT coefficient	-2	-2	5	-4	2	-1	-1	2	-2	2	-3	2	-2	-1	-1	-2	...	0
Binary Representations																		
Sign	1	1	0	1	0	1	1	0	1	0	1	0	1	1	1	1	...	0
Bitplane 1 (MSB)	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	...	0
Bitplane 2	1	1	0	0	1	0	0	1	1	1	1	1	1	0	0	1	...	0
Bitplane 3	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1	0	...	0
(RUN, LAST) Symbols																		
Bitplane 1 (MSB)	(2,0)		(0,1)															
Bitplane 2	(0,0)		(0,0)		(2,0)		(2,0)		(0,0)		(0,0)		(0,0)		(0,0)		(0,0)	(2,1)
Bitplane 3	(2,0)		(2,0)		(0,0)		(3,0)		(2,0)		(0,1)							

be delivered over a reliable channel, e.g., a channel that uses forward error correction or retransmission if the round trip time is relatively small. In this paper, 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 bitrate and it will be reliably transported to the receiver.

The components of a simplified FGS encoder for the enhancement layer are shown in Figure 3. The image residue between the original and base layer reconstructed frames is first transformed into 8x8 DCT blocks. This is followed by bitplane shifting to emphasize certain DCT coefficients to improve perceptual quality. After bitplane shifting, the maximum absolute value of all coefficients is determined, which defines the number of bitplanes required for full quality. The coefficients are then zigzag scanned and bitplane coded. In bitplane coding, digits at the same significant position form a bitplane. Last, the variable length coder (VLC) run-length codes each bitplane into (RUN, LAST) symbols and then entropy codes them [6].

To illustrate bitplane coding, we present a simple example in Table I. The table shows the first 16 coefficients of the first block of frame 1 of Akiyo video sequence. For simplicity, we assume the remaining (64-16) coefficients are zeros. Since the maximum absolute value of coefficients is 5, we need 3 bitplanes to code them. In the table, the binary representation of each coefficient is vertically aligned under that coefficient. Each horizontal row (a bitplane) is composed of bits at the the same significant position from different coefficients. Bitplanes are then separately run-length coded into (RUN, LAST) symbols and

entropy coded (not shown in the table) to form the final enhancement layer stream.

Two important observations can be made from the simple example in Table I. First, the order of the bitplanes reflects their importance in terms of their relative contributions to the quality. Bitplane 1 contributes the most because it contains the most significant bit of each coefficient. Bitplane 2 contributes less than bitplane 1 but more than bitplane 3, and so on. To illustrate, consider what happens if we were to truncate or corrupt one of the bitplanes and focus on the impact of this truncation on the 3rd coefficient from the left (with value 5) in the table. If we truncate bitplane 3, the reconstructed value of the 3rd coefficient will be 4. Whereas if bitplane 1 is removed, the reconstructed value of the coefficient will be 1, which would produce more distortion than a reconstructed value of 4. This observation indicates that the per-bit reduction in distortion decreases as we move from the most significant bitplane to the least significant one.

The second observation from Table I is that the size of bitplane 1 is usually very small. This is in fact the typical case in bitplane coding, which can be explained by inspecting the probability distribution of DCT coefficients. A sample DCT coefficient distribution is shown in Figure 4.¹ The figure shows that the majority of coefficients are centered around zero, i.e., most of them have small values. Since many of the most significant bits of small values are zeros, the most significant bitplanes will have a few 1's, and in turn a few (RUN, LAST) symbols. In addition, recall that the zigzag scanning gives higher precedence to low frequency coefficients, which generally are larger than high frequency ones. This implies that the few 1's in the most significant bitplanes will appear at the head of each block, thus, majority of the (RUN, LAST) symbols have small jumps (i.e., RUN values). This probability bias further increases the coding efficiency of the entropy coder and results in the small bitplane size.

B. 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, e.g., the square-root model [9], the logarithm model, [10], and the Generalized Gaussian model [11]. In our previous work, we have compared the accuracy and complexity of the current R-D models in the literature [8]. In each one of these analytic models, the distortion is related to the rate with a *nonlinear*

¹Interested reader is referred to [8] for rigorous analysis of DCT coefficients of video sequences and the various distributions used to model them.

