UNIVERSITY OF CALIFORNIA, SAN DIEGO

Bit-rate allocation for multiple video streams: dual-frame video coding and competitive equilibrium methods

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering
(Signal and Image Processing)

by

Mayank Tiwari

Committee in charge:

Professor Pamela Cosman, Chair
Professor Theodore Groves, Co-Chair
Professor Lawrence Milstein
Professor Truong Nguyen
Professor Geffeory Voelker

2010

The dissertation of Mayank Tiwari is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____
Co-Chair

_____
Chair

University of California, San Diego

2010

DEDICATION

To my parents,
Saroj and Dinesh Chandra Tiwari

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my deepest gratitude to my advisors, Prof. Pamela Cosman and Prof. Theodore Groves, for their support, guidance, and invaluable advice for the past several years.

Prof. Cosman was very instrumental in providing the right research direction and making sure that I got all the support to achieve my goals. Her constructive criticism and motivation helped better understand and improve myself. I am thankful to her for arranging the financial support throughout my graduate studies at UCSD. I am also thankful to her for always finding time to read and correct all my research papers.

Prof. Groves unleashed an entire new area of research to me and was very instrumental in helping me achieve my goals. Apart from providing infinite research opportunities and advice, he was also helpful in getting me resources for my research work. He was not only a good PhD advisor but also a good mentor and a friend to me. I learnt a lot from his vast knowledge and he will always be a great source of inspiration to me.

I am also thankful to my other committee members, Prof. Laurence Milstein, Prof. Truong Nguyen, and Prof. Geoff Voelker, for providing helpful comments and suggestions during my PhD.

I am thankful to my fellow researchers in the Information Coding Laboratory for the friendly and dynamic research atmosphere. Special thanks to Athanasios Leontaris, Ramesh Annavajjala, Yushi Shen, Sandeep Kanumuri, Sitaraman Ganapathy, Tig-Lan Lin, Hobin Kim, and Seok-Ho Chang for many helpful discussion and comments. I am also thankful to my friends Naomi Ramos, Debashis Panigrahi, Shoubhik Mukhopadhyay, Krishna Sekar, and Vijay Mahadevan for their constant assistance throughout my PhD.

I would like to thank my brothers, Manish and Mitul, for their guidance and encouragement. I thank my daughter, Mihika, for being the source of inspiration in the most difficult times. I thank my wife, Rashmi, for going through both good and bad times with me throughout my PhD. I started my PhD because of her faith in me and it would have been impossible to complete this journey without

her assistance.

Finally, I would like to thank my father Dr. Dinesh Chandra Tiwari and my mother Saroj Tiwari for their unconditional love and support through all these years. They provided us the best education possible with their sacrifices at various stages of life.

Chapter 2 of this dissertation contains material that appears in M. Tiwari and P. Cosman, "Selection of long-term reference frames in dual frame video coding using simulated annealing", *IEEE Signal Processing Letters*, Vol. 15, pp 249-252, 2008 and M. Tiwari, T. Groves, and P. Cosman, and "Delay constrained rate control for low bitrate dual-frame video coding", *IEEE International Conference on Image Processing*, San Diego, California, pp 2484-2487, Oct 2008. I was the primary author and the co-authors Dr. Pamela Cosman and Dr. Theodore Groves directed and supervised the research which forms the basis for Chapter 2.

Chapter 3 of this dissertation contains material that appears in M. Tiwari, T. Groves, and P. Cosman, "Delay constrained multiplexing of video streams using dual-frame video coding", *IEEE Transactions on Image Processing*, Vol. 19, No. 4, pp. 1022-1035, April 2010. I was the primary author and the co-authors Dr. Pamela Cosman and Dr. Theodore Groves directed and supervised the research which forms the basis for Chapter 3.

Chapter 4 of this dissertation contains material that appears in M. Tiwari, T. Groves, and P. Cosman, "Competitive equilibrium bitrate allocation for multiple video streams" *IEEE Transactions on Image Processing*, Vol. 19, No. 4, pp. 1009-1021, April 2010. I was the primary author and the co-authors Dr. Pamela Cosman and Dr. Theodore Groves directed and supervised the research which forms the basis for Chapter 4.

Chapter 5 of this dissertation contains material that appears in M. Tiwari, T. Groves, and P. Cosman, "Bit-rate allocation for multiple video streams using a pricing-based mechanism", *IEEE Transactions on Image Processing*, in review since February 2010. I was the primary author and the co-authors Dr. Pamela Cosman and Dr. Theodore Groves directed and supervised the research which forms the basis for Chapter 5.

VITA

1999	B. E., *Honors*, Electronics and Computer Engineering (Electronics and Communications), Indian Institute of Technology, Roorkee, India
1999-2001	Software Engineer, Hughes Software Systems, Gurgaon, India
2001-2002	R & D Engineer, Cybernetics Infotech Inc., Rockville, Maryland, USA
2002-2004	Graduate Research Assistant, Arizona State University, Tempe, USA
2004	M. S., Electrical Engineering (Communication and Signal Processing), Arizona State University, Tempe, USA
2004-2009	Graduate Student Researcher, University of California, San Diego, La Jolla, USA
2008	Graduate Teaching Assistant, University of California, San Diego, La Jolla, USA
2010-Present	Senior Engineer, Qualcomm Inc., San Diego, USA
2010	Ph. D., Electrical Engineering (Signal and Image Processing), University of California, San Diego, La Jolla, USA

PUBLICATIONS

M. Tiwari and P. Cosman, "Dual frame video coding with pulsed quality and a lookahead window", *IEEE Data Compression Conference*, Snowbird, Utah, pp. 372-381, Mar 2006.

M. Tiwari and P. Cosman, "Selection of long-term reference frames in dual frame video coding using simulated annealing", *IEEE Signal Processing Letters*, Vol. 15, pp. 249-252, 2008.

M. Tiwari, T. Groves, and P. Cosman, "Multiplexing video streams using dual-frame video coding", *IEEE International Conference on Acoustics, Speech and Signal Processing*, Las Vegas, Nevada, pp. 693-696, Mar 2008.

M. Tiwari, T. Groves, and P. Cosman, "Delay constrained rate control for low bitrate dual-frame video coding", *IEEE International Conference on Image Processing*, San Diego, California, pp. 2484-2487, Oct 2008.

M. Tiwari, T. Groves, and P. Cosman, "Bitrate allocation for multiple video streams at competitive equilibria", *IEEE Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, California, pp. 2248-2252, Oct 2008.

M. Tiwari, T. Groves, and P. Cosman, "Pricing-based decentralized rate allocation for multiple video streams", *IEEE International Conference on Image Processing*, Cairo, Egypt, pp. 3065-3068, Nov 2009.

M. Tiwari, T. Groves, and P. Cosman, "Competitive equilibrium bitrate allocation for multiple video streams", *IEEE Transactions on Image Processing*, Vol. 19, No. 4, pp. 1009-1021, April 2010.

M. Tiwari, T. Groves, and P. Cosman, "Delay constrained multiplexing of video streams using dual-frame video coding", *IEEE Transactions on Image Processing*, Vol. 19, No. 4, pp. 1022-1035, April 2010.

M. Tiwari, T. Groves, and P. Cosman, "Bit-rate allocation for multiple video streams using a pricing-based mechanism", *IEEE Transactions on Image Processing*, in review.

FIELDS OF STUDY

Major Field: Electrical Engineering

Studies in Signal and Image Processing.

Professors Pamela Cosman and Theodore Groves

ABSTRACT OF THE DISSERTATION

Bit-rate allocation for multiple video streams: dual-frame video coding and competitive equilibrium methods

by

Mayank Tiwari

Doctor of Philosophy in Electrical Engineering
(Signal and Image Processing)

University of California, San Diego, 2010

Professor Pamela Cosman, Chair
Professor Theodore Groves, Co-Chair

With the advancement of digital video technology in recent years, there has been an enormous surge in the amount of video data sent across networks. In many cases, a transmission link is shared by more than one video stream. Applications where multiple compressed video streams are transmitted simultaneously through a shared channel include direct broadcast satellite (DBS), cable TV, video-on-demand services, disaster relief response, and video surveillance. Some commercial applications are YouTube and instant video streaming by content providers, such as Netflix, where multiple streams are transmitted simultaneously, and in many cases, these streams share a common transmission channel. Recently, in cognitive radio technology, the secondary or unlicensed users share a pool of bandwidth that is temporarily going unused by the primary or licensed users. In such cases, it has been shown that joint bit-rate allocation schemes for multiple streams can perform better than an equal bit-rate allocation.

In this dissertation, we consider the problem of bit-rate allocation for multiple video users sharing a common transmission channel. We consider two separate objectives for bit-rate allocation among multiple video users: (a) improving the video quality averaged across all the users, and (b) improving the video quality of each individual user, compared to the bit-rate allocation for the users when acting independently.

We use dual-frame video with high-quality Long-Term Reference (LTR) frames, and propose multiplexing methods to reduce the sum of Mean Squared Error (MSE) for all the users. We make several improvements to dual-frame video coding by selecting the location and quality of LTR frames. An adaptive buffer-constrained rate-control algorithm is devised to accommodate the extra bits of the high-quality LTR frames. Multiplexing of video streams was studied under the constraint of a video encoder delay buffer. The high-quality LTR frames are offset in time among different video streams. This provides the benefit of dual-frame video coding with high-quality LTR frames while still fitting under the constraint of an output delay buffer. The multiplexing methods show considerable improvement over conventional rate control when the video streams are encoded independently, and over multiplexing methods proposed previously in the literature.

While the average video quality is improved for multiple video users, such methods often rely on identifying the relative complexity of the video streams. In such methods, not all the videos benefit from the multiplexing process. Typically, the quality of high motion videos is improved at the expense of a reduction in the quality of low motion videos. We use a competitive equilibrium allocation of bit-rate to improve the quality of each individual video stream by finding trades between videos across time. A central controller collects rate-distortion information from each video user and makes a joint bit-rate allocation decision. The proposed method uses information about not only the differing complexity of the different video streams at a given instant of time, but also the differing complexity of each stream over time. Using the competitive equilibrium bit-rate allocation approach for multiple video streams, we show that all the video streams perform better or at least as well as with individual encoding.

The centralized bit-rate allocation methods share the video characteristics and involve high computational complexity. In our pricing-based method, we present an informationally decentralized bit-rate allocation for multiple users where a user only needs to inform his demand to an allocator. Each user separately calculates his bit-rate demand based on his video complexity and bit-rate price, where the bit-rate price is announced by the allocator. The allocator adjusts the bit-rate price based on the bit-rate demanded by the users and the total available bit-rate supply. We show that all users improve their quality by the pricing-based decentralized bit-rate allocation method compared to their allocation when acting individually and the results are comparable to the centralized bit-rate allocation.

Chapter 1

Introduction

In recent years, multimedia transmission is emerging as an important application for all types of networks. Much of the information that was shared through voice/audio has been replaced by video. There is a huge development in the gaming industry involving real-time multimedia transmission among various players. A recent article in the Economist magazine showed how video conferencing, such as Cisco telepresence, is economical and is slowly replacing the need for business travel. Many conferences are now shifting towards webpresence where the attendees are not required to travel. Online weddings are becoming more popular everyday. Instead of sending DVDs, movie rental companies such as Netflix are gearing towards online movie streaming. Recently, online video sharing applications such as YouTube are entering the movie streaming business. With the emergence of cognitive radio, many secondary users share bandwidth that is going temporarily unused by the primary users. Video surveillance is another important application where multiple cameras are deployed at different locations to capture video which is transmitted to a central place where it is monitored. A similar video transmission scenario exists in the case of disaster relief response where multiple teams of responders transmit video to a central location. Some other applications where multiple compressed video streams are transmitted simultaneously through a shared channel include direct broadcast satellite (DBS), cable TV, and video-on-demand services.

Such transmission of multiple video streams from a central server (or from

multiple servers but with centralized rate allocation) to multiple destinations over a communication channel is a familiar scenario in many applications. In many cases, a transmission link is shared by more than one video stream. As digital video compression technology becomes more efficient, more video streams can be compressed and transmitted together. The total bit-rate of multiple video streams is limited by the bandwidth of the central server. The growth in simultaneous video transmission over communication channels by multiple users has stimulated efforts to better allocate shared resources such as bit-rate among users. Equally distributing the available resources among the video streams often produces a poor result. Instead of equally dividing available bit-rate among videos, a number of joint bit-rate allocation algorithms have been proposed to improve the overall video quality. In such cases, it has been shown that joint bit-rate allocation schemes for multiple streams can perform better than an equal bit-rate allocation. Therefore, it is important to efficiently allocate the overall bit-rate among the compressed video streams at every time instant to enhance the overall quality. With constraints on the bit-rate that can be achieved from a channel, ever increasing demand for higher resolution video, and limited compression efficiency, it is essential to research algorithms which can further improve the video quality for such conditions.

In this dissertation, we focus on two issues related to transmission of multiple video streams: (a) improving the video compression efficiency using dual-frame video coding, and (b) better bit-rate allocation among multiple video streams using dual-frame video coding and competitive equilibrium methods.

1.1 Video compression

Growth in video transmission was made possible by state-of-the-art video compression algorithms. Earlier, two standardization bodies were responsible for developing video compression standards. The Motion Picture Experts Group (MPEG), a study group for the ISO/IEC standardization body, developed the MPEG-1, MPEG-2, and MPEG-4 [1] video standards. The Video Coding Experts Group, a study group for the ITU-T standardization body, was responsible for de-

velopment of the H.261 and H.263 video standards. A joint effort between MPEG and VCEG, known as the Joint Video Team (JVT), is responsible for the development of the H.264/AVC [2] video standards, also known as MPEG-4 part 10. Typically, video data is represented by pixels, and each pixel consists of red, blue, and green color components. A high-definition television (HDTV) with a spatial resolution of 1920×1080 pixels at 30 frames per second would require a data rate of 1.39 Tbits/second, assuming each color is represented by 8 bits. Clearly, this requires a large amount of bandwidth to support the transmission of such videos. Therefore, video compression plays an important role is reducing the data size so that such videos can be transmitted over a realistic channel.

There are three types of coding that can be used to encode each frame of a video. These coding modes are described below in a general sense. There are many video standard-specific issues that are involved in all these coding modes which are ignored here.

Intra-frame coding: In Intra-frame (I) coding, each frame is encoded independently, and does not use the information content of past or future frames. Each frame is treated as an image, and only the spatial redundancy is utilized for compression. The frames are first divided into blocks (for example, 8×8 pixels) for applying transforms (such as a discrete cosine transform). The transform domain signal is passed through the quantization block which essentially scales down the coefficients. The quantization step is inherently a part of lossy coding where information is lost during encoding. Entropy coding is then applied to the quantized signal. The entropy coding is a lossless step and it does not add any further distortion to the signal. Motion JPEG is one such example where all the frames are I-coded.

By not utilizing the temporal redundancy, I-coding does not provide good compression efficiency. However, this coding mode is computationally efficient and provides random access into a video sequence. Since the frames are encoded independently of each other, this coding mode also is free from frame-to-frame error propagation.

Inter-frame coding: In Inter-frame (P) coding, the frames that are al-

ready encoded in the past are used to remove the temporal redundancy of a video sequence. The frames are divided into macroblocks (MB) (typically, 16 × 16 pixels) and a good match is searched in the past encoded frames. The process of finding a good match for a MB in the current frame from the past frames is called motion estimation. Fig. 1.1 shows an example of motion estimation where the current frame is divided into MBs. For each MB, a search is conducted in a previously encoded frame. Once a good match is found, the relative location of the MB from the previous frame, called a motion vector, is transmitted. The difference between the MB from the current frame and the best match from the previous frame is passed through the transform, quantization, and entropy coding steps to achieve further compression.

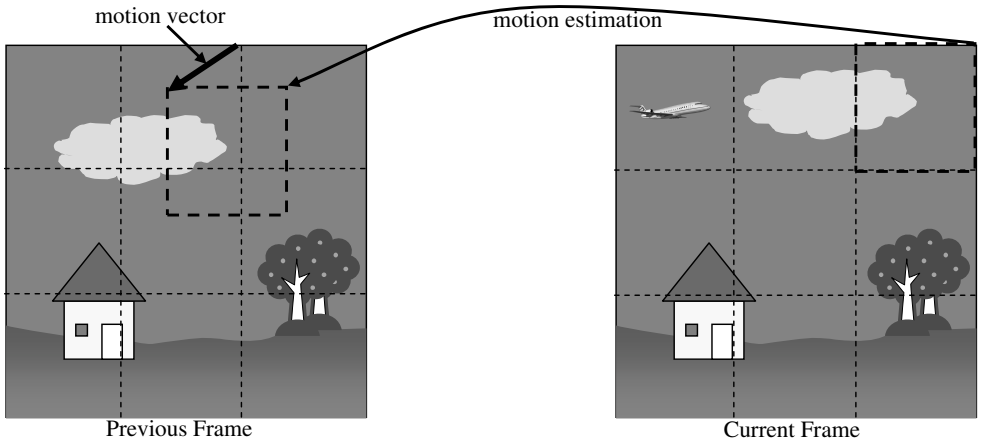


Figure 1.1: Motion estimation in Inter-frame coding.

