Optimal Partitioning of Fine-Grained Scalable Video Streams

Cheng-Hsin Hsu and Mohamed Hefeeda
School of Computing Science
Simon Fraser University, Surrey, Canada
{cha16, mhefeeda}@cs.sfu.ca

ABSTRACT

The increased popularity of video streaming over the Internet attracts numerous clients. These clients are quite heterogeneous in terms of network bandwidth and processing capacity. To accommodate this heterogeneity, fine-grained scalable (FGS) coding of video streams has been proposed in the literature. FGS streams are composed of two layers: base layer, which provides basic quality, and a single enhancement layer that adds incremental quality refinements proportional to the number of bits received. The base layer uses nonscalable coding which is is more efficient in terms of compression ratio than scalable coding used in the enhancement layer. Thus for coding efficiency larger base layers are desired. Larger base layers, however, disqualify more clients from getting the stream. In this paper, we study and quantify the trade-off between the coding efficiency and the range of clients that can be supported. Then, we design an efficient algorithm to compute the optimal size of the base layer that will yield the best video quality for a given client distribution. We implement our algorithm and apply it on video sequences with different characteristics. Our experimental results show that our algorithm improves the average perceived quality for all clients.

1. INTRODUCTION

Video streaming over the Internet is increasingly getting very popular as higher bandwidth links and more powerful machines are becoming more affordable for end users. Users typically seek the highest possible video quality. Users, however, are quite heterogeneous in terms of network bandwidth and processing capacity. A conventional nonscalable coded stream only supports one decoding rate, which is insufficient in such a heterogeneous environment. This is because supporting clients with different bandwidth requires storing and serving multiple versions of each video stream. To cope with this heterogeneity, various scalable coding techniques have been proposed in the literature. A scalable coded stream consists of various representations of the original video sequence, with different resolutions, frame rates, or quality.

Scalable coders are roughly categorized into two classes: coarse-grained scalable and fine-grained scalable. Coarse-grained scalable (CGS) coders divide a video stream into multiple layers. They provide limited rate scalability at the layer level: clients receiving incomplete layers cannot use them to enhance quality. In contrast, fine-grained scalable (FGS) coders provide finer rate scalability and better error resiliency [8, 10, 13]. An FGS encoder compresses video data into two layers: a base layer which provides basic quality, and a single enhancement layer that adds incremental quality refinements proportional to the number of *bits* received. Arbitrary truncation (at the bit level) of the enhancement layer to achieve a target rate is possible for FGS coding. This in turn enables streaming servers to fully utilize available bandwidth of individual clients, which results in better video playback quality and ultimately higher user satisfaction.

The fine rate scalability of FGS, however, comes at an expense of coding efficiency. That is, an FGS coded stream results in lower quality compared to a nonscalable coded stream when both streams are reconstructed at the same bit rate. Previous research indicates that this coding efficiency gap is up to 2 dB in MPEG-4 FGS coders [14]. The two main causes of this coding efficiency gap are: (i) less accurate motion compensation as only base layer is used for motion estimation, and (ii) un-exploited correlation between base layer and enhancement layer. The coding efficiency gap is less significant in video sequences with low temporal redundancy since motion compensation does not provide much quality gain for these sequences. Furthermore, lower base layer rates result in larger gap because less information is contained in the base layer in this case, which leads to higher motion estimation error [10, 14].

While the temporal correlation is fixed for a given sequence, the base layer rate is a configurable parameter. Therefore, content providers may code a sequence at higher base layer rate to reduce the coding efficiency gap and strive for higher quality. This may increase perceived quality for some clients, which could allow the provider to charge higher service rates. On the other hand, a higher base layer rate may disqualify other clients from receiving the complete base layer stream. Since the base layer is nonscalable, these disqualified clients can not even render basic quality and effectively they are denied access to the video stream. Hence, there is a trade-off between coding efficiency and supported rate range, which can be gauged by the FGS base layer rate. More importantly, the average perceived quality for all clients depends on this trade-off, and thus depends on the choice of base layer rate.