Fig. 4. A sample distribution of DCT coefficients in the enhancement layer. Data shown for frame 1 of Akiyo sequence.

function. Nonlinear R-D functions make our bit allocation problem (described in Section III) a nonlinear optimization problem, which is hard (if at all feasible) to solve. Moreover, the methods used to estimate parameters of the analytic R-D models are computationally expensive as indicated by the experimental study in [8], some methods take minutes (up to half an hour in some cases!) to estimate parameters of a *single* frame.

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 again computationally expensive [12, page 302]. A yet 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 piece-wise linear R-D model [13], which we adopt in this paper, falls into this category.

In the following subsections, 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.

1) The Piece-wise Linear R-D Model: The key idea of the piece-wise linear model [13] (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 5 shows an example. The problem of finding the R-D function is now reduced to measuring the distortion at bit rates that

Fig. 5. The piece-wise linear R-D model for frame 150 of Akiyo sequence.

TABLE II

THE R-D MODEL PARAMETERS FOR THE FIRST 12 FRAMES OF FOREMAN SEQUENCE.

Frame	d	bp 1		bp 2		bp 3		bp 4		bp 5		bp 6	
		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

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

correspond to bitplane boundaries (up to 8 samples in most cases), and computing the slopes of the different line segments.

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 II shows the linear R-D model parameters for the first twelve frames of Foreman sequence.

2) *Accuracy of the Linear R-D Model:* Since the accuracy of the adopted R-D model is crucial to our problem, we should validate the accuracy of the linear R-D model. To do so, 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 to the original and reconstructed sequences. 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. Figure 6 illustrates 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, shown in Figure 7, confirm the accuracy of the linear R-D model: the average error is less than 2% in almost all cases.

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, $g_1 < g_2 < \dots < g_z < 0$, where g_h is the slope of line segment in bitplane h (see the three leftmost subfigures in Figure 8). This is intuitive because, as we described in Section II-A, bits in the more significant bitplanes carry higher significant bits, and thus the per-bit reduction in distortion is higher. This means that the most significant bitplanes will have steeper and more negative slopes (refer to Figure 6).

While the above explanation of slope monotonicity makes sense, we have seen a few anomalies. For

(a) Akiyo sequence, frame 125

(b) Mother sequence, frame 110

(c) Foreman sequence, frame 100

(d) Mobile sequence, frame 90

Fig. 6. The accuracy of the linear R-D model for randomly chosen frames from different sequences.

instance, frame 2 of Foreman (see Table II) 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 II. 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 II 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: aggregate any two neighboring bitplanes into a single bitplane whenever there is a violation in monotonicity. For instance,

Fig. 7. The accuracy of the linear R-D model. The figure shows the average error between curves estimated by the linear model and the actual R-D curves.

Fig. 8. The monotonicity property of the linear R-D model. 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.

TABLE III
THE FREQUENCY OF SLOPE EXCEPTIONS $g_i \geq g_{i+1}$ IN FOUR SEQUENCES.

	Number of frames	Number of exceptions	Number of total bitplanes	Post-aggregation exceptions
Akiyo	300	86	1189	0
Mother	300	92	1269	0
Foreman	300	129	1523	2
Mobile	300	163	1753	0

the rightmost subfigure in Figure 8 shows the series of 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 the slopes of all bitplanes of all frames in four video sequences: Akiyo, Mother, Foreman, and Mobile. Table III shows the results, where the column labeled post-aggregation exceptions refer to the number of violations in the monotonicity property remained after aggregating bitplanes 1 and 2. As the table shows, 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 repeated the accuracy assessment experiments in the previous subsection, but using linear R-D curves after bitplane aggregation was done on them. Figure 9 shows that the average error between the modified linear R-D curves and the actual ones is visually the same as the (untransformed) linear R-D curves (Shown in Figure 7). Statistically, the average difference caused by the bitplane aggregation is negligible, as we observe 0.0193% difference for Akiyo sequence, 0.0340% for Mother, 0.0137% for Foreman, and 0.0116% for Mobile.

III. BIT ALLOCATION PROBLEM FORMULATION AND SOLUTION

In this section, we consider bit allocation problems for P2P video streaming systems on FGS encoded streams. We assume the base layer is coded at reasonably low bit rate and can be transmitted through secure channels, e.g., forward error correction (FEC) protected channels. 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.