P-coding achieves higher compression efficiency than I-coding by utilizing the temporal redundancy. However, random access of a P-coded frame is not possible since it depends on the previous frames. The computational complexity for motion estimation is high and losses in previous frames can propagate to future frames due to their dependency on the past frames.

Bidirectional-frame coding: In bidirectional-frame (B) coding, both past and/or future frames are used for motion estimation and compensation. If the future frames are used for reference, then this coding method requires a change in the order in which frames are encoded. The future frames that are used for

referencing must be encoded before the frames which makes use of them.

Fig. 1.2 shows a typical block diagram of a video encoding system [3]. Raw frames are input to the encoder. Each frames goes through a transform to remove the spatial redundancy, and motion estimation and compensation to remove temporal redundancy, before quantization and entropy coding. The frames are also reconstructed at the encoder that are stored in the frame memory for the motion estimation of the next frame. After the quantization, the frames are reconstructed using inverse quantization, inverse transform, and motion compensation.

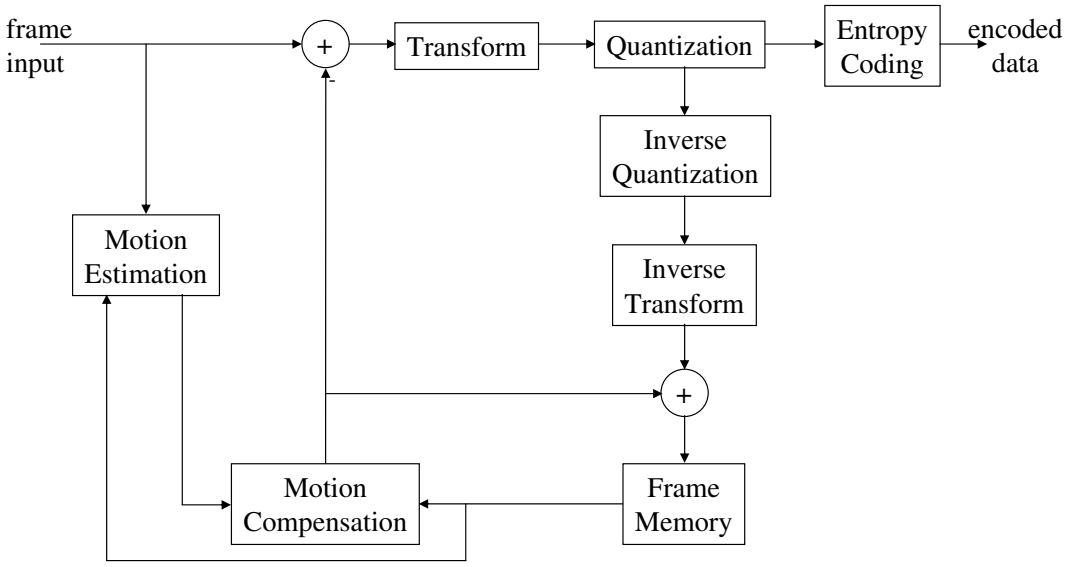


Figure 1.2: Block diagram of a video encoder.

1.2 Background on multiple reference frames

In multiple frame prediction, more than one past frame is used in the search for the best match block. In recent video standards such as H.264/AVC, up to 16 frames can be used for motion compensation and estimation. At the cost of extra memory storage and extra complexity for searching, multiple frame prediction has been shown to provide a clear advantage in compression performance. Papers on multiple reference frame motion compensation include [4–10]. Apart from improving compression efficiency, multiple reference frame prediction is also used for

improving error resiliency [11].

1.2.1 Dual-frame video coding

In dual frame video coding [12–19], which is a special case of multiple frame prediction, two frames are used for inter prediction, a short-term reference (STR) which is usually the immediate past frame, and a long-term reference (LTR) which is some frame from the more distant past. High encoding efficiency can be achieved by such dual-frame video coding [11]. An example of dual-frame coding is shown in Figure 1.3. Both encoder and decoder store LTR and STR frames. For encoding frame n, the STR is frame $n-1$ and the LTR frame is frame $n-k$, for some $k > 1$. The LTR frame can be chosen is several possible ways, including continuous update and jump update. In continuous update, k remains constant, which means that there is a constant distance between the current frame and the LTR frame. As the current frame moves forward by a frame, the LTR frame is also moved forward by one frame. So, all the frames are used as both STR and LTR. In jump update [8], the LTR frame remains the same for encoding N frames, then jumps forward by N frames and again remains the same for encoding the next N frames. In such an approach, every frame serves as an STR, but only every N^{th} frame serves as an LTR.

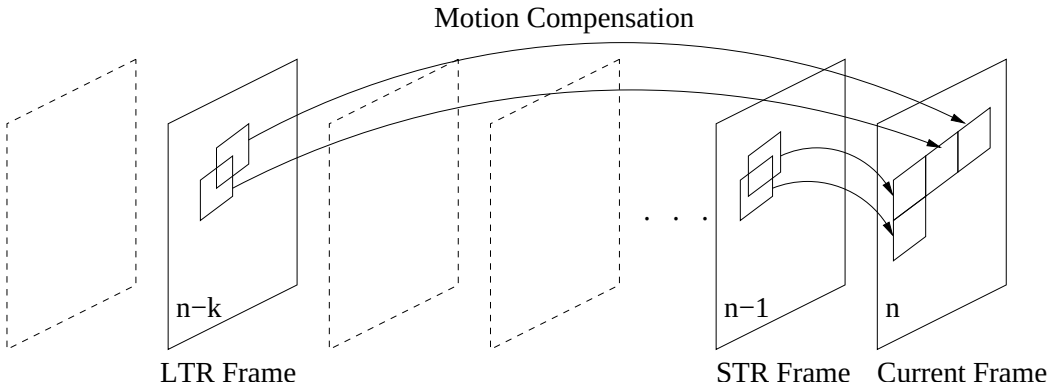


Figure 1.3: Dual-frame video coding with LTR and STR frames for motion compensation.

When both methods are deployed in a simplistic way, continuous update performs better than jump update. However, since very few frames are used as

LTR frames in jump update, this feature allows the use of high quality LTRs which are allocated more bits than regular frames. This enhanced the quality of the entire sequence [15, 20]. In [15], the assumption was that certain frames could be starved of bits so as to generate high quality LTR frames at regular intervals, provided that a long-term average bit rate constraint was met. This method improved the average video quality by 0.6 dB over a regular dual-frame encoder in which all frames were given equal importance. In [20], dual-frame coding was considered in a cognitive radio scenario consisting of a low bandwidth channel periodically supplemented by the brief rental of a substantially larger bandwidth. Here, a high quality frame was formed from the extra bandwidth and used as an LTR frame for some time.

1.3 Background on bit-rate allocation for multiple video streams

Fig. 1.4 shows independent rate control for multiple video streams. Rate control refers to the process of assigning bits to a frame in a video sequence based on, for example, its complexity, motion, type of frame encoding, and available bit-rate. Each encoder generates a variable bit-rate stream. Each encoder maintains a separate output buffer to convert its output stream to a constant bit-rate stream. Based on the output buffer fullness and the complexity of the frames, each encoder maintains a separate rate control path to encode its video. All the bitstreams are multiplexed together and transmitted through a constant bit-rate channel. At the decoder, the bitstreams are demultiplexed and each bitstream is sent to the input buffer for its decoder. Each decoder sequentially fetches its bitstream from its input buffer, decodes the frame, and sends it to an output display buffer.

Fig. 1.5 shows a method of joint bit-rate allocation for multiple video streams. Here, an output buffer is shared by all the video encoders. All the videos are encoded separately and each encoder passes some information to a central controller. Based on the information from each encoder about its video complexity and the status of the combined output buffer, the central controller decides the

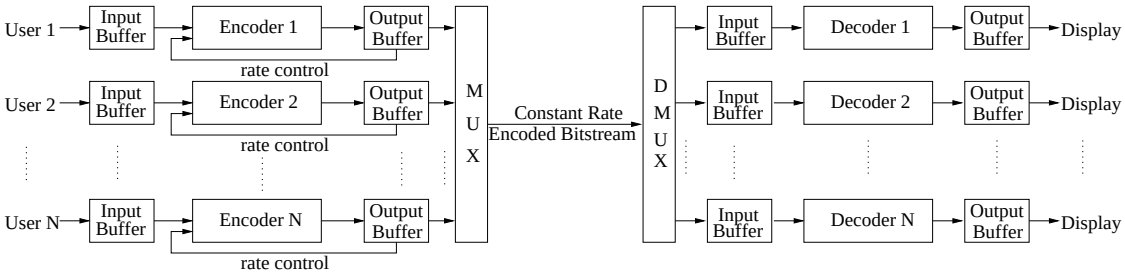


Figure 1.4: Multiple video streams with separate output buffers and rate control paths for each user.

number of bits that should be allocated to each video stream. Each video encoder uses this information to encode its video which is then stored in the output buffer. The output buffer sends the encoded bitstream through a constant bit-rate channel to the decoder. At the decoder, the bitstream is buffered and demultiplexed. Each bitstream is decoded separately and sent to its output display buffer.

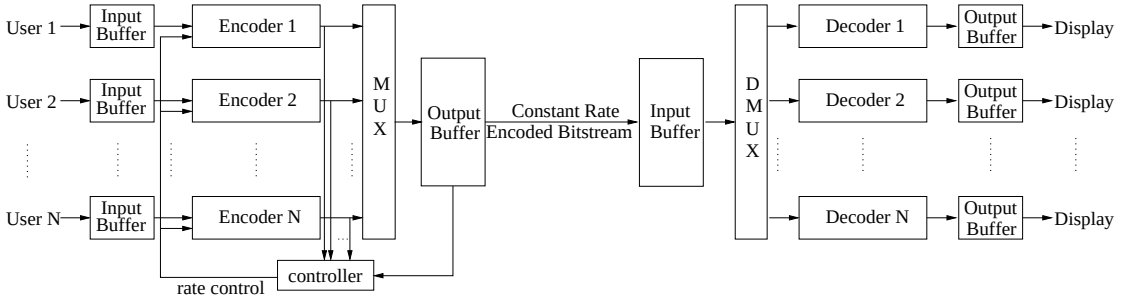


Figure 1.5: Bit-rate allocation for multiple video streams with a common output buffer and rate control path for all the users.

Rate control algorithms for encoding video streams independently were extensively studied [21] but joint bit-rate allocation has been widely used to improve the overall quality for multiple video streams [22–29]. A multi-camera surveillance system was considered in [22] where Peak Signal-to-Noise Ratio (PSNR) improvement was shown for transmitting video content only when there is any activity captured by a camera. However, it did not consider the case in which all cameras were capturing activity simultaneously. A distributed approach with high convergence time for transmitting multiple video streams was considered in [23] where the bit-rate allocation was done by link price which is updated using the subgradi-

ent method. A parallel encoder system with large delay and memory requirements was adopted in [24] where multiple streams are encoded with several bit-rates, and a combination of bit-rates for multiple streams was selected to maximize the PSNR. A superframe concept was used in [25] where one frame each from multiple video streams is combined into a superframe, and a Quantization Parameter (QP) is found based on the relative complexity of the superframes to improve the overall PSNR. In [26], a better joint rate control algorithm was proposed for the superframe method. In [27], a resource allocation algorithm was proposed to reduce PSNR fluctuation while maintaining high PSNR using Fine Granularity Scalability (FGS). It reduces PSNR fluctuations but also reduces overall PSNR. A joint rate control algorithm to dynamically distribute the channel bandwidth among multiple video encoders was proposed in [29] with the objective of assigning approximately equal quality to all videos.

Three transform domain multiplexing methods were discussed in [28]: MINAVE, MINVAR, and S-MINVAR. The overall quality when averaged across all the videos was maximized in the MINAVE method but at the expense of reducing the quality for some videos. At the cost of a peak signal-to-noise ratio (PSNR) reduction, the MINVAR method in [28] was proposed to reduce the frame level video quality variance between various video streams. Further, using an encoder buffer, S-MINVAR in [28] was proposed to reduce the quality variance across both videos streams and across frames. While the bit allocation algorithm in [28] is a good method for minimizing the quality variance between the videos, it comes at the expense of substantially reducing the average video quality, and also not all the users improve their individual video quality.

Mechanism-based resource allocation for multiple video streams was studied in [30,31]. These methods use a central controller for resource allocation. In [30], a Groves mechanism was used to control a network comprised of selfish users. Under the mechanism, a user's cost for receiving his share of resources depends on the information transmitted to the central controller, and it was shown that a user will report his true values for receiving his allocated share of resources. However, the overall quality of the system was improved at the expense of reductions of

video quality for some users. A bandwidth resource allocation procedure using the Nash and Kalai-Smorodinsky bargaining solutions was proposed in [31] for multiple collaborative users, and some important properties of the bargaining solution were presented for effective multimedia resource allocation. It was shown in [31] that the utility of all the users increased compared to a disagreement point defined by an initial resource allocation such that the initial utility is zero for all users. But, not all the available resource is allocated at this disagreement point. However, if the disagreement point is defined by any arbitrary allocation of the *total* available resource, then the method given in [31] will converge only to the disagreement point.

All the above methods use a central controller for resource allocation. Some decentralized resource allocation methods were proposed in [32, 33]. The Vickrey-Clarke-Groves mechanism was used to allocate resources in [32] and a pricing mechanism for resource allocation was used in [33]. In both these methods, the emphasis was on inducing truth-telling behavior from various users based on a mechanism that adjusts the trade-off between the value of additional resources and a cost levied to induce honest information transmittal. It is interesting to note that, in [33], a very special utility function is assumed for all users where a user's utility is a linear function of the video quality and 'money', which implies every user values an incremental change in his video quality exactly the same - a very strong and arguably unrealistic assumption. Moreover, in [33], an improvement in each user's utility was shown compared to the initial utility of zero, however, not all users would improve their utility if the total amount of resources were initially allocated, similar to [31].

1.4 Thesis outline

In this dissertation, we discuss the bit-rate allocation methods for multiple video streams given a channel bit-rate constraint. First, we investigate improving the compression efficiency of dual-frame video coding by carefully selecting the LTR frame and the number of bits given to such frames, and discuss rate control

for dual-frame video coding. We use dual-frame video coding in bit-rate allocation for transmitting multiple video streams with the objective of improving the average video quality.

Achieving higher quality averaged across all the video streams seems to be a reasonable objective for a group of video users who share a common goal. In such a case, some high complexity videos will improve their quality at the expense of the users with low complexity video. On the other hand, if each video user is independent, then users with low complexity videos would choose not to participate in a joint allocation scheme as they would have the expectation of lowering their quality by doing so. To persuade independent users to participate in joint bit-rate allocation, each user would expect to improve their video quality compared to the case when the videos are encoded separately.

Using competitive equilibrium bit-rate allocation, we propose algorithms which cause all the videos to improve or do at least as well as their individual video coding. In the competitive equilibrium bit-rate allocation, all the users transmit their rate-distortion (RD) information to a central controller which decides on the operating bit-rate for all the users. In another method, we use informationally decentralized bit-rate allocation where the users only transmit their bit-rate demand to an allocator based on their relative video complexity. For both of the approaches, we show that the video users improve their video quality compared to their individual bit-rate allocation.

In Chapter 2, we discuss various enhancements to dual-frame video coding. We use simulated annealing method for the selection of LTR frames in a video sequence. We also discuss rate-control for dual-frame video coding in this chapter.

In Chapter 3, we discuss bit-rate allocation for multiple video streams using dual-frame video coding. First, we modify the selection of LTR frames using motion activity. Then, we use dual-frame video coding with LTR frames to allocate bit-rate among various video streams in order to improve the average video quality.

In Chapter 4, we apply competitive equilibrium bit-rate allocation for multiple video streams. All the video streams send their RD information to a central allocator and, based on the video complexity for the current time and estimated

average complexity in the future, the central allocator decides the bit-rate given to each video user. All the video streams achieve higher quality by the competitive equilibrium method compared to individual encoding.

In Chapter 5, we present a decentralized bit-rate allocation method for multiple video streams in which the video users only send their bit-rate demand to an allocator instead of sending the entire RD information. The allocator computes the bit-rate price based on the total demand by the user and allocates the bit-rate to each user. The performance of the decentralized bit-rate allocation is comparable to the competitive equilibrium bit-rate allocation with less exchange of information about the video and reduced complexity for the central controller.

In Chapter 6, we summarize the contribution of this dissertation. This chapter also discusses various open problems from the dissertation and potential future work in this direction.