To FGS encode a given video sequence, content providers have many options for the base layer rate. Each base layer rate determines the average perceived quality for all clients, which can be used as a metric for user satisfaction. Users who receive higher-quality video are willing to watch more programs or pay higher subscription fees. Thus, content providers need to maximize the average perceived quality to increase their profits. Unfortunately,

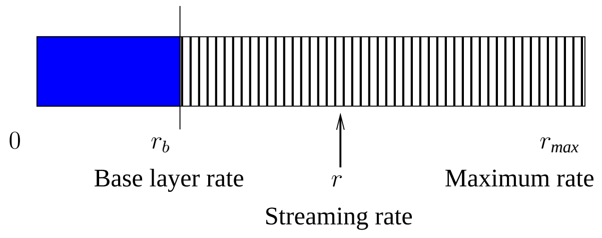


Figure 1: A simple representation of an FGS-coded stream. The stream can be decoded at any rate between r_b and r_{max}.

there are no systematic ways in the literature to aid content providers in choosing the optimal structure of FGS coded streams that would maximize the average quality for all clients.

In this paper, we first study and quantify the trade-off between the coding efficiency and the range of clients that can be supported. Then, we formulate an optimization problem to determine the base layer rate that achieves the best average video quality for a given client distribution. We design an efficient algorithm to solve this optimization problem. We implement our algorithm and apply it on video sequences with different characteristics. Our experimental results show that our algorithm improves the average perceived quality for all clients.

The rest of this paper is organized as follows. In the next section we summarize the related work. In Section 3, we present our analysis of FGS streams, formulate the optimization problem, and present our algorithm to solve it. We evaluate our algorithm in Section 4, and we conclude the paper in Section 5.

2. RELATED WORK

The coding efficiency gap of MPEG-4 FGS coders are studied in [10, 14]. The authors investigate the relationship between the FGS coding efficiency gap and the video temporal correlation. They found that the correlation coefficient between an enhancement layer frame and its motion-compensated reference frame is a good indication of the FGS coding efficiency. We study the efficiency gap of the state-of-the-art H.264 coders. Streaming systems [2, 6, 7, 9] account for the coding efficiency using a layering overhead, which represents the bit rate that does not contribute toward the video quality. Similarly, we model this overhead by a coding efficiency gap function, and we empirically estimate this function.

The performance of layered streams versus nonscalable streams is studied in [11]. The authors formulate a dynamic programming problem to compute the rate of each layer such that the average perceived video quality is maximized. The square root rate-distortion model [1] is used to estimate the coding efficiency of the layered coding. In [9], the authors consider broadcasting multi-layer video streams in a wireless cellular system with a given number of channels and client capacity distribution. They determine the optimal rate of each layer to maximize the average perceived quality. Unlike our work, these two works target coarse-grained streams which provide limited flexibility compared to fine-grained streams.

The authors of [16] study multicast streaming systems with many receivers. They partition receivers into several groups to maximize a system-wide utility function. A video stream used in such systems can be encoded into multiple cumulative layers. Several versions with different rates of the same stream can also be created. This work does not consider fine-grained streams, nor does it account for the layering overhead. Several papers have approximated layering overhead for performance comparison of layered streams and multiple version streams. For example, the work in [7] proposes a linear layering overhead function, which is inspired by the experimental results in [10]. In [2], simulation using the MPEG-2 two-layer scalable structure is employed for such comparison, where the layering overhead is assumed to be proportional to the enhancement layer rate.

3. PROBLEM FORMULATION

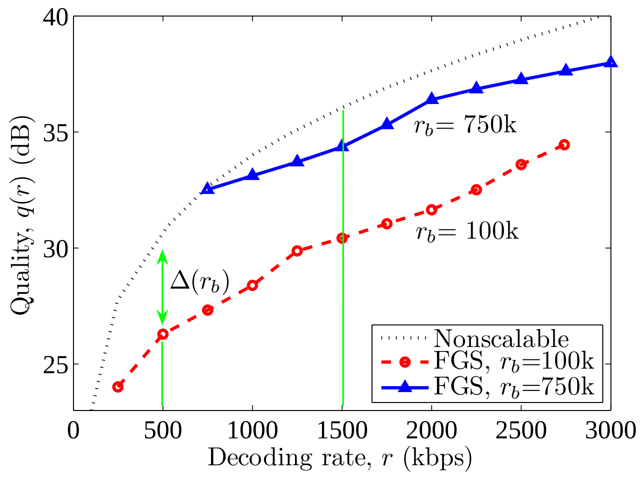
In this section, we first study the characteristics of FGS-coded streams highlighting the trade off between the coding efficiency and the range of clients that can be supported. Then, we formulate and solve the quality optimization problem considered in this paper.