Fig. 9. 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.

A. Problem Formulation

The bit allocation problem we address in this paper can be stated as follows. Given multiple senders that can potentially serve a specific FGS-encoded video sequence to a given receiver. Senders may have different portions of the sequence and may have different outgoing bandwidth. The receiver has a limit on the incoming bandwidth. Determine the streaming rate and the number of bits allocated to each sender to achieve the best possible video quality at receiver. We assume that the base layer is coded at a reasonably low (and fixed) bit rate and can be reliably delivered by any sender in the set of potential senders, or even by a sender outside this set. 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

TABLE IV
NOTATIONS USED IN THIS PAPER

Notation	Definition
n	Number of potential senders.
M	Number of frames in a sequence.
m	Number of frames in a block.
T	Frame period.
b_i	Outgoing bandwidth of sender i .
b_I	Incoming bandwidth of the receiver.
s_i	Size of the bit stream stored at sender i .
Δ_i	Number of bits allocated to sender i .
r_i	Sending rate allocated to sender i .
z	Number of bitplanes in the enhancement layer.
l_h	Size of bitplane h .
d	Distortion when no enhancement layer bits are sent.
g_h	Slope of the line segment in bitplane h .
y_h	Number of bits transmitted from bitplane h .

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 IV summarizes all the notations used in this paper.

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 ($i = 1, 2, \dots, n$). Furthermore, 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 $s_1 \leq s_2 \leq \dots \leq s_n$, 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 $i = 1, 2, \dots, n-1$. Finally, the receiver incoming bandwidth is denoted by b_I .

Solving the allocation problem should yield an allocation policy $A = \{(\Delta_i, r_i) | i = 1, 2, \dots, n\}$, 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:

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

$$\text{s.t.} \quad \Delta_i - r_i T \leq 0 \quad (1b)$$

$$\sum_{t=1}^i \Delta_t \leq s_i \quad (1c)$$

$$r_i \leq b_i \quad (1d)$$

$$\sum_{t=1}^n r_t \leq b_I \quad (1e)$$

$$\Delta_i, r_i \in \mathbb{N}; \quad i = 1, 2, \dots, n. \quad (1f)$$

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 ($\Delta_i/r_i \leq T \implies \Delta_i - r_i T \leq 0$). (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 II-B, 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 5). 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 ($h = 1, 2, \dots, z$) 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 re-written as:

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

$$\text{s.t.} \quad \Delta_i - r_i T \leq 0 \quad (2b)$$

$$\sum_{t=1}^i \Delta_t \leq s_i \quad (2c)$$

$$r_i \leq b_i \quad (2d)$$

$$\sum_{t=1}^n r_t \leq b_I \quad (2e)$$

$$y_h \leq l_h \quad (2f)$$

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

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

Notice the two new constraints 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, just finding a feasible solution to it is NP-hard, hence, there is no efficient ILP algorithm [14, page 777]. In the next subsection, we present a rounding scheme that transforms the ILP problem in (2) to a linear programming (LP) problem, which could 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, 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 $h, j = 1, 2, \dots, z$ 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 $j = 2, 3, \dots, z$, then it must be the case that $y_h^* = l_h$, where $h = 1, 2, \dots, j-1$, 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 $\hat{A}$ 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 . $\hat{A}$ is indeed a feasible allocation because it satisfies all the constraints in (2). Furthermore, the distortion associated with $\hat{A}$ is given by $\hat{D} = D^* + (g_h - g_j)q < D^*$, since g_h is always less than g_j because slopes are monotonically increasing, as discussed in Section II-A. This is a contradict possible distortion. ■

B. Rounding Scheme and Linear Programming Solution

In this subsection, 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:

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

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

$$\bar{r}_i = \lfloor \hat{r}_i \rfloor, \quad \text{and} \quad \bar{\Delta}_i = \begin{cases} \lfloor \hat{\Delta}_i \frac{\hat{r}_i - 1}{\hat{r}_i} \rfloor, & \hat{r}_i > 1 \\ 0, & 0 \leq \hat{r}_i \leq 1 \end{cases} \quad (4)$$

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 $\bar{\Delta}_i - \bar{r}_i T \leq 0$. Notice that all other constraints are automatically satisfied because we round down both of $\hat{r}_i$ and $\hat{\Delta}_i$.