Chapter 2

LTR frame selection and rate control for dual-frame video coding

2.1 Introduction

In this chapter, we discuss two enhancements to dual-frame video coding. First, we develop a method to find good locations of LTR frames using Simulated Annealing (SA). We then discuss the design of rate control algorithm for dual-frame video coding with high-quality LTR frames.

This chapter is organized as follows: Section 2.2 describes the SA method for finding LTR frames for archived video and a window-based SA method for finding LTR frame positions under the constraint of lookahead window size. Section 2.3 describes a new rate control method for dual-frame video coding using a motion activity detection algorithm. The conclusions for the dual-frame video coding enhancements are discussed in Section 2.4.

2.2 LTR frame selection

In previous work on dual-frame video coding, evenly spaced LTR frames were considered, irrespective of the video content. As illustrated in Fig. 2.1, it is possible that some of the evenly spaced LTR frames may not be good references for future frames. Fig. 2.1 was generated by repeatedly encoding the QCIF size Mother-Daughter video sequence with one high-quality LTR frame each time. The x-axis shows the frame number that is being chosen as the one high-quality LTR frame, and the y-axis represents the percentage of macroblocks (MBs) of the following k frames (k = 20, 50, and 100) which choose to reference the LTR frame over the STR frame. For example, to generate the point on the top curve for x = 40 (frame number), we encoded the sequence with only frame 40 as a high quality LTR frame. We then counted how many MBs out of the next 20 frames (frame 42 to 61) referenced the LTR rather than the STR frame. As 9.2% of MBs referenced the LTR, this gives rise to the plotted point (40, 9.2) on the curve for next 20 frames. We found that all the frames are not equally useful as an LTR frame. The frames where we see the peaks (e.g., frames 34, 35, 36, 78) are more useful as LTRs than the frames in the valleys (e.g., frames 24, 25, 26, 60, 61). For example, consider the top curve in the figure which shows, for each possible LTR frame, the percentage of MBs in the next 20 frames which reference the LTR. The plot shows that when frame 78 is chosen to be an LTR, over 12% of the MBs of the next 20 frames prefer to reference it rather than the STR. Therefore, if frame 78 is chosen as an LTR frame, the video quality will likely be high. In contrast, when frame 127 is the LTR, only 3% of MBs in the next 20 frames choose to reference it, which means 97% of the MBs find a better match in the STR. Therefore, if frame 127 is used as an LTR frame then it will not improve the video quality much. So, if we take LTR frames at regular intervals and give them high quality, then they would be ineffective if they fell in such valleys. Note also that, in Fig. 2.1, the curve for 'next 20 frames' is almost always above the curve for 'next 50 frames'. This suggests that, as we move away from the LTR frames, the percentage of MBs using the LTR frame decreases. So, the effect of the LTR frame fades with time.

A method for LTR frame selection was studied in [34] using color layout

descriptors which assumes a large frame buffer at the input to the encoder and the decoder to preselect the possible LTR frames. It requires either a standard incompatible bitstream if the descriptions are sent to the decoder, or an increase in complexity at the decoder to generate these descriptions.

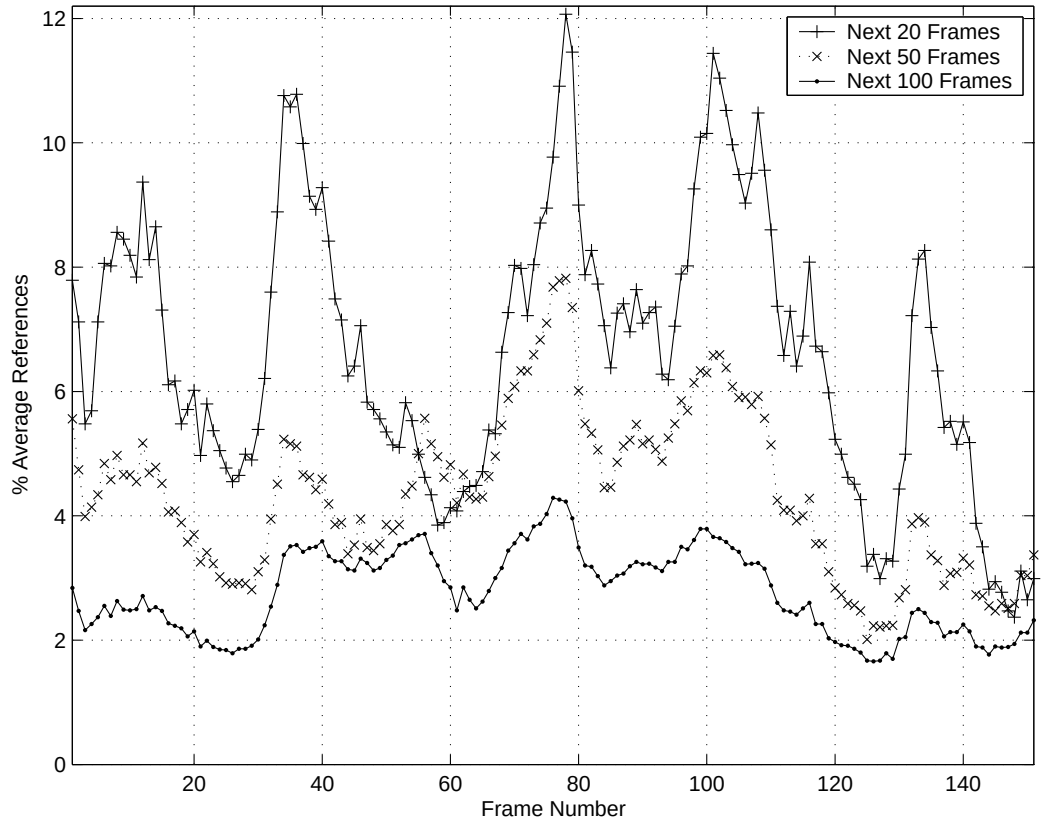


Figure 2.1: Percentage average references to a frame when it is created as a high-quality LTR frame in Mother-Daughter video. 'Next 20 Frames' shows the effect of an LTR frame on following 20 frames using it.

In the following sections, we aim to find a set of frames in a video sequence that can serve as good LTR frames. We use simulated annealing (SA) for the LTR frame search. We consider two scenarios: (1) when the entire video sequence is available for the LTR frame search at the encoder and (2) when there is a constraint on the size of the lookahead window. The first scenario is solely done offline for the transmission of archived video while the second scenario can be done in real-time assuming that the encoder is computationally efficient and that a small delay can be tolerated. We show that overall video quality can be improved by proper

selection of high quality LTR frames instead of just choosing them at regular intervals. SA is one method to find such good LTR frames, but other optimization methods could be applied. We note that when the LTR frames are afforded too much quality (too many bits) compared to the other frames which are not used as long-term motion compensation references, there can be an annoying pulsing of quality that is visibly perceptible. However, the slight increase in quality for LTR frames used in this work does not lead to visible quality pulsing, and raises the perceived quality of the entire video sequence. Our modifications are done only at the encoder, and produce a standard compatible bitstream.

2.2.1 Simulated Annealing for LTR Frame Search

Algorithm 1 Simulated Annealing algorithm for finding best LTR frame positions in a video sequence.

```
Initialize ltr_dist, total_ltr, num_iter, swing_width, thr_accept, ϵ
currLTRpos ← LTR frame position at regular ltr_dist, starting from frame 1
currPSNR ← PSNR of video sequence using currLTRpos
while swing_width ≥ 1 do
  for iter = 1 to num_iter do
    for ltr_frame = 1 to total_ltr do
      newLTRpos ← Randomly pick one LTR frame from remaining undisplaced LTR frames
      in currLTRpos and displace selected LTR frame randomly within ±swing_width of its
      current position (remaining LTR positions remain same)
      newPSNR ← PSNR of video sequence using newLTRpos
      if newPSNR ≥ (currPSNR - thr_accept) then
        currPSNR ← newPSNR
        currLTRpos ← newLTRpos
      end if
    end for
  end for
  Reduce thr_accept by ϵ
  Decrement swing_width by 1
end while
Display final LTR frame positions and PSNR
```

Simulated annealing (SA) [35] is an optimization process derived from the

physical process of cooling molten material down to the solid state. It is a stochastic method to avoid getting stuck in local minima, when searching for the global minimum. It has been proved that SA will converge to a global minimum in infinite time [35]. SA has been widely used for various combinatorial and other optimization problems [36]. SA starts with an initial solution that can be generated either randomly or using some known solution. A constraint-based new solution is then generated. If the new solution is better than the current solution, it is accepted unconditionally and becomes the next current solution. If, however, the new solution is worse than the current solution, it is not rejected outright, but is accepted with a certain probability. At the beginning, to avoid a local optimum, the probability of acceptance of a worse solution is kept high. As the simulation progresses, the probability is lowered according to some pre-defined schedule, and after some point, a new solution is no longer accepted unless it is better than the current solution.

We use SA for LTR frame choice in a video sequence for dual-frame video coding. If we know the video characteristics as shown in Figure 2.1 for the Mother-Daughter sequence, then we can pick the peaks as our initial solution. However, since generating such characteristics is computationally intensive, we instead choose our initial solution by creating high quality LTR frames at a uniform interval of *ltr_dist*, starting from the first frame. Evenly spaced high quality LTR frames were used in [15] which is the reference point for comparing our results. Then one of the current set of LTR frames is randomly selected and is replaced by a new frame which is also randomly selected in the range of $\pm$*swing_width* from its original position. The new arrangement of LTR frames is accepted as the new current solution if the average PSNR of the video sequence is no less than *thr_accept* below the PSNR of the current solution. Otherwise, the new solution is rejected. We then randomly choose another LTR position from among those not yet perturbed on this round, and repeat the same process. After we have gone through all *total_ltr* LTR frame positions in some random order, we have completed one iteration. After completing *num_iter* such iterations, we reduce *swing_width* by one step and *thr_accept* by ϵ amount and continue with the next round of itera-

tions. When *thr_accept* becomes zero, we stop accepting inferior solutions. The simulation stops when *swing_width* becomes zero. The position of the high quality LTR frames at the end of the simulation is our final solution. The brief outline of this algorithm is presented in Algorithm 1.

Results

We modified H.264/AVC [37] reference software JM 9.6, obtained from [38]. We used the 4:2:0 QCIF (176×144 pixels) video sequences Foreman, Carphone, Container, Mother-daughter (M-D), and Claire at 30 fps for our simulations. SA was performed on 200 frames with the first frame intra-coded and the remaining frames inter-coded. A lossless channel was assumed with a constant average bit rate of 58 kbps. The initial LTR frame position was chosen at a regular interval (*ltr_dist*) of 25 frames starting from the first frame. So, there were a total of 8 high quality LTR frames (*total_ltr*). Parameter *swing_width* was initialized to 5 and each LTR frame position was iterated $num\_iter = 4$ times for every value of *swing_width*. We initialize $thr\_accept = 0.04$, which was found empirically, as the PSNR decrease that could still be accepted. *thr_accept* was reduced by $\epsilon = 0.01$ whenever *swing_width* was reduced by 1.

Figure 2.2 shows the results for different test video sequences. We ran six SA simulations for each video sequence. The three bars for each video sequence show the average, maximum, and minimum PSNR improvement for these six runs over the PSNR achieved by using evenly spaced LTR frames. For the Carphone video sequence, the average PSNR improvement of six SA simulations is 0.5 dB over the evenly spaced LTR frames, with the highest improvement of 0.6 dB and the lowest improvement of 0.4 dB. Best results were obtained for the M-D video sequence where the average improvement by using SA is 0.7 dB with the highest improvement of 0.8 dB and the lowest improvement of 0.6 dB. The trend of the results at CIF resolution at 58 kbps for the M-D video (M-D CIF) is consistent with the results for QCIF video. Similar results were also found for QCIF resolution for the M-D video at 82 kbps (M-D 2). Both M-D CIF and M-D 2 are shown in Figure 2.2.

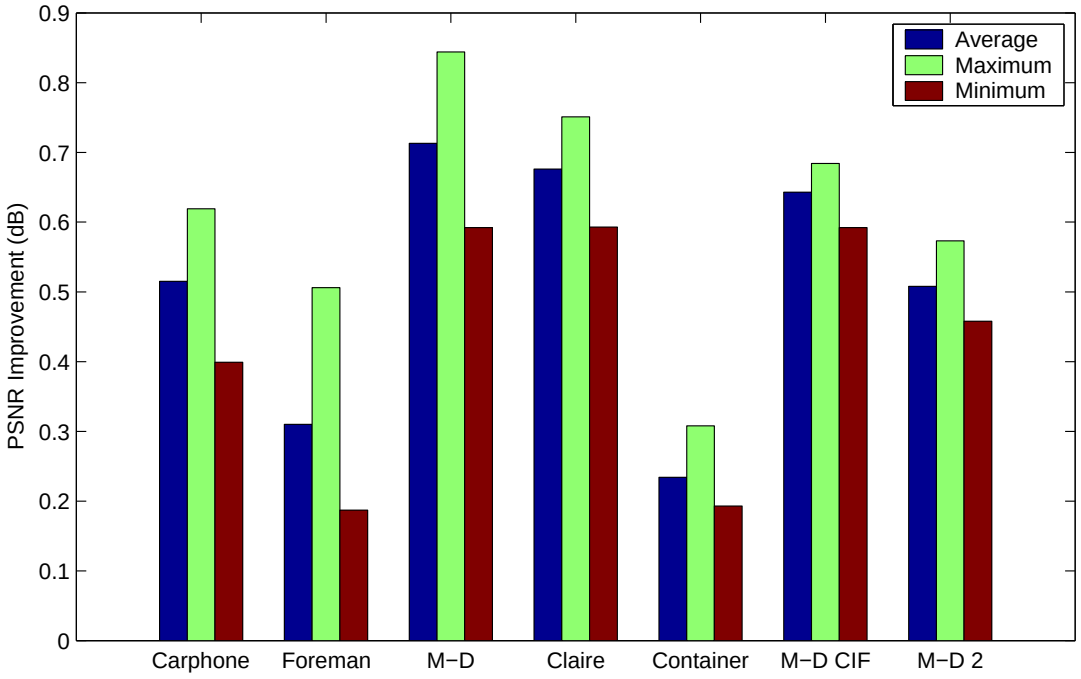


Figure 2.2: Improvement over evenly spaced high quality LTR frames using the simulated annealing approach. Average PSNRs for the sequence with evenly spaced high quality LTR frames are 33.0 dB for carphone, 29.0 dB for Foreman, 37.4 dB for M-D, 41.6 dB for Claire, 38.1 dB for Container, 33.8 dB for M-D CIF, and 40.0 dB for M-D 2.

As an example of the frame selection: For the Claire video sequence, evenly spaced LTR frames are 0, 25, 50, 75, 100, 125, 150, and 175. One SA run produced a PSNR gain of 0.7 dB over evenly spaced LTR frames and chose the final LTR frames 8, 32, 59, 77, 110, 122, 146, and 167. Five of the six SA runs had frames 32, 77, and 146 in their final LTR sets, suggesting that these frames are particularly useful as LTR frames. After frames 32 and 146, the video content moves very slowly. So, having these frames as high quality LTRs improves the PSNR of subsequent frames through long-term as well as subsequent short-term references. In general, SA selects one of the first few frames of a low motion part in a video sequence and assigns it as a high quality LTR frame. SA tries to avoid assigning a high quality LTR in a high motion part of a video sequence because the video content changes rapidly and a high quality LTR would not be useful long. This is also the reason for getting higher PSNR improvement for low motion sequences such as Mother-Daughter and Claire compared to the higher motion sequences such as Foreman under the constraint of having the same number of LTR frames. In Claire, the video is constant for around 15 frames after frame 77 and then the face moves rapidly causing SA to avoid assigning new LTR frames. Therefore, the next LTR frame comes around frame 110 when the high motion part is over, resulting in a longer LTR frame distance than the average.

In the container video sequence, a ship moves slowly into the ocean. It was shown in [15] that this sequence has the largest gains for evenly spaced high quality dual-frame coding over regular quality dual-frame coding among all the video sequences tested. Because of the rather constant motion between the ship and the camera, evenly spaced LTR frames do well. For a LTR spacing of 25, it showed about 0.8 dB PSNR improvement over regular quality evenly spaced LTR frames. As we can see in Figure 2.2, this video sequence gives the least improvement using SA over evenly spaced LTR frames. Since there is no significant change in motion, the importance of all the frames is almost the same. When we make a plot similar to Figure 2.1 for this video sequence, it produces an almost flat number of references to any frame in the video sequence. Therefore, while the evenly spaced high quality LTR frames produce a big PSNR gain compared to evenly spaced

regular quality LTR frames, further change in LTR position will give just a small additional PSNR improvement. Conversely, while only 0.3 dB improvement was achieved for the Claire video sequence in [15] for evenly spaced high quality LTR frames with spacing of 25 compared to regular quality dual-frame coding, we were able to achieve a further 0.7 dB of PSNR improvement on top of the evenly spaced LTR frames. In general, more than 1.0 dB PSNR improvement is achieved over evenly spaced regular quality LTR frames in dual-frame video coding by using both high quality LTRs [15] and uneven spacing of LTRs as discussed in this section. The PSNR improvement is achieved with a high computational complexity which is on the order of $swing\_width \times num\_iter \times num\_ltr \times$ length of input sequence.