3.1 Characteristics of FGS-Coded Streams

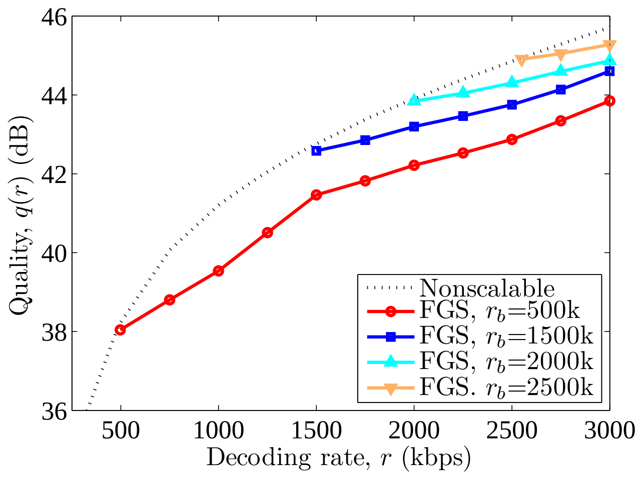
A fine-grained scalable (FGS) video stream is composed of two layers: base layer and enhancement layer. As depicted in Fig. 1, the base layer is nonscalable and must be received in its entirety to provide basic quality, while the enhancement layer can be truncated at arbitrary bit positions. Therefore, an FGS-coded stream can support a wide range of streaming rates, and thus many heterogeneous clients. Let us denote the bit rate of the base layer as r_b, and the maximum bit rate of the video stream as r_{max}. r_{max} corresponds to the maximum possible quality of the video stream, and it is specified by the resources (storage and bandwidth) allocated to the video stream by the system administrator. An FGS-coded stream can be served at any bit rate r, where $r_b \leq r \leq r_{max}$.

Our problem in this paper is to determine the best base layer rate r_b so that the average quality is maximized for all clients. To solve this problem, we need to study the implications of varying r_b. We design the following experiments to analyze these implications. We use the Joint Scalable Video Model (JSVM) reference software version 8.0 [5] in our experiments. A brief description of this software and how we configured it is given in Section 4. We choose two standard video sequences: City and Mobile, both in CIF format with 30 frames per second. We set r_b at a specific value and encode the whole stream with a maximum rate $r_{max} = 3000$ kbps. Then we determine the quality that would be perceived by various clients decoding the stream at different rates. We consider clients in the range between 250 kbps and 3000 kbps with a step of 250 kbps. The quality is determined by decoding the stream and computing the peak signal to noise ratio (PSNR) in dB. We repeat the whole experiment for several values of the base layer rate, and for the two sequences. These are computationally intensive experiments and took many processing hours to complete.

The results of these experiments are presented in Fig. 2. Several observations can be drawn from this figure. First, the FGS streams have lower coding efficiency. For example, Fig. 2(a) indicates that decoding a nonscalable stream at rate 500 kbps results in 30 dB video quality, while decoding an FGS stream (with r_b =100 kbps) results in 26 dB video quality. We define the *quality gap* $\Delta(r_b)$ as the quality difference between a nonscalable stream and a fine-grained scalable stream coded with base layer rate r_b. The quality gap can be explained by the additional overhead and un-exploited video redundancy caused by the scalable coding structure. A second observation we can make from Fig. 2 is that higher base layer rates lead to smaller quality gaps. For example, Fig. 2(a) shows that at decoding rate 1500 kbps, an FGS stream with r_b = 750 kbps results in about 1 dB quality gap compared to nonscalable stream, while an FGS stream with r_b = 100 kbps results in 6 dB quality gap. These differences can be explained by the fact that more temporal redundancy can be exploited if the base layer contains more information, i.e., is coded at a higher rate. This observation indicates that the quality gap $\Delta(r_b)$ is a non-increasing function of the base



(a) Mobile



(b) City

Figure 2: The coding efficiency gap between FGS and nonscalable streams. The gap decreases as the base layer rate r_b increases. Increasing r_b, however, limits the number of clients that can receive the stream.

layer r_b. We further validate this property in the Section 4. We will use this non-increasing property in solving the quality optimization problem in the next subsection.

Finally, we note that similar scalable coding inefficiencies were observed in MPEG-4 FGS coders [10]. This is consistent with our observations on the recent H.264 coders.