We first consider the case when $\hat{r}_i > 1$. For any $i = 1, 2, \dots, n$, we have:

$$\frac{\bar{\Delta}_i}{\bar{r}_i} = \frac{\lfloor \hat{\Delta}_i \frac{\hat{r}_i - 1}{\hat{r}_i} \rfloor}{\lfloor \hat{r}_i \rfloor} \leq \frac{\hat{\Delta}_i \frac{\hat{r}_i - 1}{\hat{r}_i}}{\lfloor \hat{r}_i \rfloor} < \frac{\hat{\Delta}_i \frac{\hat{r}_i - 1}{\hat{r}_i}}{\hat{r}_i - 1} = \frac{\hat{\Delta}_i}{\hat{r}_i} \leq T,$$

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 $\lfloor \hat{r}_i \rfloor > \hat{r}_i - 1$. Next we consider the case when $0 \leq \hat{r}_i \leq 1$. In this case, $\bar{\Delta}_i = 0$. Therefore, $\bar{\Delta}_i - \bar{r}_i T = -\bar{r}_i T \leq 0$, because $\bar{r}_i \geq 0$, 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 $nT + n$.

Proof: We first compute the gap between the optimal and rounded solutions for an arbitrary sender i , where $i = 1, 2, \dots, n$. If $\hat{r}_i > 1$, we have:

$$\hat{\Delta}_i - \bar{\Delta}_i = \hat{\Delta}_i - \lfloor \hat{\Delta}_i \frac{\hat{r}_i - 1}{\hat{r}_i} \rfloor < \hat{\Delta}_i - \hat{\Delta}_i \frac{\hat{r}_i - 1}{\hat{r}_i} + 1 = \frac{\hat{\Delta}_i}{\hat{r}_i} + 1 \leq T + 1.$$

If $0 \leq \hat{r}_i \leq 1$, we have

$$\hat{\Delta}_i - \bar{\Delta}_i = \hat{\Delta}_i \leq \hat{r}_i T \leq T < T + 1.$$

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

To put this gap into perspective, consider an extreme case streaming session in which 30 senders are concurrently streaming to a single receiver. Assume the frame rate is 15 frames per second. Then, the gap is at most $30/15 + 30 = 32$ bits. That is, the rounded solution may transmit at most 4 bytes less than the optimal one from any given frame, which indeed is a negligible gap given that frame sizes are in the order of kilo bytes.

C. Multiple Frame Formulation

In this subsection, 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 I, 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 straightforward, and is given in (5). Notice the addition of the suffix f to the variables denote a frame within the block.

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

$$\text{s.t.} \quad \sum_{u=1}^m \Delta_{iu} - m r_i T \leq 0 \quad (5b)$$

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

$$r_{if} \leq b_i \quad (5d)$$

$$\sum_{t=1}^n r_{tf} \leq b_f \quad (5e)$$

$$y_{hf} \leq l_{hf} \quad (5f)$$

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

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

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 $O(mn)$. We examine an extreme instance in which $m = 100$ and $n = 16$, this formulation requires 3700 variables and 3916 constraints, if we assume every frame consists of 5 bitplanes. Despite modern LP solvers are able to solve problems in this scale, repeatedly solving them for every block still might result in high time complexity in practice.

IV. 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 [15, Section 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 running times, which in many cases exceed the average running time of the Simplex method [16].

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, and on average, it runs an order of magnitude faster than the optimal algorithm.

FGSAssign

```

/* Inputs:
n: number of senders.
bi: outgoing bandwidth for every sender i (i = 1, 2, ..., n).
bI: incoming bandwidth for the receiver.
si: portion of the stream cached by every sender i.
*/
/* Outputs:
ri: allocated sending rate for every sender i.
Δi: assigned stream length for every sender i.
*/
1.   Sort all senders based on si, where s1 ≤ s2 ≤ ... ≤ sn;
2.   x0 = ... = xn = 0; Δ1 = ... = Δn = 0; ragg = 0;
3.   for i = 1 to n do
4.       xi = min(xi-1 + biT, si);
5.       ri = (xi - xi-1)/T;
6.       if (ragg + ri < bI) then
7.           ragg = ragg + ri;
8.           Δi = xi - xi-1;
9.       else
10.          ri = bI - ragg;
11.          Δi = T × ri;
12.       return
13.   endfor

```

Fig. 10. Pseudo code for an optimal and efficient algorithm for the single-frame bit allocation problem.