2.2.2 Window-based approach for LTR frame search

The PSNR improvement achieved in Section 2.2.1 assumes that the encoder has access to all 200 frames of the video sequence in advance. It is good for broadcast video where long encoding delay is possible but is not suitable for real-time or near real-time applications. For long video sequences, it requires huge memory to store the input video and also a large amount of computational resources. Basically, both memory requirement and computations increase with the length of video sequence and at some point it is impossible to perform the method discussed in Section 2.2.1. To overcome this problem, we propose a window-based heuristic approach to find LTR frame positions. Here, we select LTR frame positions one at a time, thereby upper bounding the size of the input video buffer and the computational resources needed. Assuming sufficient computational resources, this approach can be used for real-time video transmission with a small encoding delay.

Identifying a good LTR frame within a lookahead window involves careful selection of the location of such a window and a search range within the window. Figure 2.3 shows the average percentage of a frame that references the LTR frame (y-axis) as a function of the distance back to the LTR frame position (x-axis) for all the five video sequences. Each of the first 150 frames of each video was sequentially selected as a high quality LTR frame and the number of references

made to this LTR frame was observed over the next 100 frames (and averaged over the 150 frames), except for the frame next to the LTR frame. From this figure, we can clearly see that the importance of LTR frames decreases with increasing frame distance. The curve is not necessarily monotonically decreasing since the importance of each frame as a reference is different, as discussed previously. One approach would therefore be to assign frequent LTR frames. However, we must limit the number of LTR frames (because each one requires more bits than a typical frame).

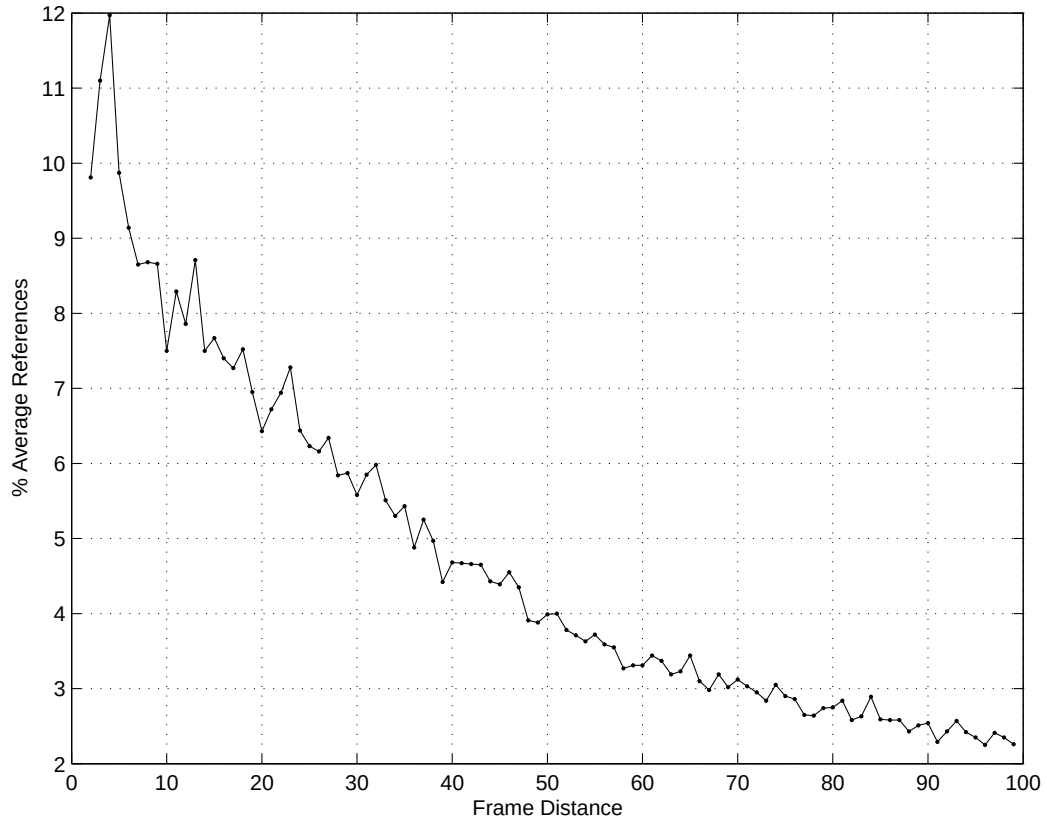


Figure 2.3: Average percentage references to a high quality LTR frame as a function of frame distance.

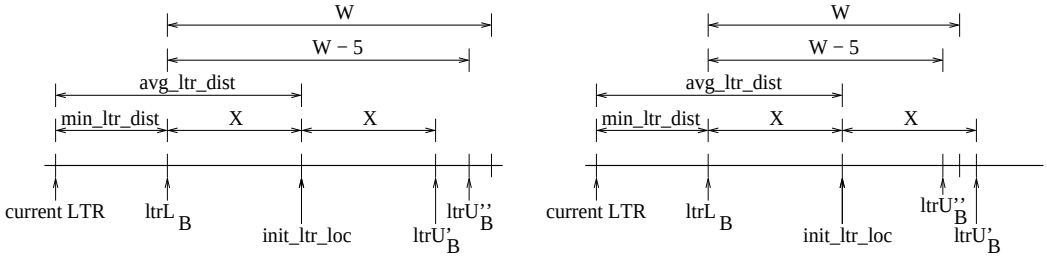
We define the average LTR distance (*avg_ltr_dist*) at any given time to be the ratio of the number of frames remaining to be encoded to the number of LTR frames remaining to be created. Initially, *avg_ltr_dist* is 25 (200 frames to be encoded using 8 LTR frames). Experimentally, for *avg_ltr_dist* of 25 frames, we

found that keeping an LTR frame for at least 15 frames provides a good quality. We denote this distance as min_ltr_dist. We keep an LTR frame for a minimum of min_ltr_dist frames, and after that, we begin to look for the next LTR frame position. Therefore, the lower boundary for the current LTR frame search (frame number $ltrL_B$) is min_ltr_dist from the previous LTR frame position. We recalculate avg_ltr_dist after choosing each LTR frame position. In general, we set $min\_ltr\_dist = \max(avg\_ltr\_dist - 10, 0)$, so that min_ltr_dist increases if avg_ltr_dist increases (that means we are getting frequent LTR frames) and vice versa. This reduces the chances of getting LTRs too close to each other. The next LTR frame position is initialized at avg_ltr_dist from the previous LTR frame position and its frame number is $init\_ltr\_loc$. We search for the LTR frame position starting from frame $ltrL_B$, keeping frame $init\_ltr\_loc$ in the middle of the search range by extending the search range to the same number of frames ($X = init\_ltr\_loc - ltrL_B$) on the other side of frame $init\_ltr\_loc$. We denote the upper boundary of the search range as $ltrU'_B = init\_ltr\_loc + (init\_ltr\_loc - ltrL_B)$.

But the upper boundary of the search range is also dictated by the size of the lookahead window. Experimentally we found that we need at least 5 frames after the LTR frame to observe its effect, so we want to create an LTR frame 5 frames or more back from the end of the lookahead window so that we have at least 5 frames over which to judge whether it is a useful LTR frame or not. If W is the size of the lookahead window and it starts from frame $ltrL_B$, then the upper boundary is restricted to frame $ltrU''_B = ltrL_B + W - 5$. Therefore, the upper boundary of the LTR search range is given by $ltrU_B = min(ltrU'_B, ltrU''_B)$. Figure 2.4 depicts both the cases of LTR frame search range where (a) $ltrU'_B < ltrU''_B$ which means that the search range is not restricted by the size of the lookahead window, and (b) $ltrU'_B > ltrU''_B$ when the upper boundary of the search range is restricted by the size of the lookahead window.

The process of searching for one LTR frame in the specified range is done using SA as described in Section 2.2.1. Once an LTR frame is found, we recalculate avg_ltr_dist, $ltrL_B$, and $ltrU_B$. We then move on to find the next LTR frame position using the same approach. We repeat this process until the end of the

video sequence.



(a) $ltrU_B^{'} < ltrU_B^{''}$ (b) $ltrU_B^{'} > ltrU_B^{''}$

Figure 2.4: LTR search range (a) $ltrU_B^{'} < ltrU_B^{''}$, (b) $ltrU_B^{'} > ltrU_B^{''}$, where $ltrU_B^{'} = init\_ltr\_loc + (init\_ltr\_loc - ltrL_B)$, $ltrU_B^{''} = ltrL_B + W - 5$, and $X = init\_ltr\_loc - ltrL_B$. $ltrU_B = min(ltrU_B^{'}, ltrU_B^{''})$.

Results

Let W be the number of frames in the lookahead window. These frames are assumed to be available at the encoder but are not yet encoded. After determining one LTR frame location, for computing the next LTR frame location, the lookahead window starts at frame $ltrL_B$ and extends to frame $ltrL_B + W - 1$. All the frames before frame $ltrL_B$ are assumed to have already been encoded. Frame $init\_ltr\_loc$ is first selected as an LTR frame to calculate the PSNR for all W frames in the lookahead window. Then the same SA procedure is applied to select the best LTR frame in the search range, where the search range is between $ltrL_B$ and $ltrU_B$ in these W frames as described above. Once an LTR frame is selected, we calculate the new avg_ltr_dist, $init\_ltr\_loc$, $ltrL_B$ and $ltrU_B$ and repeat the procedure. Since the LTR search range is quite small compared to the range in Section 2.2.1, we initialize the $swing\_width$ to 4 and num_iter to 2, thereby reducing the computational complexity by reducing the number of searches for an LTR. Assuming the same variation of PSNR by changing an LTR frame position, we initialize $thr\_accept = \frac{8}{num\_frm}$, where num_frm is the number of frames in the LTR search range ($ltrU_B - ltrL_B + 1$), and it is reduced to 0 in $swing\_width$ steps ($\epsilon = 0.25 \times thr\_accept$).

Figure 2.5 shows the average PSNR improvement for various test video

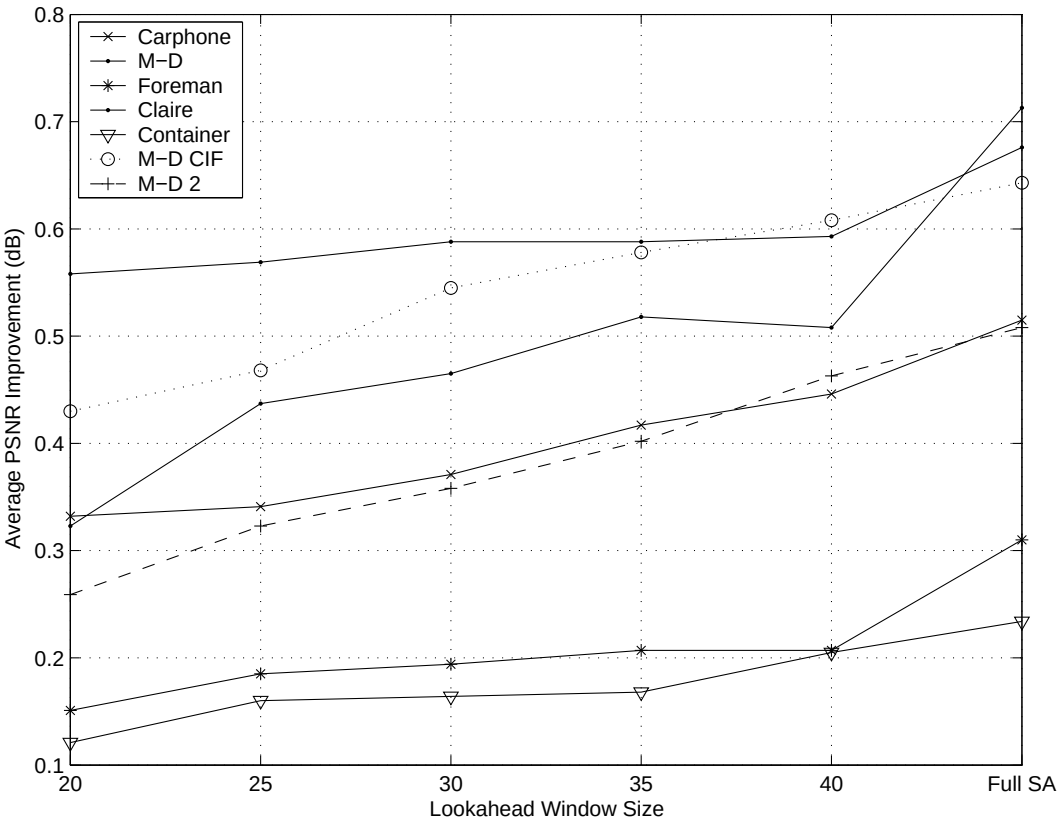


Figure 2.5: Variation of PSNR improvement over evenly spaced LTR frames as a function of the lookahead window size.

sequences over evenly spaced LTR frames as a function of W which was varied from 20 to 40 in steps of 5 frames. The PSNR improvement was averaged over eight simulations for each W in a video sequence and compared with the results from Section 2.2.1, shown here as 'Full SA'. As discussed in the previous section, we found that the improvement for the Container video sequence remains almost flat for various window sizes and is very close to 'Full SA'. Claire and Carphone have ample choices for LTR frames and so these videos are also insensitive to the lookahead window size. Figure 2.1 for the Mother-Daughter video sequence shows narrow peaks and, for small lookahead window sizes, sometimes we miss these peaks for LTR selection. In general, even with a small lookahead window, we achieve significant PSNR improvement over evenly spaced LTR frames. The computational complexity is on the order of $swing\_width \times num\_iter \times num\_ltr \times (ltrU_B - ltrL_B)$. With the reduction in $swing\_width$, num_iter, and LTR search range, the computational complexity in the window-based approach for finding LTR frames is drastically reduced compared to the full LTR search and is feasible for real-time implementation.

2.3 Rate-control for dual-frame video coding

While a performance improvement is achieved using dual-frame video coding, a price is paid in larger delay buffer to accommodate the high quality LTR frames. In real-time video transmission, the amount of delay is limited. For example, the maximum delay that can be tolerated in video telephony is less than 300ms. Therefore, high quality LTR frames may pose a challenge for using dual-frame coding in real-time video transmission.

Rate control for video coding has been extensively studied, such as [39, 40]. A buffer constrained rate control algorithm for H.264 in [40] uses a pre-analysis unit to accurately achieve the target bit-rate. However, rate control for dual-frame video coding is largely untouched. In particular, assignment of high quality to LTR frames presents difficulties for rate control. One can reserve a portion of the buffer to accommodate LTR frames but this reduces the buffer usage for other frames. In

this section, we examine a rate control method for dual-frame video coding with a delay buffer constraint. This work is inspired by the rate control used in [17] where separate rate control was implemented for both regular and high quality LTR frames.

In this section, we propose a buffer threshold strategy to accommodate large size LTR frames. For reducing loss due to buffer overflow, we use a buffer threshold for quantization parameter (QP) adjustment that limits the buffer usage. We use a motion activity detection algorithm to determine the number of bits for a high quality LTR frame. The proposed method outperforms the standard H.264 rate control [41] and the rate control for dual-frame video coding proposed in [17] even when a modification for reducing the loss due to buffer overflow is incorporated in both these methods. Note that the typical rate control algorithms proposed for H.264 do not perform well because those algorithms are not designed to handle the extra bits for the LTR frames in dual-frame video coding.

2.3.1 Video encoding delay

Delay at the encoder comes from input buffer delay, encoder processing delay, and output buffer delay as shown in Figure 2.6. We use I-P-P-P coding format where the frames are either Intra (I)-coded or Inter (P)-coded. The I-frames are encoded independently while the P-frames use only previously encoded frames for reference. Consequently all frames are processed sequentially. Therefore, there is a constant input buffer delay of one frame for any video stream. If we use the Bidirectional (B)-coded frames where the B-coded frames use both past and/or future frames for referencing, then we need to buffer the future frames that are used as a reference in order to encode the B-frames and that will increase the size of the input delay buffer. The processing delay is platform dependent and, for the purpose of rate control, we ignore this delay. The encoder generates a variable size of encoded bitstream for each frame while we assume transmission at a constant bit-rate. Therefore, we need to store bits in an encoder output buffer. The size of the output buffer controls the tightness of the rate control algorithm. For an end-to-end delay in video transmission, we also need to take into account

the propagation delay and the delays at the decoder which are the same as the delays at the encoder but in the reverse order.