3.2 Problem Formulation

In this section, we formulate an optimization problem to find the base layer rate r_b that achieves the highest average perceived quality for all clients. We consider heterogeneous client populations. We model this heterogeneity by dividing clients into C classes. All clients belonging to the same class c $(1 \le c \le C)$ have the same bandwidth b_c. We assume that $b_1 < b_2 < \cdots < b_C$. The fraction of clients in each class c is given by a probability mass function $f(b_c)$, where $\sum_{c=1}^{C} f(b_c) = 1$. No assumptions are made on the number of client classes or on the probability function. Without loss of generality, we assume that $b_C \le r_{max}$. If otherwise, we combine clients with bandwidth larger than r_{max} in a class with bandwidth equal to r_{max}. We can do that because no matter how large the client bandwidth is, it cannot receive more than the maximum rate r_{max}.

We write the optimization problem P that maximizes the average perceived quality as follows:

$$P: \quad \max_{r_b} \sum_{c=1}^{C} q(b_c) f(b_c), \text{where } r_b \in [0, r_{max}], \tag{1}$$

where $q(b_c)$ is the quality (measured as PSNR in dB) achieved by clients in class c.

A naive approach to solve the above problem is to try all possible values for r_b in the range $[0, r_{max}]$. This is very costly because FGS coders allow for too many possibilities for r_b. We propose a better solution that takes at most $O(C)$ steps. Our approach is enabled by the following lemma.

LEMMA 1. *An optimal solution r_b^* for the base layer rate that maximizes the average perceived quality for all users can be found at one of the rates b_c, where $1 \le c \le C$.*

PROOF. Referring to Fig. 2(a), we can re-write the quality of the FGS stream $q(b_c)$ for clients in class c as:

$$q(b_c) = \begin{cases} 0, & b_c < r_b \\ q_{ns}(b_c) - \Delta(r_b), & b_c \ge r_b, \end{cases} \tag{2}$$

where $q_{ns(b_c)}$ is the quality achieved by the nonscalable encoder at rate b_c, and $\Delta(r_b)$ is the quality gap between the FGS and nonscalable streams as defined in the previous subsection. Notice that the quality for clients in any class c is zero if these clients do not have enough bandwidth to receive the complete base layer, i.e., if $b_c < r_b$.

We divide the search range $[0, r_{max}]$ into non-overlapping intervals $(b_{c-1}, b_c]$, where $c = 1, 2, \ldots, C$ and $b_0 = 0$. Now assume that the optimal base layer rate r_b occurs in an arbitrary interval $(b_{z-1}, b_z]$. Since all classes with $b_c \le b_{z-1}$ receive quality of zero, the maximization problem becomes:

$$\max_{r_b} \sum_{c=z}^{C} [q_{ns}(b_c) - \Delta(r_b)] f(b_c), \text{where } r_b \in (b_{z-1}, b_z]. \tag{3}$$

Notice that the only term that depends on r_b in the above equation is the quality gap $\Delta(r_b)$. Thus to maximize quality, we need to minimize $\Delta(r_b)$. Recall that in the previous subsection we argued that $\Delta(r_b)$ is non-increasing function of r_b, we validate this argument in Section 4. Since $\Delta(r_b)$ is non-increasing in the interval $(b_{z-1}, b_z]$, no point in that interval could make the quality gap smaller than $\Delta(r_b = b_z)$. Thus, an optimal solution for r_b occurs at b_z. $\square$

The above lemma tells us that to find an optimal base layer rate r_b^*, it suffices to check only the rates $b_c (c = 1, 2, \ldots, C)$. A straightforward approach to implement this lemma is to compute equation (3) at $c = 1, 2, \ldots, C$ and choose the rate the corresponds to the maximum quality. This would require computing the summation at every iteration, which would make the time complexity of the algorithm $O(C^2)$. A better approach is to iteratively compute each term from $c = C$ towards $c = 1$, and every iteration only adds the difference in quality to the quality computed in the previous iteration. The difference d in quality between class c and

FGSOPT

```
1.  $q_1 = q_2 = \cdots = q_C = 0$;
2.  $q_{max} = q_C = [q_{ns}(b_C) - \Delta(b_C)]f(b_C)$;
3.  $r_b^* = b_C$;
4.  $f_{rcv} = f(b_C)$;
5.  for $c = C - 1$ to $1$ {
6.      $d = [q_{ns}(b_c) - \Delta(b_c)]f(b_c) - f_{rcv}[\Delta(b_c) - \Delta(b_{c+1})]$;
7.      $q_c = q_{c+1} + d$;
8.      if $q_c > q_{max}$ {
9.          $q_{max} = q_c$;
10.         $r_b^* = b_c$;
11.     }
12.     $f_{rcv} = f_{rcv} + f(b_c)$;
13. }
14. return $r_b^*$;
```

Figure 3: An algorithm to compute the optimal base layer rate for FGS-coded streams.