A. FGSAssign: An Optimal Algorithm 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 10. The basic idea of the algorithm is to send the maximum number of bits from the senders. It does so by first sorting all senders based on the portion of the stream stored at each sender, such that $s_1 \leq s_2 \leq \dots \leq s_n$. Then, it sequentially allocates to each sender i ($i = 1, 2, \dots, n$) the maximum number of bits which sender i can transmit within the frame period T . By doing so, it 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 lemma proves that FGSAssign is optimal and efficient.

Lemma 4: 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.

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. Moreover, if the algorithm stops when the receiver incoming bandwidth is reached, it will allocate the maximum number of bits to senders and yield an optimal solution. We prove cases where the for-loop is finished without early termination as follows.

Every iteration i of the for-loop of the FGSAssign 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 solve a subproblem in the iteration $i + 1$. To prove this greedy approach leads to an optimal solution, we show the problem has two properties: greedy-choice and optimal substructure.

Greedy-choice property guarantees that a greedy choice can lead to a globally optimal solution. Suppose $\{(\Delta_u^*, r_u^*) | u = i, i + 1, \dots, n\}$ is an optimal solution to the original problem (iteration i). Since (Δ_i, r_i) is the greedy choice, $\Delta_i \geq \Delta_i^*$ (and $r_i \geq r_i^*$), we can construct a new solution from the optimal one by shifting responsible portions (and rate) 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. Suppose the combination of

the greedy decision $\{(\Delta_i, r_i)\}$ and a subproblem optimal solution $\{(\Delta_v, r_v)|v = i + 1, i + 2, \dots, n\}$ is not an optimal solution to the original problem. Then, we can find an optimal solution $\{(\Delta_u^*, r_u^*)|u = i, i + 1, \dots, n\}$, such that $\sum_{u=i}^n \Delta_u^* > \sum_{v=i}^n \Delta_v$. Because $\Delta_i \geq \Delta_i^*$ by greedy, we reach a contradiction $\sum_{u=i+1}^n \Delta_u^* > \sum_{v=i+1}^n \Delta_v$. ■

B. 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 the 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 III-C).² 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 [5]. Our experimental results (Section V), 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 times are significantly shorter than those of the optimal algorithm.

The pseudo code of the mFGSAssign algorithm is given in Figure 11. 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 mFGSAssign algorithm are: (i) The 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 ($f = 1, 2, \dots, m$) 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 II-B). 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 12, 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

²Formulating 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 [17].

mFGSAssign

/* Inputs:

n : number of senders.

m : number of frames in the block.

b_i : outgoing bandwidth for every sender i ($i = 1, 2, \dots, n$).

b_I : incoming bandwidth for the receiver.

s_{if} : portion of the stream cached by every sender i for each frame f ($f = 1, 2, \dots, m$) in the block.

rd_tbl : R-D table of linear model.

*/

/* Outputs:

r_{if} : allocated sending rate for every sender i for each frame f .

Δ_{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. $D_u = \text{getMaxDistortion}(rd_tbl)$; $D_l = \text{getMinDistortion}(rd_tbl)$;
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. $B_f = T \times \text{getRate}(rd_tbl, f, D)$; //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. 11. Pseudo code for a heuristic algorithm for the multiple-frame bit allocation problem.

Fig. 12. An example showing the estimation of the target distortion and the associated bit budget for a block of four frames from the Foreman sequence.

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 12 have maximum distortion values of 7.98, 4.52, 8.51, and 8.02, 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 12). $B < B^{max}$ 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 pre-defined threshold ($B^{th} \geq 0$) 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 successfully search for all practical situations. In our experimental study, we used a very small threshold $B^{th} = 1$ byte, and all 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 5: The mFGSAssign terminates in $O(mn \log n)$ 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 bitplanes (MPEG-4 standard [18] reserves 5 bits for *max_bitplanes* field, which supports up to 31 bitplanes). Thus the cost for the while loop is $O(m)$. Furthermore, the m calls to FGSAssign cost $O(mn \log n)$ steps, the lemma follows. ■

V. 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 mFGSAssign versus the optimal algorithm, (iv) the average running times of the 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.