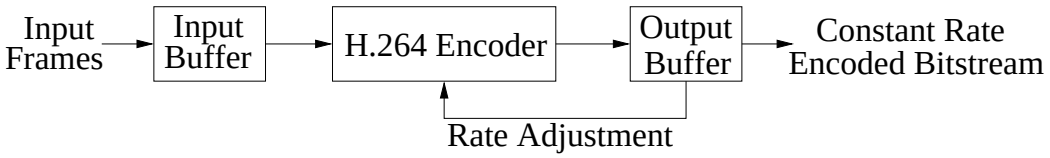


Figure 2.6: Delay components at the video encoder.

Let R be the transmission rate and let the video be encoded at f frames per second. If each frame is encoded with $\frac{R}{f}$ bits then we do not need any encoder output buffer. However, this leads to a very poor video quality since not all frames require the same number of bits. In practical scenarios, frames are assigned bits based on their relative complexities. Frame complexity is often estimated using mean absolute difference (MAD) which is the difference between the original frame and the predicted frame. Since the current frame is not yet encoded, the rate control algorithm recommended for H.264 [41] predicts the current frame MAD from the previously encoded frame MAD. Many algorithms are proposed in the literature for accurate MAD prediction, such as [42]. For a given target rate for the current frame and MAD, the QP is calculated using a quadratic rate-distortion (RD) model. However, it is difficult to predict the exact QP that will produce the encoded bits for a frame close to its target bits. This leads to the requirement of having an encoder output buffer that can convert variable encoder output rate to constant rate for transmission. With a buffer constraint, the frames (or part of a frame) that exceed the buffer limit are dropped. This leads to error propagation and video quality deterioration.

2.3.2 Bit allocation for LTR frames

In dual-frame video coding, one key issue is to allocate an appropriate number of bits to ensure a high-quality LTR frame. As seen in Section 2.2, for low motion videos, we should allocate many bits to the LTR frame since subsequent frames are similar to the LTR frame and will benefit from the high quality of the

LTR frame. For high motion parts, it is not desirable to spend many bits on an LTR frame because its higher quality will soon be useless as the subsequent frames rapidly become different from the LTR. We use the activity in a video to determine the quality given to an LTR frame.

To measure activity, we divide a frame into MBs of standard size (16×16 pixels) and calculate the pixel-by-pixel Sum of Absolute Differences (SAD) between each MB and the co-located MB in the previous frame.

For MB k in the current frame n ($MB_{n,k}$),

$$SAD(n,k) \;=\; \sum_{i=1}^{16}\sum_{j=1}^{16} |MB_{n,k}(i,j) - MB_{n-1,k}(i,j)| \tag{2.1}$$

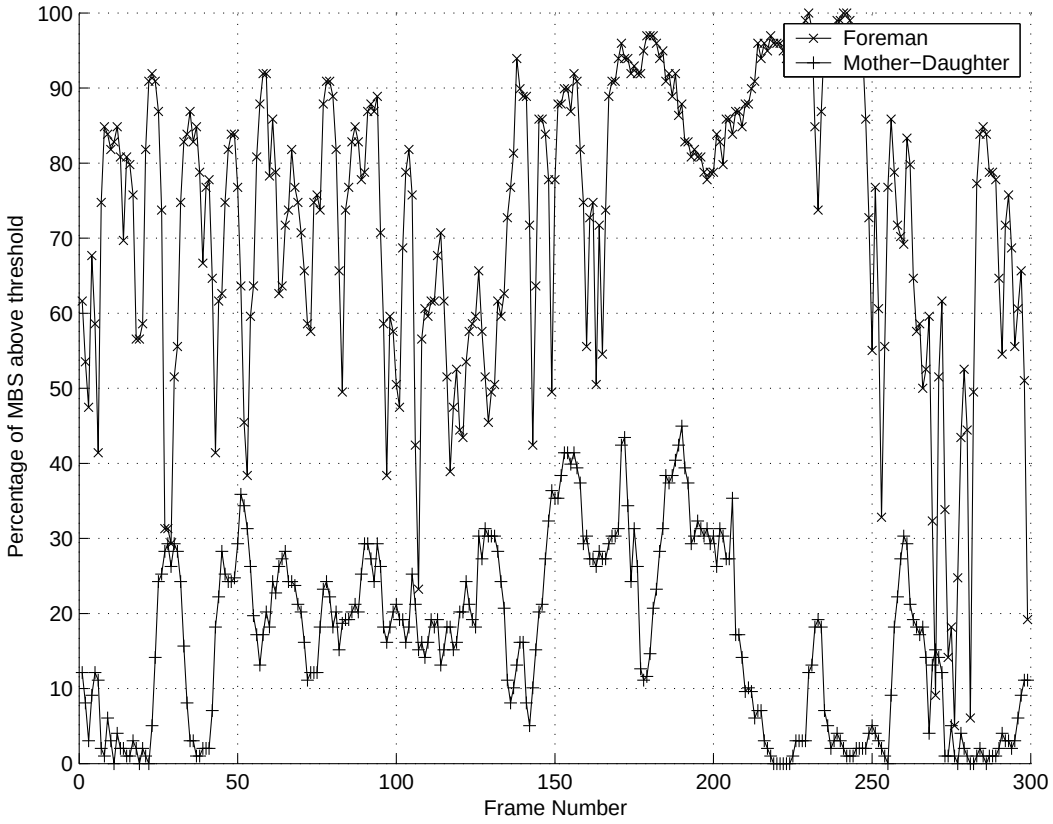


Figure 2.7: Percentage of MBs above the motion threshold of 500 for the Foreman and Mother-Daughter video streams.

The MB is considered to be active if $SAD(n,k) > T$, for some predetermined threshold T. We chose $T = 500$ after examining a range of thresholds for

various QCIF video sequences. Activity measurement is done in real-time. A similar method in [22] considered binary classification of activity for video surveillance. Note that motion vectors can also be used to perform motion activity detection. Figure 2.7 shows the result of activity detection for the Foreman and Mother Daughter videos. On the x-axis is the frame number and on the y-axis is the percentage of MBs above the threshold. It shows that Foreman is of higher motion than Mother-Daughter.

A larger number of extra bits (beyond those normally assigned to non-LTR frames) are assigned to an LTR frame for a low motion part of a video because the high quality of LTR frames will be retained for a long time. On the other hand, fewer extra bits are assigned to an LTR frame in a high motion part of a video because the high quality will soon be lost and a new LTR frame will soon be needed. To operate in real-time and avoid buffering future frames, we consider the motion of past frames to predict the motion of future frames.

Let m be the average fraction of active MBs in the 10 frames prior to an LTR frame. Based on m, the bit allocation for the LTR frame (LTR_bits) is given by

$$LTR\_bits = \begin{cases} 2 \times reg\_bits, & \text{if } m > 0.5 \\ 10 \times reg\_bits, & \text{if } m < 0.1 \\ (12 - 20 \times m) \times reg\_bits, & otherwise, \end{cases} \tag{2.2}$$

where reg_bits is the average number of bits assigned to a regular frame. These allocations and threshold values were determined experimentally and not carefully optimized for any particular video stream. Improvement was found for nearly all of the video streams [43] compared to having a fixed allocation of high-quality for LTR frames.

2.3.3 Delay buffer threshold for dual-frame video coding

With the addition of high-quality LTR frames in dual-frame video coding where many bits are assigned to the LTR frames, the chances of a portion of an LTR frame getting dropped is higher than for a regular frame. This also happens in

the rate control implementation given in [17] where two separate rate control paths, one each for the regular and LTR frames, were used by the encoder. In [17], the bit-rate for LTR frames was assumed to be three times that of the regular frames. The rate control implementation in H.264/AVC was used for both the paths and quality improvement was shown over video coding with two STR frames. Since the LTR frames are encoded with separate rate control, there may be cases where the quality of an LTR and its adjacent regular frames are similar, thus losing the importance of an LTR frame. The method of rate control for dual-frame coding in [17] uses the skip mode to drop MBs in case a frame would cause a buffer overflow. While this rate control method works well for high bit-rates, the LTR frames in low bit-rate coding suffer many MB drops due to their large size.

In our approach, we set a Buffer Fullness Threshold (BFT) for rate control using a delay buffer for encoding a video stream. Most of the time, BFT is set to bf_low, some predetermined fraction of total buffer size. If an LTR frame comes, then we increase BFT to a higher level (bf_high) because we know that LTR frames are assigned more bits compared to other frames. After the LTR frame, we slowly reduce BFT from bf_high to bf_low within bf_slope frames. This process is shown in Fig. 2.8. We use the rate control algorithm recommended for the H.264 in the JM implementation if the buffer fullness is below BFT, otherwise we increase the QP by 2. Note that bf_high, bf_low, and bf_slope were determined experimentally using a set of training videos and were not optimized for any particular type of video.

With this single rate control path to accommodate both regular and LTR frames, the LTR frames are of higher quality than adjacent frames yet we seldom need to skip MBs to avoid buffer overflow. The number of bits for a high-quality LTR frame is determined by motion activity as discussed in Section 2.3.2.

2.3.4 Results

The simulation was performed using JM 10.1 [38] reference software for H.264/AVC baseline profile. All the video sequences used in the simulation were 300 frames QCIF (176×144 pixels) at 30 fps. The distance between two LTR

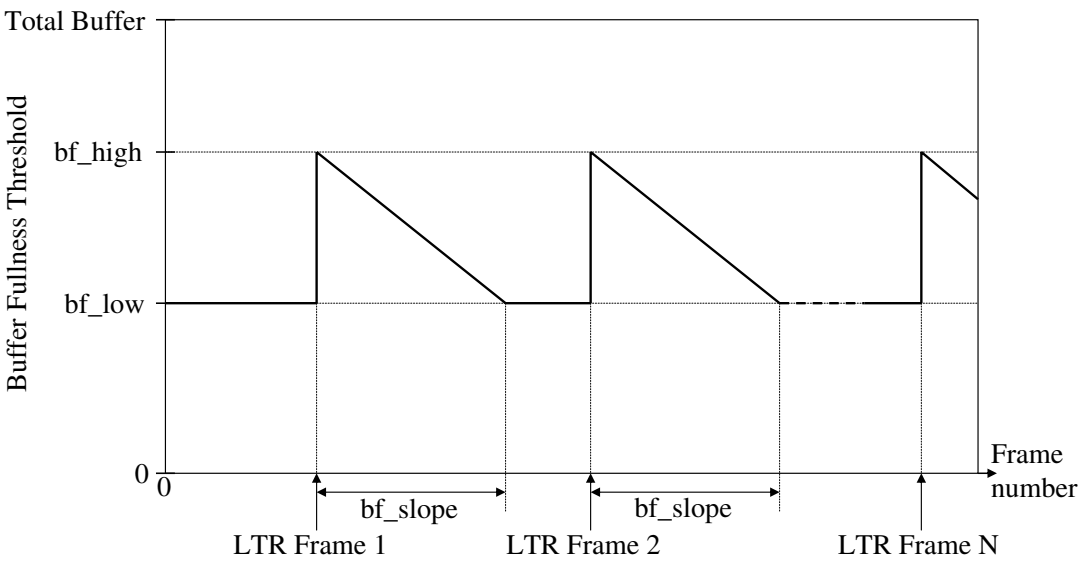


Figure 2.8: Buffer Fullness Threshold (BFT) for the proposed rate control algorithm to encode individual video streams using dual-frame video coding with high-quality LTR frames.

frames was 25 frames. We calculate the average MSE for each frame and across all frames of a video sequence, and then convert to PSNR for reporting our results.

To make an unbiased comparison, we modified the rate control algorithm in H.264/AVC by maintaining a buffer threshold in the encoder output buffer. If the buffer fullness exceeds this threshold, we adjust the QP to reduce MB losses. A similar modification is also applied in the rate control proposed in [17]. These improved rate control algorithms are then compared with our proposed rate control method.

For rate control in JM H.264 with two STR frames (**JM RC**) and the rate control used in [17] (**JM PULSE**), we keep the encoder output buffer threshold at 50%. This means we start adjusting the QP at this threshold for avoiding MB losses in the frame. If the frame size exceeds the buffer size, then we drop MBs using the skip mode. The skipped MBs are reconstructed using motion compensated prediction from the STR frame. Neighboring motion vectors are used to find the motion vector of the lost MB. In our work (**LTR BUF MGMT**), we keep bf_low at 40% and bf_high at 65% of the total buffer size. The bf_slope is 15 frames for the LTR distance of 25 frames.

Figure 2.9 shows the variation of PSNR with the encoder output buffer delay for News and Container videos at 18 kbps. In both videos, JM RC performs better than JM PULSE for smaller encoder buffer size because the chances of loss in the LTR frames due to buffer overflow are very high. Even with the QP adjustments, sometimes it is not possible to avoid MB loss. As expected, JM PULSE performs better than JM RC at larger encoder buffer size due to the advantage of high quality LTR frames over the two STR frames. By appropriately managing the buffer usage for LTR frames, MB losses are further reduced in LTR BUF MGMT thus improving the performance over JM PULSE. The performance is further boosted by choosing the appropriate number of bits for high quality LTR frames based on motion activity instead of using some average number of bits. Therefore, when averaged over the entire sequence, LTR BUF MGMT outperforms both other methods at all encoder buffer sizes. The frame-by-frame effect of high quality LTR frames can be seen from Figure 2.10. The LTR BUF MGMT curve is almost always above the other two curves. The pulsing of LTR frames is not perceptually visible. Similar results were found for Akiyo and Container videos at 36 kbps (the latter is shown in Figure 2.11).

2.4 Conclusion

In this chapter, we presented two main enhancements to dual-frame video coding. We first discussed a simulated annealing method to find good locations for LTR frames in a video sequence. We then proposed a rate control algorithm for dual-frame video coding with high-quality LTR frames, where the LTR quality was determined by the motion activity of the video sequence.

In the simulated annealing approach for finding good locations for high quality long-term reference frames, the experimental results show PSNR improvement of 0.2 dB to 0.7 dB for various test video sequences over evenly spaced high quality LTR frames. On combining our results with [15], more than 1.0 dB PSNR improvement was achieved over video encoding using regular quality evenly spaced long-term reference frames. The process of LTR frame selection was further

performed on a constrained lookahead window size in a long video sequence for real-time video transmission which reduced delay and computational complexity. For most of the video sequences, the PSNR improvement in this case was close to the PSNR improvement when the whole video sequence was considered. In both cases, changing the parameters such as the bit rate (50 kbps to 100 kbps), length (100 to 300 frames) or resolution (QCIF and CIF) of the video sequence, or the number of LTR frames (5 to 8) produces similar results.

In the rate control for dual-frame video coding, we proposed a a method for handling high quality LTR frames in dual-frame video coding for buffer constrained real-time video communication. There are two major contributions for the rate control for dual-frame video coding:

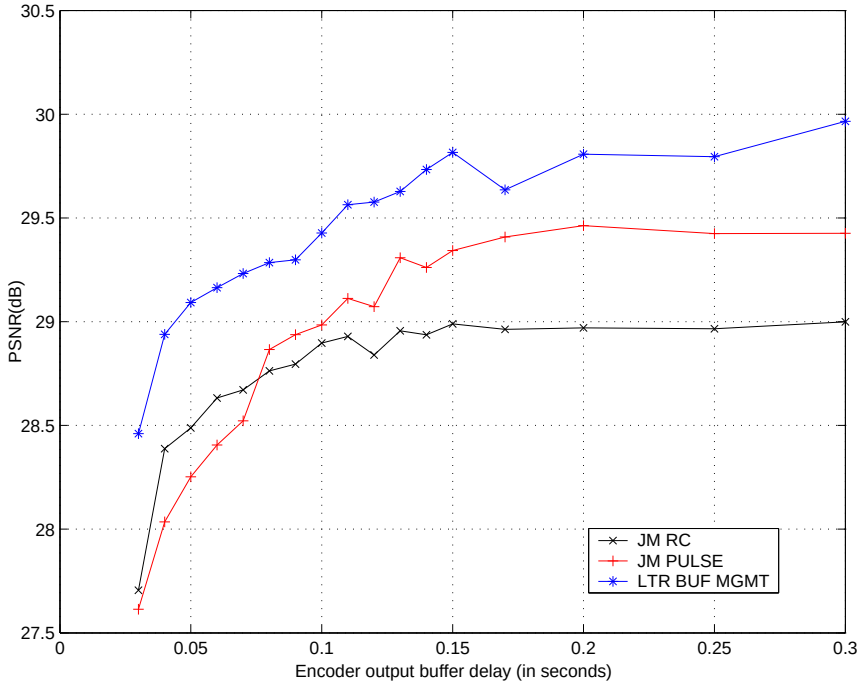
1. The number of bits, and therefore the quality level, to be assigned to an LTR frame can be determined using a simple video activity measure, and this performs better than the previous work which allocated a fixed number of bits to the high-quality LTR frames.