$c + 1$ is given by the following equation:

$$d = [q_{ns}(b_c)) - \Delta(b_c)]f(b_c) - \sum_{i=c+1}^{C} f(b_i)[\Delta(b_c) - \Delta(b_{c+1})].$$

The first term represents the quality improvement because clients with bandwidth b_c is capable to receive coded streams. The second term represents the quality degradation on all clients that have bandwidth larger than b_c because of a larger coding efficiency gap. Using this idea, we propose an efficient algorithm, called FGSOPT, that computes an optimal value for the base layer rate. The pseudo code of the algorithm is given in Fig. 3. The inputs to the algorithm are: (i) a probability mass function $f(b_c)$ that describes the distribution of bandwidth to different client classes, (ii) a rate-distortion function $q_{ns}(r)$ that yields the expected quality when decoding the nonscalable video stream at rate r, and (iii) a quality gap function $\Delta(r_b)$ that describes the reduction in quality if the video stream were to be encoded in FGS manner with base layer rate r_b. The output of the algorithm is the optimal base layer rate. The time complexity of the algorithm is clearly $O(C)$.

Our algorithm does not target live streaming systems. Rather, it targets video on-demand systems, where a video sequence is expected to be streamed to many clients over an extended period of time. Therefore, the cost of computing or estimating the inputs of the algorithm is justified by the quality improvement observed by the clients. Moreover, this cost is controllable: For videos with expected low demand, the administrator can quickly compute rough estimates of the inputs, while for popular videos more elaborate estimations can be done. For example, $\Delta(r_b)$ can be estimated by a line with negative slope. This requires only two sample points. As we show in Section 4, a degree 4 polynomial function is a better representation of $\Delta(r_b)$, but would require more sample points. Similarly, the rate-distortion function can be estimated by sophisticated analytic methods [3] or by simple curve fitting. In Section 4, we show that a quadratic function models the relationship between the rate and expected distortion fairly accurately.

4. EVALUATION

In this section, we first describe our experimental setup, and present the quality improvement by using our algorithm. We then study the non-increasing property of the gap function, and show that a simple quadratic function can approximate the R-D function. We present a sample of our results in this section due to space limitations.

4.1 Setup

The Video Coding Experts Group (VCEG) and the Moving Picture Experts Group (MPEG), known as the Joint Video Team (JVT), are developing a scalable video coding (SVC) standard as an extension of H.264 standard [4]. The emerging SVC design is detailed in the SVC Amendment Working Draft [15] and Joint Scalable Video Model (JSVM) [12]. The JSVM reference software is provided by the JVT team to demonstrate an effective implementation that complies with the SVC standard. We use the JSVM reference software version 8.0 [5] in our experiments.