A. Experimental Setup

Software used and developed. In our experiments, we use the MPEG-4 Reference Software Version 2.5 [19] 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 bitrate. It does so by calculating the bit budget of each frame at the target bitrate. 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 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 10) and the optimal algorithm (referred to as OPT) using the Simplex method. For the multiple-frame

case, we have implemented the mFGSAssign algorithm (Figure 11) and the optimal algorithm (referred to as mOPT) using the Simplex 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 have been large enough to carry out the streaming session. All allocation algorithms are implemented in Matlab. We run the experiments on a 3.0 GHz Pentium 4 workstation running windows XP.

Streaming scenarios and video test sequences. To compare the performance of the allocation algorithms, we design four representative streaming scenarios, which we believe capture various Internet and Intranet streaming settings. 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 senders store different versions of the FGS-encoded stream: 512 kbps, 4 Mbps, 8 Mbps, and 10 Mbps, respectively.

The second scenario assumes the receiver has 1 Mbps incoming bandwidth, and there are six senders. Four of them subscribe 128 kbps uplink access service while the other two only have 64 kbps uplink. Among those four 128 kbps senders, three of them store 1 Mbps 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 4Mbps, respectively. The fourth 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.

In each of the above 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 [8]. We present results for two sequences: Foreman and Mobile. Both are coded at 30 fps, and have a 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

Fig. 13. Comparison between OPT, FGSAssign, and Nonscalable allocation algorithms: (a) Distortion (in MSE) achieved by each algorithm, and (b) Potential quality improvement (in dB) by using scalable over nonscalable algorithms. Data shown for streaming Foreman sequence in scenario I.

colors and several moving objects, and therefore is a high-complexity sequence.

B. Results

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 13 shows a sample plot for streaming of Foreman sequence in scenario I, where we see an average quality improvement of at least 5 dB in Figure 13(b).

We use this example (scenario I with Foreman) 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 shown in Figures 14–20.

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

Fig. 14. Comparison between OPT, FGSAssign, and Nonscalable allocation algorithms: (a) Distortion (in MSE) achieved by each algorithm, and (b) Potential quality improvement (in dB) by using scalable over nonscalable algorithms. Data shown for streaming Foreman sequence in scenario II.

versus the distortion achieved by the FGSAssign algorithm. In all cases, FGSAssign produced exactly the same distortion values as the optimal (see the left subfigures in Figures 13–20).

Performance of mFGSAssign versus mOPT. Next we consider the potential quality improvement when solving the bit allocation problem for blocks of multiple frames each. In Figure 21, 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 22 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% (about 0.25 dB in absolute value). The final observation is that the quality starts to degrade for all algorithms after the 6th block in Foreman sequence. This is because of the scene change in Foreman sequence around frame number 200. The second scene is more complex, and thus using the same resources (bandwidth) results in higher distortion.

As discussed in Section IV-B, 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 23 and 24 the quality (in PSNR) of

Fig. 15. Comparison between OPT, FGSAssign, and Nonscalable allocation algorithms: (a) Distortion (in MSE) achieved by each algorithm, and (b) Potential quality improvement (in dB) by using scalable over nonscalable algorithms. Data shown for streaming Foreman sequence in scenario III.

individual frames for mOPT (upper subfigure) and mFGSAssign (lower subfigure) for streaming Foreman and Mobile sequence, respectively. These figures show that mFGSAssign achieves a much less quality variation 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.

Running Times. We plot the average running times per block for executing mFGSAssign and mOPT for different block sizes in Figures 25 and 26 for Foreman and Mobile sequence, respectively. The figures illustrate the efficiency of the mFGSAssign algorithm versus the mOPT algorithm, which uses the (highly-optimized) Matlab implementation for solving LP problems. For a block size of 100 frames, on average, mFGSAssign runs an order of magnitude faster than mOPT. 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 because, as m increases, the number of constraints in the LP problem increases, and the efficiency of the Simplex algorithm starts to degrade.