2. High-quality LTR frames require more bits on average than regular frames, and standard approaches for buffer-constrained rate control are not designed for this. We designed a rate control algorithm that uses a tighter target buffer level for most frames, and a less restrictive buffer fullness threshold at and after an LTR frame, and this approach outperformed previous methods.

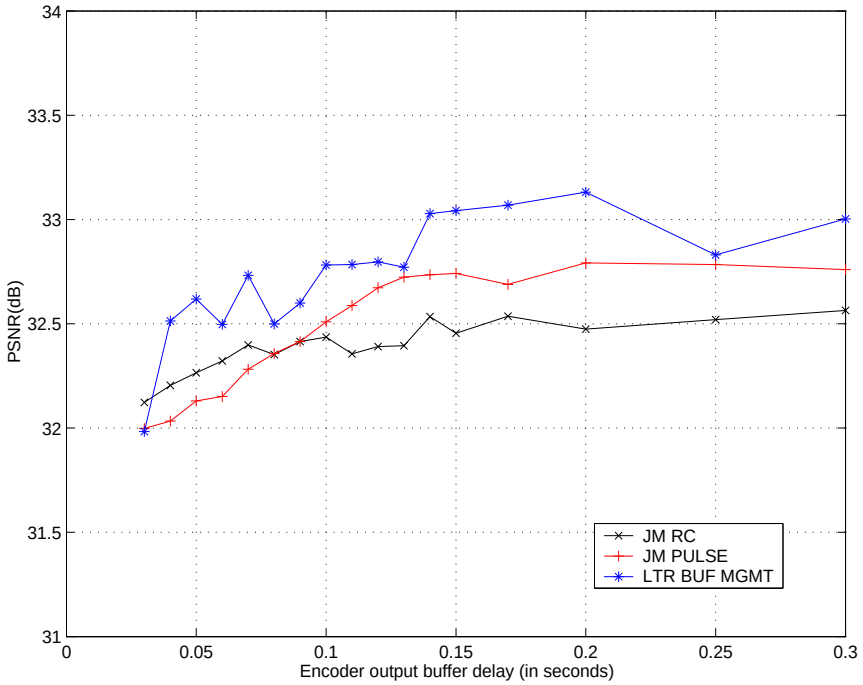
The proposed method outperforms JM H.264 rate control and a previously proposed rate control method for dual-frame video coding, even when these methods are modified to reduce the MB losses in a frame. The rate control algorithms were studied at low rates as the importance of high quality LTR frames fades away at high rates. We studied low motion video sequences in our work since dual-frame video coding does not provide significant gain for high motion video sequences. The buffer level and bit allocation for LTR frames can be optimized for a particular video sequence to further improve the performance.

2.5 Acknowledgement

Chapter 2 of this dissertation contains material that appears in M. Tiwari and P. Cosman, "Selection of long-term reference frames in dual frame video coding using simulated annealing", *IEEE Signal Processing Letters*, Vol. 15, pp 249-252, 2008 and M. Tiwari, T. Groves, and P. Cosman, and "Delay constrained rate control for low bitrate dual-frame video coding", *IEEE International Conference on Image Processing*, San Diego, California, pp 2484-2487, Oct 2008. I was the primary author and the co-authors Dr. Pamela Cosman and Dr. Theodore Groves directed and supervised the research which forms the basis for Chapter 2.



(a)



(b)

Figure 2.9: Variation of PSNR with the encoder output buffer delay (in seconds) for (a) News and (b) Container video sequence at 18 kbps.

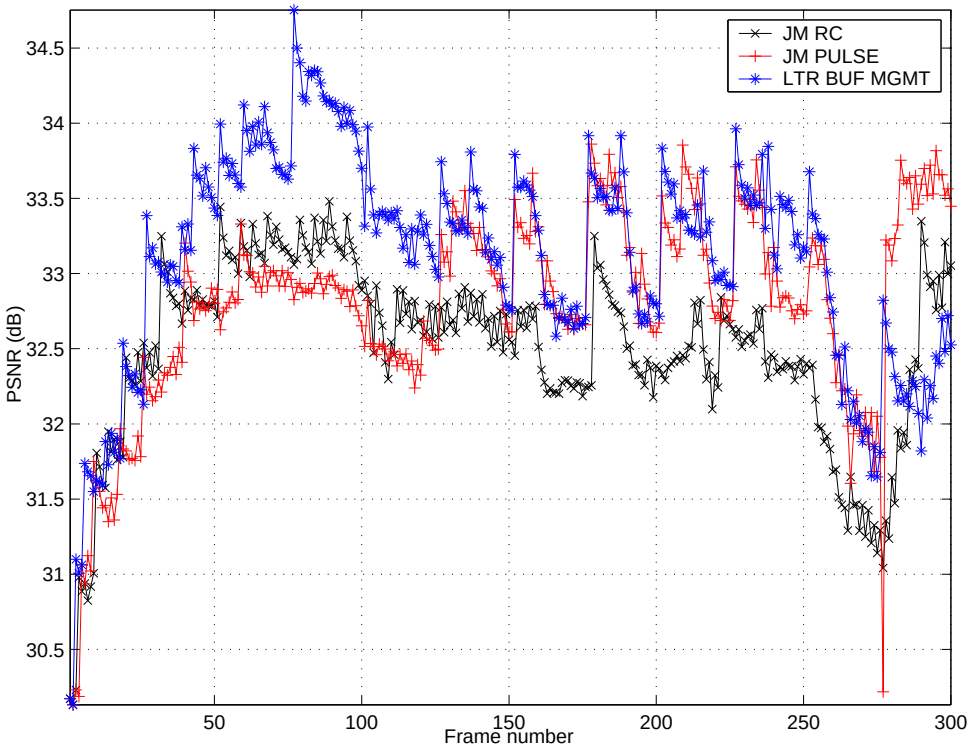


Figure 2.10: PSNR fluctuation with frame number for Container video at 18 kbps and 0.15 seconds encoder output buffer delay.

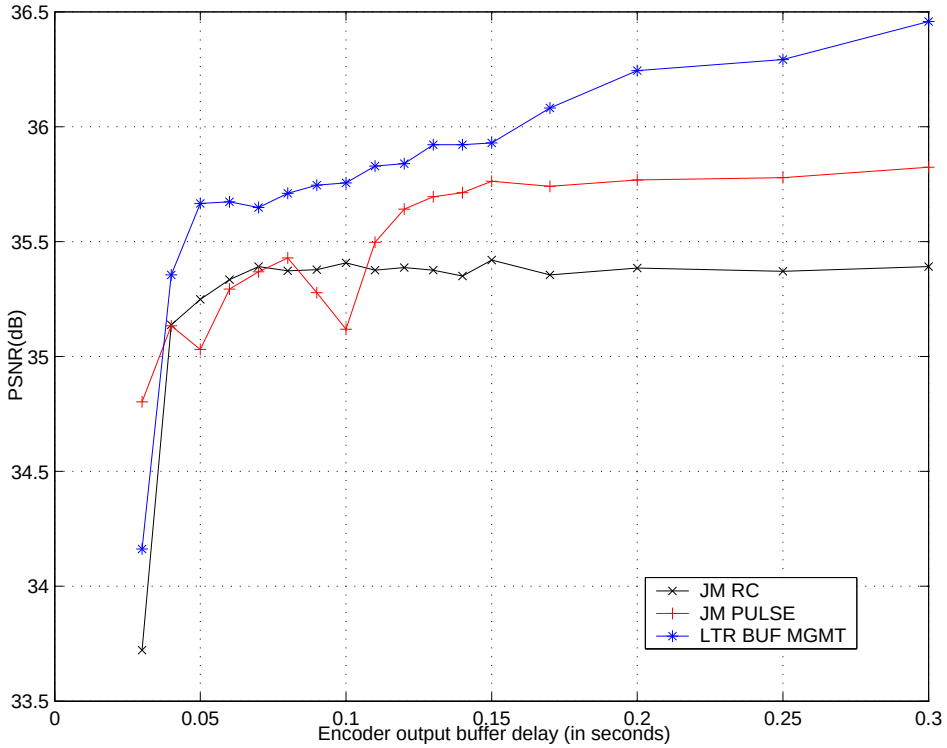


Figure 2.11: Variation of PSNR with the encoder output buffer delay (in seconds) for Container video sequence at 36 kbps.

Chapter 3

Delay constrained multiplexing of video streams using dual-frame video coding

3.1 Introduction

In this chapter, we present joint bit-rate allocation methods for multiple video streams using dual-frame video coding. The video Rate-Distortion (RD) information from each encoder is sent to a controller. Based on the RD information for the videos and the status of the output buffer, the controller calculates the optimal operating point for each video and sends the number of bits allocated to each encoder. Rate control generates variable size bitstreams for each frame in a video. Buffer constrained rate control for a video stream using dual-frame video coding was studied in Chapter 2. We extend this concept to multiple video streams where an encoder output buffer is shared among various users as in Fig. 1.5. We compare our multiplexing methods against rate allocation using (a) JM H.264 rate control, (b) dual-frame video coding, and (c) the superframe methods described in [25, 26]. While there are many coding variations that can yield improved compression performance, the use of high-quality LTR frames in dual-frame video coding not only improves performance for individual videos, but also has a particular advantage in

a buffer constrained multiplexing situation because the high-quality LTR frames which consume a large share of the delay buffer can be staggered among the different multiplexed streams.

In [44], three optimization objectives for transmitting multiple video streams over a shared channel were studied: maximizing overall PSNR, minimizing overall Mean Squared Error (MSE), and minimizing the maximum MSE. Using subjective tests, it was shown in [44] that minimizing the overall MSE corresponds best to subjective preferences. Using this result from [44], we chose the performance criterion of minimizing the overall MSE. The results here should not be compared directly with a method that assumes any other performance criterion.

For video multiplexing, our main contributions over the previous methods are as follows: (a) none of the previous methods in video multiplexing used dual-frame video coding. The previous methods of video multiplexing allocated more bits to a video with high motion by taking bits from low motion videos. Therefore, the low motion video quality suffers. Our dual-frame video coding technique gives huge MSE reduction for low motion videos compared to high motion videos. Therefore, we combine our dual-frame video coding method with the existing methods of allocating bits based on the motion. In our method, the LTR frames receives bits based on the activity measurement while the remaining frames are allocated bits based on the relative motion between the videos. The new multiplexing method further reduces the overall MSE. (b) the buffer-constrained rate control for dual-frame video coding was modified to accommodate the high quality LTR frames from all the video streams in order to avoid buffer overflow.

The chapter is organized as follows: Section 3.2 discusses multiplexing methods where the bits are allocated to multiple video streams depending on relative complexity of the videos. Section 3.3 describes several multiplexing methods using dual-frame video coding with high-quality LTR frames. Section 3.4 introduces the delay constrained rate control for dual-frame video coding. This section suggests a rate control modification to all the multiplexing methods discussed in previous sections. The results are discussed in Section 3.5. Section 3.6 concludes the chapter.

3.2 Multiplexing video streams with no encoder output delay

We use a simple form of dual-frame video coding for the multiplexing methods described in this section where two immediate past frames are used as reference frames to encode the current frame. We call these frames Short-Term Reference (STR) frames. Initially, we assume a negligible size of encoder output buffer.

Suppose we have N video streams and B kbps of total bit-rate available to transmit these video streams. The simplest bit-rate allocation is to divide bits equally among the video streams and among the frames. If a video stream is f frames per second, then each frame gets $\frac{B}{N \times f}$ bits. Each video stream is encoded with dual-frame video coding with two STR frames. We call this method **STR_EB** and it could be called a "fair" allocation. Given the number of bits to encode a frame of a video stream, the reference software model of H.264 [38] will search for and choose the best prediction mode to reduce the MSE.

Better multiplexing exploits the relative complexity (RD properties) of video streams. We take the curve-fitting model for the RD curve of a frame in video stream n to be

$$D_n(R_n) = a_n + \frac{b_n}{R_n} \tag{3.1}$$

where R_n is the number of bits and D_n is the distortion for a frame in video stream n, and a_n, b_n are the curve-fitting coefficients found using least squares. Other curve-fitting models are available in the literature [45]. We generate RD measurement points using 14 different QPs (ranging from 10 to 51) and that was found to be sufficient to calculate the curve-fitting coefficients for a broad range of bit-rates (ranging from 3 kbps to 1500 kbps for QCIF videos, depending on the video complexity). The complexity of generating RD curves can be reduced by using the method described in [46] and is not included in this dissertation.

Given the RD curve-fit for a frame in each video stream, the sum of MSE for all the video streams can be minimized using standard optimization techniques such as the Lagrangian multiplier [47]. Consider the i^{th} frame of each video stream.

The optimization problem can be formulated as

$$\min_{R_n} \sum_{n=1}^{N} D_n(R_n) \qquad subject\ to\ \sum_{n=1}^{N} R_n \leq \frac{B}{f} \tag{3.2}$$

Using Lagrange multipliers [47] for solving Eq. 3.2, the bit allocation for video j is

$$R_j = \frac{\sqrt{b_j}}{\sum_{n=1}^{N} \sqrt{b_n}} \times \frac{B}{f}, \qquad \forall j \in 1, 2, ..., N \tag{3.3}$$

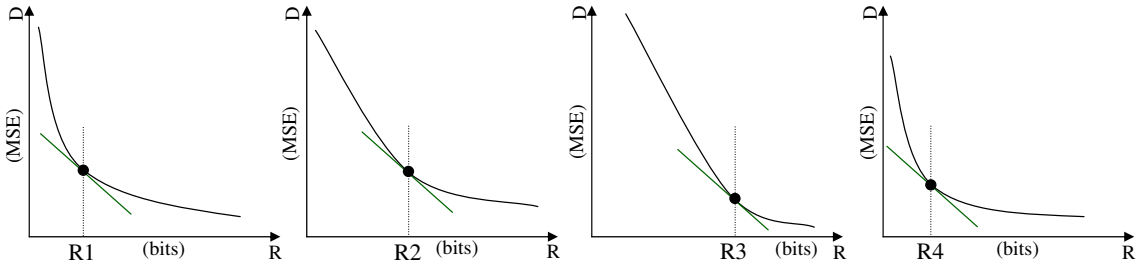


Figure 3.1: STR_ES method for four videos. R1, R2, R3, and R4 are the number of bits allocated to each video. At this allocation, the slopes of the RD curves are equal, and the sum of the four rates equals B/f.

The bit allocation achieved by Eq. 3.3 essentially finds a point in each RD curve where the slope is the same for all the curves. Known as the *equal slope* technique, this is shown in Fig. 3.1. After bit allocation, the videos are encoded with dual-frame coding with two STR frames. We denote this method of bit-rate allocation **STR_ES**. Although not identical, the basic approach is described in [44]. This minimizes, on a per frame basis, the sum of MSEs for all the video streams. This method gives extra bits to a video stream that is experiencing high motion by taking some bits from the low motion streams. STR_ES will result in STR_EB allocation if the RD curves for all the video streams are the same, in which case this method will allocate $\frac{B}{N.f}$ bits for each stream. The RD curve exists only at certain discrete points (because the QP takes on discrete values). Therefore, it may not be possible to achieve the exact bit-rate for a video specified by Eq. 3.3. The STR_ES method chooses the RD point that is closest to the bit allocation determined by Eq. 3.3. Ideally, we do not need an output buffer for these two multiplexing methods to store the encoded bitstream to be transmitted (because

the multiplexed bitstreams achieve the constant target output rate for each frame). In practice however, because of the discrete nature of the RD curve, we need a small output buffer to accommodate the difference between the target rate for a frame and the actual bit-rate achieved at some QP.

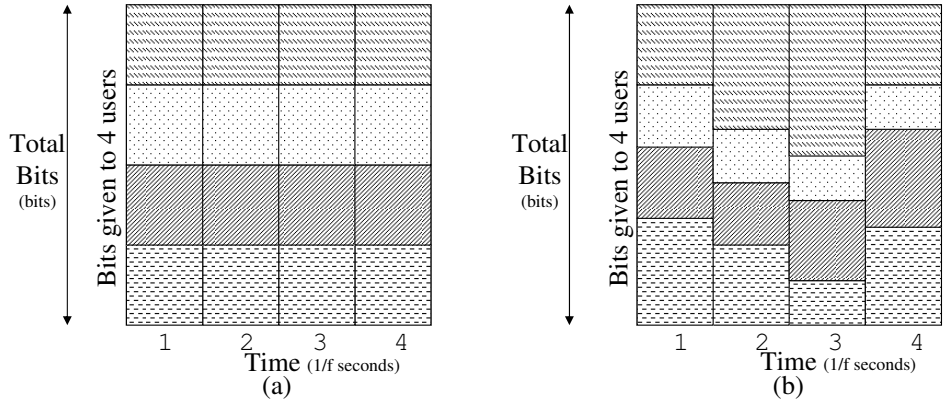
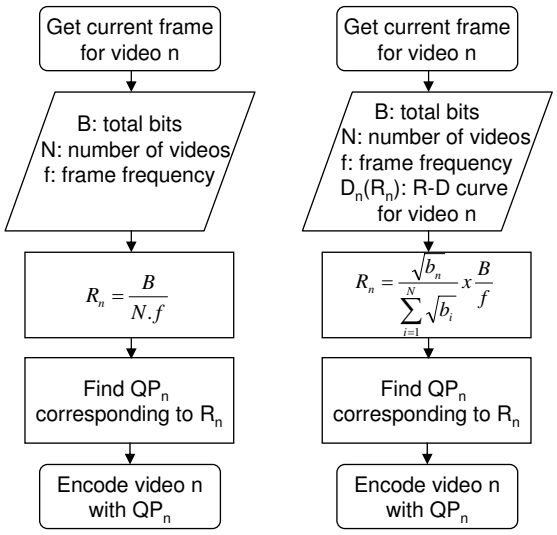


Figure 3.2: Multiplexing methods for four video streams (a) STR_EB, (b) STR_ES.