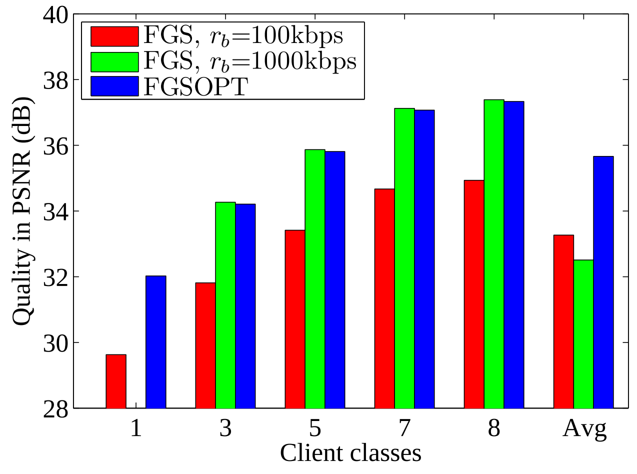
The JSVM reference software is implemented in C++, and contains several executables. We use the following executables: *H264-AVCEncoderLibTest*, *BitStreamExtractor*, *H264AVCDecoderLib-Test*, and *FixedQPEncoder*. The *H264AVCEncoderLibTest* is a configurable SVC encoder that can compress a raw video file into a global stream. This global stream consists of many embedded substreams, which deliver lower quality video representations at lower rates. The global stream is stored as a file. The *BitStreamExtractor* tool extracts a user-specified substream from an existing global stream and stores it in a new file. Further stream extractions from this substream file are possible as the syntax and semantics of the global stream files and substream files are identical. The *H264AVCDecoderLibTest* is an SVC decoder that decompresses coded stream into a raw video file. Since the *H264AVCEncoderLib-Test* does not implement rate control algorithm for a user-specified rate constraint, we have to use quantization parameter (QP) to gauge the resulted stream rate. The *FixedQPEncoder* is a tool that searches the proper QPs to satisfy rate constraints. It iteratively calls *H264-AVCEncoderLibTest* with estimated QP values, and stops when the resulted stream rate is within an acceptable range of the desired rate.

We choose two standard video sequences for our experiments: City and Mobile, both are in CIF format with frame frequency 30Hz. We encode them with the widely adopted IBBBPBBBP group of picture (GoP) structure. We first encode a sequence with single layer configuration using the *FixedQPEncoder* tool to get appropriate QP values for the target base layer rate. We then use the same QP values to code an FGS stream.

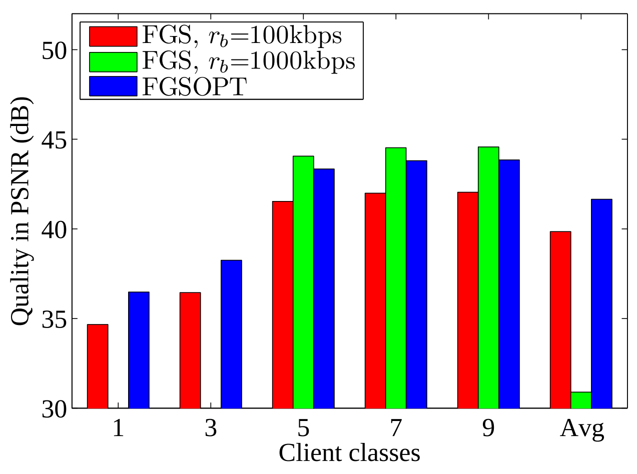
4.2 Average Quality Improvement

We have implemented our FGSOPT algorithm in Matlab. As mentioned in the introduction section, we are not aware of similar algorithms in the literature that optimize the average quality by controlling the base layer rate. Therefore, we compare the results of our algorithm to the results of heuristic methods. That is, we choose two *reasonable* rates for the base layer and compare the resulting quality against the quality produced by our algorithm. Indeed, there are too many other choices and we cannot cover all of them in our experiments. This is not really an issue because our algorithm is provably optimal, and the best that heuristic methods can do is to approach our algorithm by trial and error.

We choose ten classes and uniformly distribute clients among these classes. The bandwidth range for clients is between 0 and 3000 kbps. We run the FGSOPT algorithm to compute the optimal base layer rate. We choose two base layer rates for comparison: 100 and 1000 kbps. We compute the perceived quality for each client class and the average quality over all classes. The results are shown in Fig. 4. The figure clearly shows that the average quality over all classes has been improved using our FGSOPT algorithm.



(a) Mobile



(b) City

Figure 4: Expected quality of individual client classes and the overall average quality for all clients when the base layer rate r_b is set to three different values: the optimal computed by our algorithm (denoted by FGSOPT), and two other rates. Clients in even number classes are omitted for figure legibility.