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 I in Figure 27. 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

(a) Distortion

(b) Potential Quality Improvement

Fig. 16. Comparison between OPT, FGSAAssign, and Nonscalable allocation algorithms: (a) Distortion (in MSE) achieved by each algorithm, and (b) Potential quality improvement (in dB) by using scalable over nonscalable algorithms. Data shown for streaming Foreman sequence in scenario IV.

mOPT does not consider quality fluctuations, it may actually yield wild fluctuations as m increases, which are not desirable in streaming systems. Similar results are obtained for other sequences and scenarios as shown in Figures 28–34.

Fig. 17. Comparison between OPT, FGSAssign, and Nonscalable allocation algorithms: (a) Distortion (in MSE) achieved by each algorithm, and (b) Potential quality improvement (in dB) by using scalable over nonscalable algorithms. Data shown for streaming Mobile sequence in scenario I.

Fig. 18. Comparison between OPT, FGSAssign, and Nonscalable allocation algorithms: (a) Distortion (in MSE) achieved by each algorithm, and (b) Potential quality improvement (in dB) by using scalable over nonscalable algorithms. Data shown for streaming Mobile sequence in scenario II.

Fig. 19. Comparison between OPT, FGSAssign, and Nonscalable allocation algorithms: (a) Distortion (in MSE) achieved by each algorithm, and (b) Potential quality improvement (in dB) by using scalable over nonscalable algorithms. Data shown for streaming Mobile sequence in scenario III.

Fig. 20. Comparison between OPT, FGSAssign, and Nonscalable allocation algorithms: (a) Distortion (in MSE) achieved by each algorithm, and (b) Potential quality improvement (in dB) by using scalable over nonscalable algorithms. Data shown for streaming Mobile sequence in scenario IV.

(a) Scenario I

(b) Scenario II

(c) Scenario III

(d) Scenario IV

Fig. 21. 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.

(a) Scenario I

(b) Scenario II

(c) Scenario III

(d) Scenario IV

Fig. 22. Potential quality improvement achieved by scalable (mOPT and mFGSAssign) and nonscalable algorithms for the multiple-frame bit allocation problem. Data shown for streaming Mobile sequence in different scenarios.

(a) Scenario I

(b) Scenario II

(c) Scenario III

(d) Scenario IV

Fig. 23. 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.

(a) Scenario I

(b) Scenario II

(c) Scenario III

(d) Scenario IV

Fig. 24. 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.

(a) Scenario I

(b) Scenario II

(c) Scenario III

(d) Scenario IV

Fig. 25. Average running time of mFGSAssign and mOPT for different block sizes, Foreman sequence.

(a) Scenario I

(b) Scenario II

(c) Scenario III

(d) Scenario IV

Fig. 26. Average running time of mFGSAssign and mOPT for different block sizes, Mobile sequence.

Fig. 27. The impact of varying block size m on the quality variation among consecutive frames, scenario I for Foreman sequence.

Fig. 28. The impact of varying block size m on the quality variation among consecutive frames, scenario II for Foreman sequence.

Fig. 29. The impact of varying block size m on the quality variation among consecutive frames, scenario III for Foreman sequence.

Fig. 30. The impact of varying block size m on the quality variation among consecutive frames, scenario IV for Foreman sequence.

Fig. 31. The impact of varying block size m on the quality variation among consecutive frames, scenario I for Mobile sequence.

Fig. 32. The impact of varying block size m on the quality variation among consecutive frames, scenario II for Mobile sequence.

Fig. 33. The impact of varying block size m on the quality variation among consecutive frames, scenario III for Mobile sequence.

VI. RELATED WORK

Distributed and peer-to-peer streaming has recently received significant research attention. For example, the distributed video streaming framework [4] 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 [1] to optimize quality at the receiver. Both work do not consider scalable-coded streams. Layered-scalable streams are considered in [3] and [2] to cope with the bandwidth heterogeneity. In [3], the multi-sender streaming problem is formulated to maximize the streaming quality of all peers and minimize the load on the originating media distributor. The authors of [2] design 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 consider a minimization problem for the transmission time of FGS-encoded images in [20]. They formulate the problem as a non-linear program, and then they transform it to a series of LP subproblems. Each of this LP subproblem determines a peer allocation to maximize the number of received bits in a given delay bound. Unlike our work, the work in [20] 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 [21]. This framework has been generalized by many researchers from various perspectives. For instance,

Fig. 34. The impact of varying block size m on the quality variation among consecutive frames, scenario IV for Mobile sequence.