These two methods of multiplexing are shown in Fig. 3.2. Each shaded pattern in the figure represents one video stream, and each block represents the size in bits of a frame. On the y-axis is the number of bits given to each video and on the x-axis is the time (frame) slot, assuming a time slot of $\frac{1}{f}$ seconds at f frames per second. In Fig. 3.2(a), each video in each time slot gets the same number of bits, so the bits per user per time slot can be depicted as identical boxes. Fig. 3.2(b) shows the STR_ES method where the videos with higher activity take bits from the ones with lower activity. The videos do not take bits from any other time slot, so the depiction of total bits per time slot still shows vertical boundaries. Fig. 3.3 shows the flow chart for the (a) STR_EB method and (b) STR_ES method.

We compared our methods for multiplexing video streams with the superframe method given in [25]. We applied the superframe method with the total number of bits at each frame as discussed in STR_EB. This method is denoted by **SF_EB**. For a fair comparison, we use two STR frames for the SF_EB method so it has the same advantage of dual-frame video coding. The picture in Fig. 3.2(b) applies to this method, except that the block boundaries are now defined by assigning the same QP for all the videos instead of the same slope.



(a) STR_EB (b) STR_ES

Figure 3.3: Flow chart for the (a) STR_EB, (b) STR_ES multiplexing methods.

3.3 Multiplexing video streams using adaptive dual-frame video coding

In the previous section, we used dual-frame coding where two frames immediately preceding the current frame to be encoded were used as references. It was shown in [8, 12] that temporally separated reference frames perform better than consecutive reference frames. We now use the dual-frame video coding with high quality LTR frames that was discussed in Chapter 2.

As discussed in Chapter 2, let m_n be the average fraction of active MBs in the 10 frames prior to an LTR frame for n^{th} video. Based on m_n, the bit allocation for the LTR frame (L_n) of n^{th} video stream is given by

$$L_n = \begin{cases} 2 \times reg\_bits, & \text{if } m > 0.5 \\ 10 \times reg\_bits, & \text{if } m < 0.1 \\ (12 - 20 \times m) \times reg\_bits, & otherwise, \end{cases} \tag{3.4}$$

where reg_bits is the average number of bits assigned to a regular frame.

3.3.1 Multiplexing video streams using dual-frame video coding with high-quality LTR frames

Both the multiplexing methods in the previous section use dual-frame video coding with two STR frames for motion compensation. We can further reduce the sum of MSE by exploiting the high-quality LTR frame in dual-frame video coding. Let L_n be the number of bits assigned to an LTR frame for the n^{th} video stream. Note that L_n varies with the motion of a video stream. The extra bits given to the LTR frame are taken equally from the regular frames between two high quality LTR frames. Let K be the distance between two LTR frames. Using STR_EB, each video stream should get $\frac{K \times B}{N \times f}$ bits for K frames. In the high-quality LTR extension to STR_EB, out of this pool of bits, L_n bits are assigned to an LTR frame. The remaining bits ($\frac{K \times B}{N \times f}$ - L_n) are equally divided among each of the remaining $K-1$ frames in that group of K frames in the stream. If r_n denotes the number of bits allocated to each of these $K-1$ frames, then

$$r_n = \frac{\frac{K \times B}{N \times f} - L_n}{K-1} \tag{3.5}$$

This may also be deemed a "fair" allocation since each video stream receives an equal number of bits for the entire video. Although the number of bits L_n given to the LTR for stream n may be higher than that for some other stream, the number of bits r_n given to the other $K-1$ frames in that group of K frames for stream n will be correspondingly lower, so each video stream is allocated an equal number of bits over all frames. This method is called **eLTR_EB**.

We can also incorporate dual-frame video coding with high-quality LTR frames in the STR_ES method. Again, the bit allocation for LTR frames is as described in eLTR_EB. Using Eq. 3.3, for any frame where no stream has an LTR, the bits allocated to video j are

$$R_j = \frac{\sqrt{b_j}}{\sum_{n=1}^{N} \sqrt{b_n}} \times \sum_{n=1}^{N} r_n, \qquad \forall j \in 1, 2, ..., N \tag{3.6}$$

where r_n is defined by Eq. 3.5. If, at any time instant, a video stream k has an LTR frame, then that video stream receives L_k bits and the remaining video streams

will receive an allocation similar to STR_ES, i.e.

$$R_j = \frac{\sqrt{b_j}}{\sum_{n=1,n\neq k}^{N}\sqrt{b_n}} \times \sum_{n=1,n\neq k}^{N} r_n, \quad \forall j \in 1,2,...,N, j \neq k \tag{3.7}$$

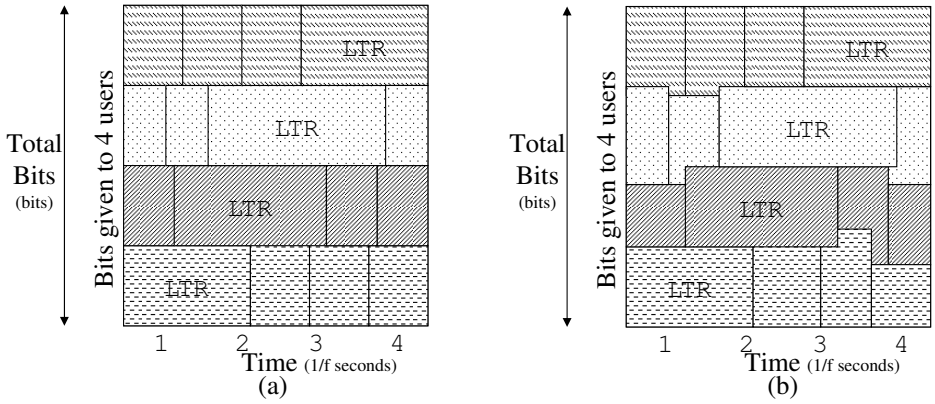
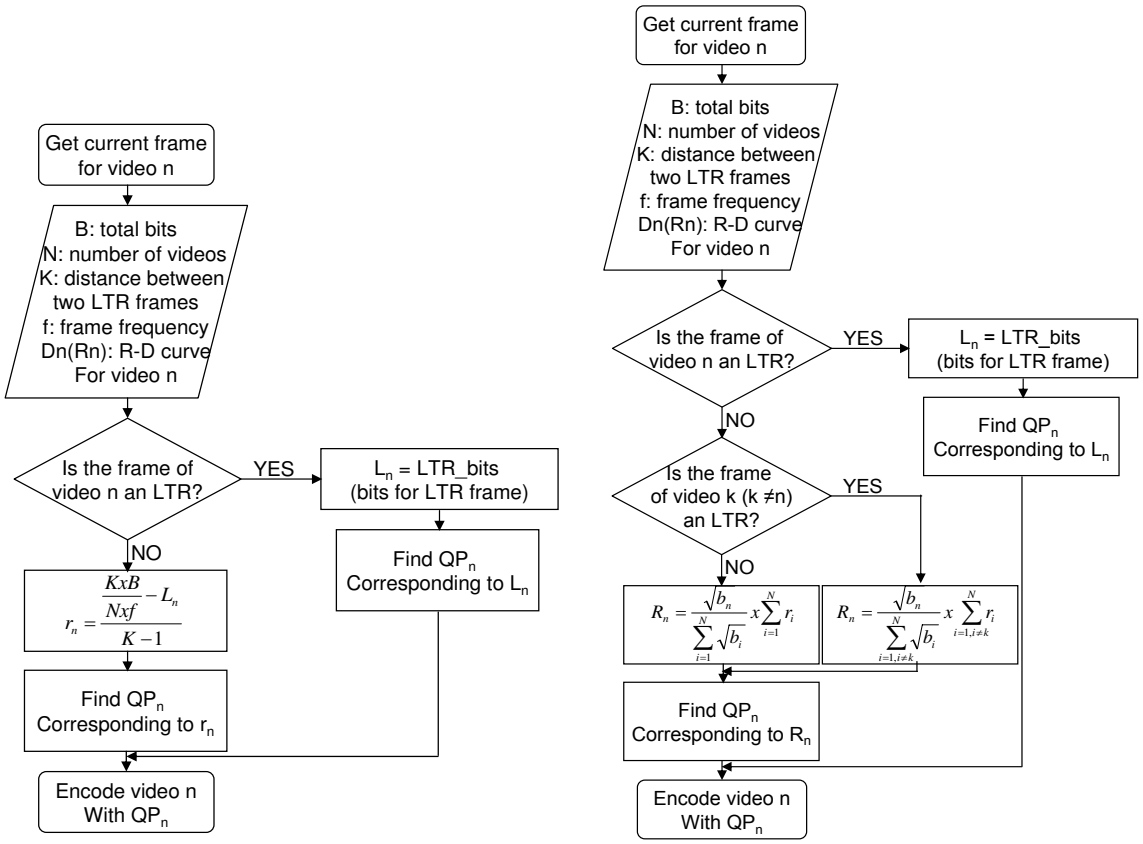


Figure 3.4: Multiplexing methods for four video streams (a) eLTR_EB, and (b) eLTR_ES.

This method uses dual-frame video coding with high-quality LTR frames, and equal slope allocation for regular frames, and is denoted by **eLTR_ES**. Note that both eLTR_EB and eLTR_ES require an output buffer, due to high-quality LTR frames, to store the encoded bitstream for transmission over a constant bit-rate channel. Since LTR frames are assigned more bits than regular frames, the chances of buffer overflow are higher for an LTR frame. Irrespective of the number of bits for an LTR frame determined by the motion activity, the size of an LTR frame is upper bounded by the available space in the output buffer ($avail\_buf$), i.e.,

$$LTR\_bits = min(LTR\_bits,\ avail\_buf) \tag{3.8}$$

Fig. 3.4(a) depicts the eLTR_EB method. Extra bits for the LTR frames of each video are taken from regular (non-LTR) frames of the same video. Since the extra bits for the LTR frames of one video are not taken from any other video, the depiction of bits per user still has horizontal boundaries. In Fig. 3.4(b), bits are taken across users and across time slots to accommodate both higher activity videos and LTR frames. Fig. 3.5 shows the flow chart for the (a) eLTR_EB, and (b) eLTR_ES multiplexing method. We expect STR_ES will perform better than



(a) eLTR_EB (b) eLTR_ES

Figure 3.5: Flow chart for the (a) eLTR_EB, (b) eLTR_ES multiplexing methods

STR_EB because, in STR_ES, bits are allocated to different video streams based on their complexity. We expect that eLTR_EB will perform better than STR_EB, and eLTR_ES will perform better than STR_ES, because of the advantages of dual-frame video coding with high-quality LTR frames.

3.3.2 High-quality LTR frame selection

A method was proposed in Chapter 2 using simulated annealing to select LTR frames that are more effective for subsequent frames. Video quality improvement was achieved at the cost of higher computational complexity. We propose a simpler method to select the location of LTR frames using the motion activity of the video sequence. Our modifications are done only at the encoder, and produce a standard compatible bitstream.

To decide if the current frame should be designated an LTR frame, we calculate the activity of the current frame to be encoded with respect to the current LTR frame. If the number of active MBs is more than some adaptive threshold (denoted by *active_thr*), then it is time for the next LTR frame. Depending on the frequency of LTR frames, we update *active_thr*. The *active_thr* is incremented by *thr_incr* amount, given by

$$\begin{aligned} thr\_incr = \frac{1}{4} \times (avg\_ltr\_dist - curr\_ltr\_dist) \\ \times \frac{(max\_mot\_thr - min\_mot\_thr)}{(max\_ltr\_dist - min\_ltr\_dist)} \end{aligned} \tag{3.9}$$

where *min_ltr_dist* is set to 10 frames which is our chosen threshold for the minimum number of frames for which an LTR frame is useful (without a scene change). A smaller number of frames can also be considered for *min_ltr_dist* for high motion videos but this may result in too frequent LTRs to assign higher quality. *max_ltr_dist* is set to 40 frames which is our chosen threshold for the maximum number of frames for which an LTR frame is useful. *avg_ltr_dist* is set to 25 frames (average of *min_ltr_dist* and *max_ltr_dist*) between two LTRs, similar to the one used in [48]. These values are determined experimentally using a large set of videos. The *curr_ltr_dist* is the distance between the current frame and the last assigned LTR frame. We increase *active_thr* to space out the future LTR frames, so as

not to use up the bit budget, if the LTR frames have been assigned frequently. If the LTR frames are assigned far apart then *thr_incr* is negative which decreases *active_thr* to reduce the LTR distance.

We do not take future frames into account for choosing LTR locations. Therefore, this method will fail to determine good LTR locations in some cases such as a scene change. In such a case, ideally the new LTR frame should be assigned at the scene change. If the scene change location is less than the minimum LTR distance, then the algorithm will assign the next LTR soon after the minimum LTR distance, and the LTR frame will again be useful for subsequent frames. If we use future frames then the LTR frame selection performance will increase but at the expense of large delay.

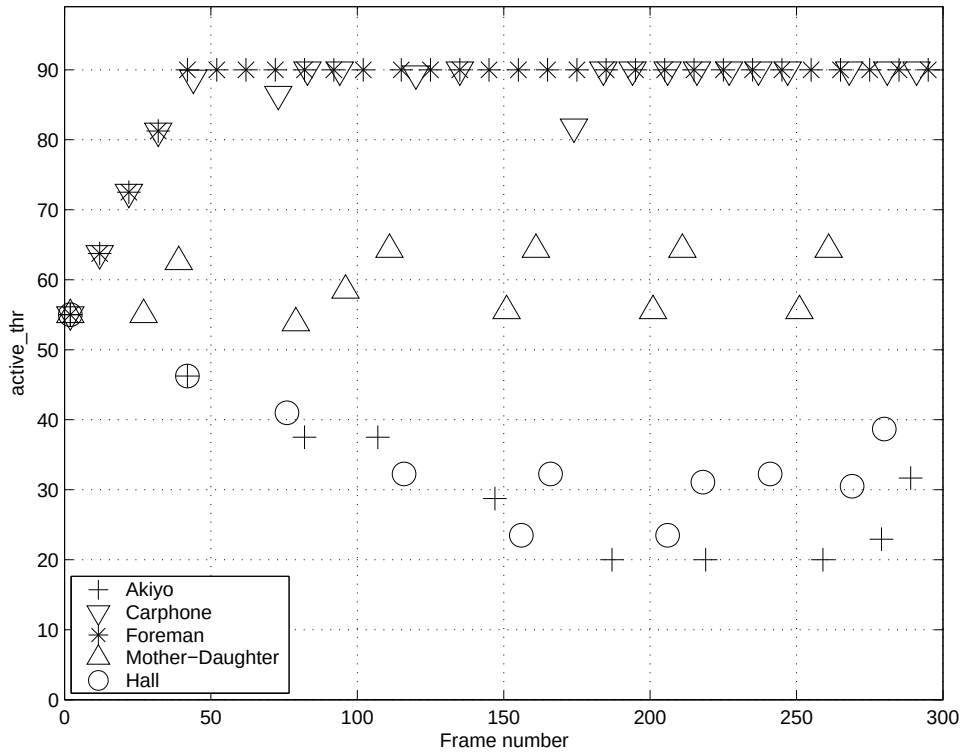


Figure 3.6: LTR frame locations and *active_thr* for various video streams. Frame number 2 is the first LTR frame with *active_thr* of 55.