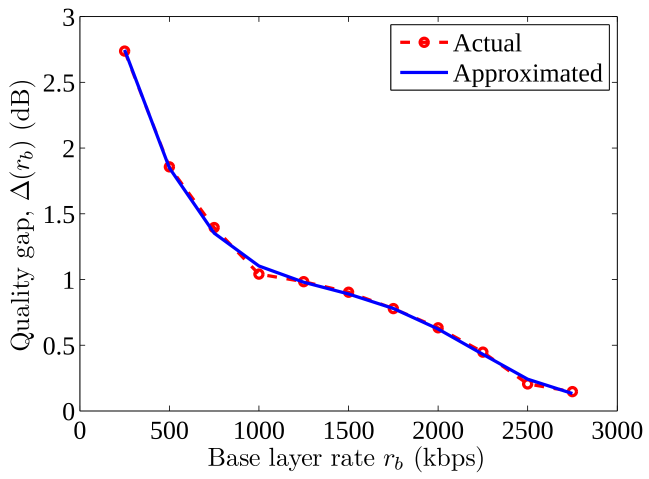


Figure 5: The coding efficiency gap $\Delta(r_b)$ between FGS and nonscalable streams. The figure shows that $\Delta(r_b)$ is non-increasing function, and it can be modeled by a polynomial function with degree 4. Sample data shown for Mobile sequence.

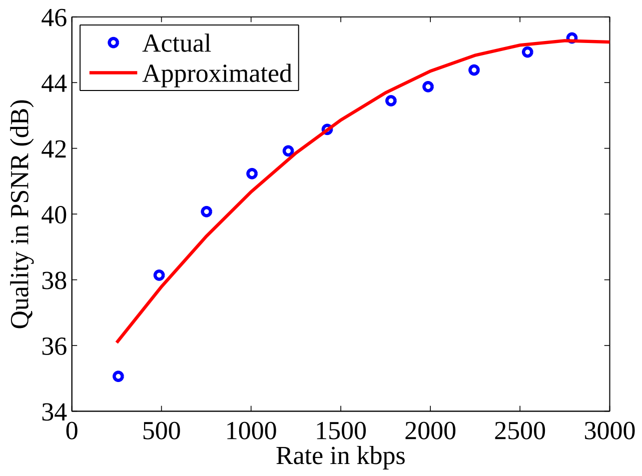


Figure 6: The rate-distortion (R-D) function $q_{ns}(r)$ of nonscalable streams. The figure shows that a quadratic function provides a good approximation for the R-D function. Sample data shown for City sequence.

4.3 Quality Gap Function

The FGSOPT algorithm assumes that the quality gap $\Delta(r_b)$ is a non-increasing function of base layer rate r_b. To validate the accuracy of this assumption, we compute the quality gap at various base layer rates. We use the reference software to encode the two test sequences with base layer rates between 100 kbps and 3000 kbps with an increment of 250 kbps. Each base layer rate results in a unique FGS coded stream that supports decoding rates between r_b and 3000 kbps. To quantify the coding efficiency gap at a specific base layer rate r_b, we decode the stream at many decoding rates between r_b and 3000 kbps and we take the average over all of them. We compute the reconstructed quality at each decoding rate by first extracting the substream that matches this rate. Then we decode the extracted substream and compare it against the original video stream. A sample result is shown in Fig. 5. The figure confirms the non-increasing property of the quality gap function. On the same figure we plot a degree 4 polynomial function that best-fits the quality gap curve. We do this because the FGSOPT algorithm needs to computes the quality gap $\Delta(r_b)$ at different base layer rates. Thus instead of empirically measuring the quality gap at too many base layer rates, which is computationally expensive, we estimate the polynomial function and employ it in the algorithm. Estimating the polynomial function requires measuring the quality gap only at a few base layer rates.

4.4 Rate-Distortion Function

The FGSOPT algorithm requires a rate-distortion (R-D) function that estimates the expected distortion at a given decoding rate when the stream is encoded in a nonscalable manner. Through extensive experiments, we have found that this R-D function can be approximated by a simple quadratic function. Fig. 6 shows a sample result, where we compute the R-D function at 12 sampling bit rates for City sequence. The figure also shows the best-fit quadratic function produced by the Matlab curve-fitting tool for the sample points. We note that the sample result in Fig. 6 provides guidelines for the administrator on the shape of the R-D functions and should be considered as a first approximation. Indeed, more elaborate R-D models can be found in the literature, but they are quite complex and expensive to implement. For detailed discussion and comparisons of various R-D models, see for example [3] and references therein.