[22] and [23] 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.

VII. CONCLUSION

In this paper, 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 also 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 chance for optimization. Furthermore, our experiments indicate that increasing the block size enables our mFGSAssign 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 time.

REFERENCES

- [1] M. Hefeeda, A. Habib, D. Xu, B. Bhargava, and B. Botev, "CollectCast: A peer-to-peer service for media streaming," *ACM/Springer Multimedia Systems Journal*, vol. 11, no. 1, pp. 68–81, November 2005.
- [2] N. Magharei and R. Rejaie, "Adaptive receiver-driven streaming from multiple senders," *ACM/Springer Multimedia Systems Journal*, vol. 11, no. 6, pp. 1–18, April 2006.
- [3] Y. Cui and K. Nahrstedt, "Layered peer-to-peer streaming," in *Proc. of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'03)*, Monterey, CA, June 2003.
- [4] T. Nguyen and A. Zakhori, "Distributed video streaming over Internet," in *Proc. of Multimedia Computing and Networking (MMCN'02)*, San Jose, CA, January 2002.
- [5] M. Zink, O. Künzel, J. Schmitt, and R. Steinmetz, "Subjective impression of variations in layer encoded videos," in *Proc. of the IEEE International Workshop on Quality of Service (IWQoS'03)*, Monterey, CA, June 2003.
- [6] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 301–317, March 2001.
- [7] H. Radha, M. V. D. Schdhar, and Y. Chen, "The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP," *IEEE Transactions on Multimedia*, vol. 3, no. 1, pp. 53–68, March 2001.
- [8] C. Hsu and M. Hefeeda, "On the accuracy and complexity of rate-distortion models for FGS-encoded video sequences," Simon Fraser University, Tech. Rep. TR 2006-12, May 2006.
- [9] M. Dai and D. Loguinov, "Analysis of rate-distortion functions and congestion control in scalable Internet video streaming," in *Proc. of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'03)*, Monterey CA, June 2003.
- [10] M. Dai, D. Loguinov, and H. Radha, "Rate-distortion modeling of scalable video coders," in *Proc. of IEEE International Conference on Image Processing (ICIP'04)*, Singapore, October 2004.
- [11] J. Sun, W. Gao, D. Zhao, and Q. Huang, "Statistical model, analysis and approximation of rate-distortion function in MPEG-4 FGS videos," in *Proc. of SPIE International Conference on Visual Communication and Image Processing (VCIP'05)*, Beijing, China, July 2005.
- [12] Y. Wang, J. Ostermann, and Y. Zhang, *Video Processing and Communications*. Prentice Hall, 2002.
- [13] X. Zhang, A. Vetro, Y. Shi, and H. Sun, "Constant quality constrained rate allocation for FGS-coded video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 2, pp. 121–130, February 2003.
- [14] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. MIT Press, 2001.
- [15] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, 1st ed. Dover, 1998.
- [16] D. Goldfarb and M. Todd, "Linear programming," *Handbook in Operations Research and management Science, Optimization*, vol. 1, pp. 73–170, 1989.
- [17] R. Marler and J. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, no. 6, pp. 369–395, April 2004.
- [18] "Coding of audio-visual objects - part 2: Visual," June 2004, ISO/IEC 14496-2.
- [19] "MPEG-4 Visual reference software," February 2004, ISO/IEC 14496.
- [20] X. Su and T. Wang, "Sequence of linear programming for transmission of fine-scalable coded content in bandwidth-limited environments," *ACM/Springer Multimedia Systems Journal*, vol. 11, no. 5, pp. 455–466, June 2006.
- [21] P. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Transactions on Multimedia*, vol. 8, no. 2, pp. 390–404, April 2006.

- [22] J. Chakareski and B. Girod, "Rate-distortion optimized packet scheduling and routing for media streaming with path diversity," in *Proc. of Data Compression Conference (DCC'03)*, Snowbird, UT, March 2003.
- [23] A. Begen, Y. Altunbasak, and M. Begen, "Rate-distortion optimized on-demand media streaming with server diversity," in *Proc. of IEEE International Conference on Image Processing (ICIP'03)*, Barcelona, Spain, September 2003.