5. CONCLUSION AND FUTURE WORK

In this paper, we first investigated the characteristics of FGS coded video streams. We designed several experiments using the emerging H.264/MPEG-4 SVC coder to study the trade-off between the coding efficiency and the range of clients that can be supported. The base layer rate is the main controlling parameter: Larger base layer rates yield higher coding efficiency but support fewer client classes, and vice versa. Our experiments show that the coding efficiency gap is a non-increasing function of the base layer rate. Then, we formulated an optimization problem to determine the base layer rate that achieves the best average video quality for a given client distribution. Solving this optimization problem is expensive, because there are too many possible choices for the base layer rate of FGS coded streams. To address this complexity, we proposed a simple algorithm that runs in linear time. We proved that our algorithm yields the optimal base layer rate. We implemented our algorithm and compared the FGS structures produced by it against two rule-of-thumb coding structures, which is the current practice. Our results indicated that our algorithm achieves better average perceived quality for all clients. Our proposed algorithm can be used in various applications. For example, given a long-term client distribution, our algorithm can aid content providers in choosing the optimal structure of stored FGS coded streams, so that the maximum average quality for all clients is achieved. Consequently, clients will have better viewing experience and higher satisfaction.

We are currently extending this work to multiple FGS video streams. We are considering a streaming server with a given network capacity. The server has multiple streams to encode and serve to diverse client communities. The streams are assumed to have different popularities. Our objective is to determine the structure of each video stream such that the server maximally utilizes its capacity and achieves the best perceived quality for all clients receiving the video streams. We are solving the problem for unicast as well as multicast streaming systems.

6. REFERENCES

[1] M. Dai, D. Loguinov, and H. Radha. Rate-distortion analysis and quality control in scalable Internet streaming. *IEEE Transactions on Multimedia*, 8(6):1135–1146, December 2006.

[2] P. de Cuetos, D. Saparilla, and K. Ross. Adaptive streaming of stored video in a TCP-friendly context: Multiple versions or multiple layers? In *Proc. of International Packet Video Workshop (PV'00)*, Kyongju, Korea, April 2001.

[3] C. Hsu and M. Hefeeda. On the accuracy and complexity of rate-distortion models for fine-grained scalable video sequences. Technical Report TR 2006-12, Simon Fraser University, August 2006. Available online at http://nsl.cs.surrey.sfu.ca/projects/fgs/.

[4] ITU-T Rec. H.264 & ISO/IEC 14496-10 AVC. Advanced video coding for generic audiovisual services, 2005.

[5] Joint Video Team. Joint scalable video model reference software. JSVM 8.0, February 2007.

[6] T. Kim and M. Ammar. A comparison of layering and stream replication video multicast schemes. In *Proc. of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'01)*, Port Jefferson, NY, June 2001.

[7] T. Kim and M. Ammar. A comparison of heterogeneous video multicast schemes: layered encoding or stream replication. *IEEE Transactions on Multimedia*, 7(6):1123–1130, December 2005.

[8] W. Li. Overview of fine granularity scalability in MPEG-4 video standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(3):301–317, March 2001.

[9] J. Liu, B. Li, Y. Hou, and I. Chlamtac. Dynamic layering and bandwidth allocation for multi-session video broadcasting with general utility functions. In *Proc. of IEEE INFOCOM'03*, San Francisco, CA, March 2003.

[10] H. Radha, M. van der Schaar, and Y. Chen. The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP. *IEEE Transactions on Multimedia*, 3(1):53–68, March 2001.

[11] I. Radulovic, P. Frossard, and O. Verscheure. Adaptive video streaming in lossy networks: versions or layers? In *Proc. of IEEE International Conference on Multimedia and Expo (ICME'04)*, Taipei, Taiwan, June 2004.

[12] J. Reichel, H. Schwarz, and M. Wien. Joint scalable video model JSVM-8. Technical Report JVT-U202, Joint Video Team, Hangzhou, China, October 2006.

[13] H. Schwarz, D. Marpe, and T. Wiegand. The scalable H.264/MPEG4-AVC extension: Technology and applications. In *European Symposium on Mobile Media Delivery (EuMob'06)*, Sardinia, Italy, September 2006.

[14] M. van der Schaar and H. Radha. Adaptive motion-compensation fine-granular-scalability (AMC-FGS) for wireless video. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(6):32–51, June 2002.

[15] T. Wiegand, G. Sullivan, J. Reichel, H. Schwarz, and M. Wien. Joint draft 8 of SVC amendment. Technical Report JVT-U201, Joint Video Team, Hangzhou, China, October 2006.

[16] Y. Yang, M. Kim, and S. Lam. Optimal partitioning of multicast receivers. In *Proc. of IEEE International Conference on Network Protocols (ICNP'00)*, Osaka, Japan, November 2000.