Fig. 3.6 shows the result of our LTR frame selection method along with the change in *active_thr* with the frame number. The simulation was performed for 300 frames with the second frame as the starting LTR frame and an initial *active_thr*

of 55, which is the average number of active MBs per frame when averaged over a large set of QCIF videos containing both high and low motion videos. The choice of initial value of *active_thr* minimally affects the overall performance since its value is automatically adjusted depending on the motion of the video stream. The x-axis represents the frame number and the y-axis represents the *active_thr*. Each set of symbols represents a different video stream and the marked symbols show the locations of LTR frames and its corresponding *active_thr*. For low motion streams such as Akiyo, we see that *active_thr* tends to decrease from the beginning and the LTR frames are usually separated by a large distance. In contrast, for videos like Foreman which are relatively higher motion streams, the LTR frames occur frequently and *active_thr* quickly saturates to the maximum limit. This LTR frame selection process improves video quality compared to evenly spaced LTR frames for nearly all of the video streams.

3.3.3 Multiplexing video streams using dual-frame video coding with unevenly spaced high-quality LTR frames

The quality of multiple videos using eLTR_EB and eLTR_ES can further be improved by choosing the high-quality LTR frames based on their motion activity. The first high-quality LTR frame in each video stream is assigned sequentially. For the first video, the high quality LTR frame is the first P-frame. The high-quality LTR frame for the next video is assigned one frame after the LTR of the previous video. After the first LTR frame, the location of the next LTR frame for any video stream will be calculated using the activity detection algorithm as discussed earlier. Since the location of the next LTR frame is unknown at the time of encoding the current LTR, we do not know the number of regular frames between the two LTR frames that will be used to extract the extra bits for the LTR frames. Therefore, we assume that the next LTR frame will be assigned after the same number of frames as the distance between the previous two LTR frames. We then extract the extra bits evenly from these frames for the LTR frame. Depending on the actual location of the next LTR frame, we calculate the excess (or shortage) of bits that were previously allocated. These bits are distributed among the regular frames

between the next two LTR frames. This multiplexing method is called **LTR_EB**.

Similarly, we can also improve eLTR_ES by incorporating motion based high-quality LTR frame selection. Again, the bit allocation for LTR frames is as described in eLTR_EB. This method uses improved dual-frame video coding with high-quality LTR frames and equal slope allocation for regular frames and is denoted by **LTR_ES**. The depiction of LTR_EB is the same as eLTR_EB which is shown in Fig. 3.4(a) and the depiction of LTR_ES is the same as eLTR_ES which is shown in Fig. 3.4(b). We expect LTR_ES will perform better than LTR_EB because, in LTR_ES, bits are allocated to different video streams based on their complexity. We expect that LTR_EB will perform better than eLTR_EB, and LTR_ES will perform better than eLTR_ES, because of the advantages of adaptive location of high quality LTR frames.

3.4 Multiplexing video streams with delay constrained rate control

The methods described earlier did not use rate control, and the MSE improvement over STR_EB is achieved only by comparing the relative complexity across video streams at each frame and by using dual-frame video coding with high-quality LTR frames. The performance of all these methods can further be improved by using rate control which also exploits the relative complexity across the frames in each video. The quality improvement due to rate control comes with a penalty of increased delay at the encoder. With a fixed delay constraint, the encoder may drop a frame partially and this may cause severe error propagation. In this section, we propose multiplexing methods using rate control with a fixed delay buffer. We first consider multiplexing methods using rate control for dual-frame video coding with separate delay buffers. We then extend the multiplexing methods to consider a joint delay buffer.

3.4.1 Rate control for multiplexing using dual-frame video coding with separate delay buffers

We again start with STR_EB as the most simple method of bit allocation among multiple video streams. This method neither shares bits among the videos nor shares bits across frames of a single video stream. The basic idea of rate control is to share bits across frames of a video stream in such a way that high motion or complex frames get more bits. By doing so, we increase the overall quality of a video stream. For a fixed delay buffer, we apply the JM H.264 rate control where the encoder uses the RD optimization to efficiently encode each video stream separately. We denote this method **STR_RC**. To avoid buffer overflow, we set a Buffer Fullness Threshold (BFT), as discussed in Chapter 2. We start increasing the QP by 2 if the buffer occupancy exceeds the BFT which is initially set to 50%. Otherwise, we let the H.264 RD optimization determine the QP. If the encoded frame size exceeds the available buffer size, then we drop MBs using the skip mode. The skipped MBs are reconstructed using motion compensated prediction from the STR frame where neighboring motion vectors are used to estimate the motion vector of the lost MB.

With the addition of high-quality LTR frames in dual-frame video coding where many bits are assigned to the LTR frames, the chances of a portion of an LTR frame getting dropped is higher than for a regular frame. We modify the BFT as shown in Fig. 2.8 of Chapter 2, depending on whether a frame is an LTR or not. We denote this rate control approach for dual-frame video coding with evenly spaced high-quality LTR frames as **eLTR_RC**.

The performance of eLTR_RC can further be improved by selecting the locations of the high-quality LTR frames, in addition to their quality levels, using motion activity as given in section 3.3.2. We denote this method **LTR_RC**. The buffer control operates as for eLTR_RC. The quality improvement in LTR_RC over eLTR_RC is attributed only to the adaptive location of high-quality LTR frames.

3.4.2 Rate control for multiplexing using dual-frame video coding with a joint delay buffer

In multiplexing video streams, we replace the separate encoder output buffer for each video stream by one common encoder output buffer as shown in Fig. 1.5. Therefore, the above multiplexing methods need to be modified for one common delay buffer. The rate control extension of STR_ES is denoted by **STR_ES_RC**. Here, the recommended rate control [41] for the H.264 reference software was used to predict the combined frame level complexity of all the videos in order to determine the target bit-rate. Given the combined target bit-rate, the equal slope technique was then applied to allocate bits among videos. Both the rate control and equal slope technique improve the quality when compared to the STR_EB method. Similarly, **SF_RC** is the rate control extension of SF_EB. The target bit-rate for the superframe was assigned by calculating the relative superframe complexity and then superframes are encoded with H.264 rate control (bit constraint for individual superframes is waived). Quality improvement over SF_EB is achieved by assigning the bits to a superframe based on their relative complexity.

While there are many coding variations that can improve compression performance, the use of high-quality LTR frames has an intuitive appeal in a delay buffer constrained multiplexing scenario. Because the LTR frames which demand extra buffer space can be staggered among the different multiplexed video streams, all videos get some benefit from high-quality LTR coding, without overflowing the buffer. With high-quality evenly spaced LTR frames in dual-frame video coding, (eLTR_RC), each encoder independently allocates the bits to its frames based on the relative complexity and its delay buffer. In **eLTR_ES_RC**, the multiplexing method not only assigns the bits to a frame based on the relative frame complexity in a video stream, but also considers the complexity among the video streams at the frame level with a joint delay buffer, thus improving the overall quality. **LTR_ES_RC** is a modified multiplexing method of eLTR_ES_RC where the LTR frames are selected using motion activity. We still use the same method to predict the MAD to estimate the complexity of each frame. The QP for a frame in each video is decided using equal slope based on the MAD prediction and target

bit-rate.

With multiple video streams having LTR frames at different locations, we need to avoid a situation where the combined buffer fullness crosses the BFT. Bits for LTR frames are assigned based on the method described in section 2.3.2. Following each LTR, we decrease the buffer fullness in bf_slope frames. The QP is increased for a frame in any video if it causes the buffer fullness to go above BFT. With a large common buffer, the LTR frames are not limited by the small individual buffer size and it is possible to take more advantage of LTR frames in such a case. The QP for each video is adjusted in such a way that the total bits for all the videos at any time should not overflow the output buffer. As in the previous case, MBs are skipped to avoid buffer overflow. The importance of a combined output buffer for various multiplexing methods will become more clear in the simulation results.

For multiplexing video streams using dual-frame video coding, we assign the high-quality LTR frames using the method described above. For a delay buffer, it is sometimes difficult to accommodate the high-quality LTR frames for different videos close to each other. Suppose there are two video streams to be multiplexed together and both the streams are about to assign an LTR frame close to each other. Suppose we are currently encoding frame $k-1$. We calculate the motion between the current LTR frame and frame $k-1$ for both the videos. We also know the motion between the current LTR frame and frame $k-2$ for both the videos. By extrapolating these two motions, suppose we predict that both the videos will exceed their *active_thr* for frame k. Then the LTR frame of a video that is moving faster towards its *active_thr* compared to the other video is moved ahead by one frame to frame $k-1$ (as long as frame $k-1$ is not within the minimum LTR distance). The LTR frame of the other video which is moving slower towards its *active_thr* is delayed by one frame to frame $k+1$ (as long as frame $k+1$ is not beyond the maximum LTR distance). By doing so, we create some space between the LTR frames and allocate the desired number of bits to each LTR frame, yet avoid overflowing the buffer.

In summary, we have the following multiplexing methods using delay constrained

rate control:

1. **STR_RC**: This is the rate control extension of STR_EB. A target bit-rate was assigned and the encoder was allowed to use RD optimization to efficiently encode each video stream separately for a given size of its separate output buffer.

2. **STR_ES_RC**: This is the rate control extension of STR_ES. Here the recommended rate control for the H.264 reference software was used to predict the combined frame level complexity of all the videos and then the equal slope technique was applied to allocate bits among videos.

3. **eLTR_RC**: This is the rate control extension of eLTR_EB. Each video stream is encoded separately with dual-frame coding with evenly spaced high-quality LTR frames. The target bit-rate for a frame is calculated using the frame complexity. Bits are taken from the regular frames and given to the LTR frames. Quality improvement over STR_EB is achieved using rate control and dual-frame coding with evenly spaced high-quality LTR frames.

4. **eLTR_ES_RC**: This is the rate control extension of eLTR_ES. First the combined frame level complexity is estimated as described for eLTR_RC and then the equal slope technique is applied. This method combines rate control with equal slope and dual-frame coding with evenly spaced high-quality LTR frames to further improve the overall video quality over STR_EB.

5. **LTR_RC**: This method is the rate control extension of LTR_EB. It is similar to eLTR_RC with the exception that the locations of high-quality LTR frames are chosen using the motion activity of a video.

6. **LTR_ES_RC**: This is the rate control extension of LTR_ES. It is similar to eLTR_ES_RC with the exception that the locations of high-quality LTR frames are chosen using the motion activity of a video.

7. **SF_RC**: This is the rate control extension of SF_EB. The target bit-rate for the superframe was assigned by calculating the relative superframe complexity and then superframes are encoded with H.264 rate control. Quality

improvement over SF_EB is achieved by assigning the bits to a superframe based on their relative complexity.

For LTR_ES_RC, the number of bits for a frame in a video is estimated by predicting the complexity of that frame. With the total bits at each frame level, we apply the equal slope technique to all the frames except the LTR frames. Thus, we combine the LTR_ES with rate control to achieve the bit allocation for different video streams. With rate control, the above seven methods should perform better than their respective methods without rate control. We expect that LTR_ES_RC will perform better than the other multiplexing methods discussed in this chapter because it has the advantage of unevenly spaced high-quality LTR frames, rate control, and equal slope bit allocation among videos.

3.5 Results

The video multiplexing methods described in the previous sections were simulated using the baseline profile of H.264/AVC [37]. The H.264/AVC reference software JM 10.1 [38] was modified for our simulation purpose. All the video streams used for the simulations are QCIF (176 $\times$ 144 pixels) at 30 frames per second and of length 300 frames. The first frame is an I-frame and the remaining frames are P-frames. The multiplexing methods can be used for any Group Of Picture (GOP) structure. We can either use the equal slope technique for I-frames if the GOP is of the same size for all the videos, or the I-frames can be encoded traditionally where its QP is determined by the QPs from the previous GOP. We can still use our multiplexing methods if there are B-frames with the exception that the B-frames that are not used for referencing can not be used as LTR frames. We considered a lossless channel at various bit-rates for the simulation.

Table 3.1 shows the results of multiplexing two video streams at 60 kbps. For any video and any multiplexing method in Table 3.1, the MSE is averaged within and across all the frames. The average MSE for any multiplexing method shows the MSE average over both the video streams. The PSNR for any video stream is calculated from the overall MSE of that video stream, and the average

Table 3.1: MSE and PSNR for multiplexing two video streams.

	STR _EB	STR _ES	SF _EB	eLTR _EB	eLTR _ES	LTR _EB	LTR _ES	STR _RC	STR _ES_RC	SF _RC	eLTR _RC	eLTR _ES_RC	LTR _RC	LTR _ES_RC
Foreman (MSE)	114.5	71.2	65.2	107.2	78.8	101.0	75.5	108.6	67.0	61.7	110.9	74.0	104.1	68.7
Foreman (PSNR)	27.54	29.60	29.99	27.83	29.17	28.09	29.35	27.77	29.87	30.23	27.68	29.44	27.95	29.76
Akiyo (MSE)	20.8	29.2	37.6	10.6	14.6	10.7	14.0	9.2	26.6	34.4	7.7	14.7	8.1	14.9
Akiyo (PSNR)	34.95	33.47	32.38	37.89	36.50	37.85	36.67	38.50	33.89	32.76	39.25	36.47	39.06	36.40
Average MSE	67.7	50.2	51.4	58.9	46.7	55.8	44.7	58.9	46.8	48.1	59.3	44.3	56.1	41.8
PSNR (Avg MSE)	29.83	31.12	31.02	30.43	31.44	30.66	31.62	30.43	31.43	31.31	30.40	31.66	30.64	31.92

PSNR is calculated from the MSE averaged over all frames of both video streams. The following inferences can be drawn from Table 3.1:

- For higher motion videos such as Foreman, STR_ES reduces its MSE by assigning more bits. Lower motion videos such as Akiyo receive fewer bits and experience an increase in MSE. The MSE reduction for the high motion video is much larger than the MSE increase for the low motion video when compared to the STR_EB case. Therefore, there is a large reduction in overall MSE.

- eLTR_EB uses evenly spaced high-quality LTR frames. We see MSE reduction in both the videos compared to STR_EB. This shows the advantage of using LTR frames and assigning appropriate high quality to them. In this case, both the videos receive equal numbers of bits. The lower motion video, Akiyo, gains more from the high-quality LTR frames than the higher motion video.

- When the equal slope technique is applied along with evenly spaced LTR frames in eLTR_ES, we see that Foreman further reduces its MSE compared to eLTR_EB while there is some increase in MSE for Akiyo. Again, because of equal slope allocation, Foreman receives more bits than Akiyo.

- Although both STR_ES and SF_EB assign bits to each video stream based on the complexity, the performance of STR_ES is better than that of SF_EB due to the fact that STR_ES gives the optimal bit allocation based on the RD curve of each video stream (Eq. 3.2 and Eq. 3.3) at the frame level.

- Comparing evenly spaced high-quality LTR frames in dual-frame video coding with the high-quality LTR location found using the activity detection algorithm, we see that our method of finding LTR location, in general, performs better than taking evenly spaced LTR frames. Therefore, LTR_ES and LTR_EB perform better than eLTR_ES and eLTR_EB respectively.

Of all the multiplexing methods without rate control, we see that LTR_ES performs the best. When we compare the overall MSE, we find that LTR_ES

performs better than STR_EB by 1.79 dB, STR_ES by 0.50 dB, SF_EB by 0.60 dB, eLTR_EB by 1.19 dB, eLTR_ES by 0.18 dB, and LTR_EB by 0.96 dB.

During simulation trials, it was observed that if we allocate many bits to LTR frames, then the performance of LTR_EB tends to be very close to LTR_ES. When more bits are given to LTR frames, then fewer bits are left for other frames. Therefore, when we equalize the slope for these other frames, there are not many bits to adjust and we get a very small advantage from doing equal slope. If we give fewer bits to LTR frames, then the effect of LTR frames is small. In that case, the performance of LTR_EB is close to that of STR_EB. Therefore, it is necessary to moderate the amount of extra quality given to LTR frames to achieve better performance.

Table 3.1 also shows the results for the multiplexing methods using rate control. In general, the following trends can be observed from the table:

- Using rate control, STR_RC performs better than STR_EB by 0.60 dB because, with an output buffer, there is more freedom in assigning the bits to various frames according to their relative complexity.
- The multiplexing methods, STR_ES_RC, LTR_RC, LTR_ES_RC, and SF_RC using rate control perform better than their counterparts without rate control: STR_ES, LTR_EB, LTR_ES and SF_EB respectively.
- Using the equal slope technique, STR_ES_RC marginally outperforms SF_RC.
- With the help of dual-frame video coding, LTR_RC performs better than STR_RC, and LTR_ES_RC outperforms STR_ES_RC.

Overall, LTR_ES_RC outperforms STR_RC by 1.49 dB, LTR_RC by 1.28 dB, SF_RC by 0.61 dB, and STR_ES_RC by 0.49 dB. When comparing individual video streams we find that, for high motion video streams, equal slope allocation reduces the MSE by a huge margin, but at the expense of an increase in the MSE for low motion video streams. On the other hand, dual-frame video coding decreases the MSE by a small amount for high motion videos but it is more successful in reducing the MSE for low motion videos. Rate control also reduces the MSE of

each video stream separately. The combination of dual-frame video coding, equal slope allocation, and rate control outperforms all other methods of multiplexing multiple video streams.

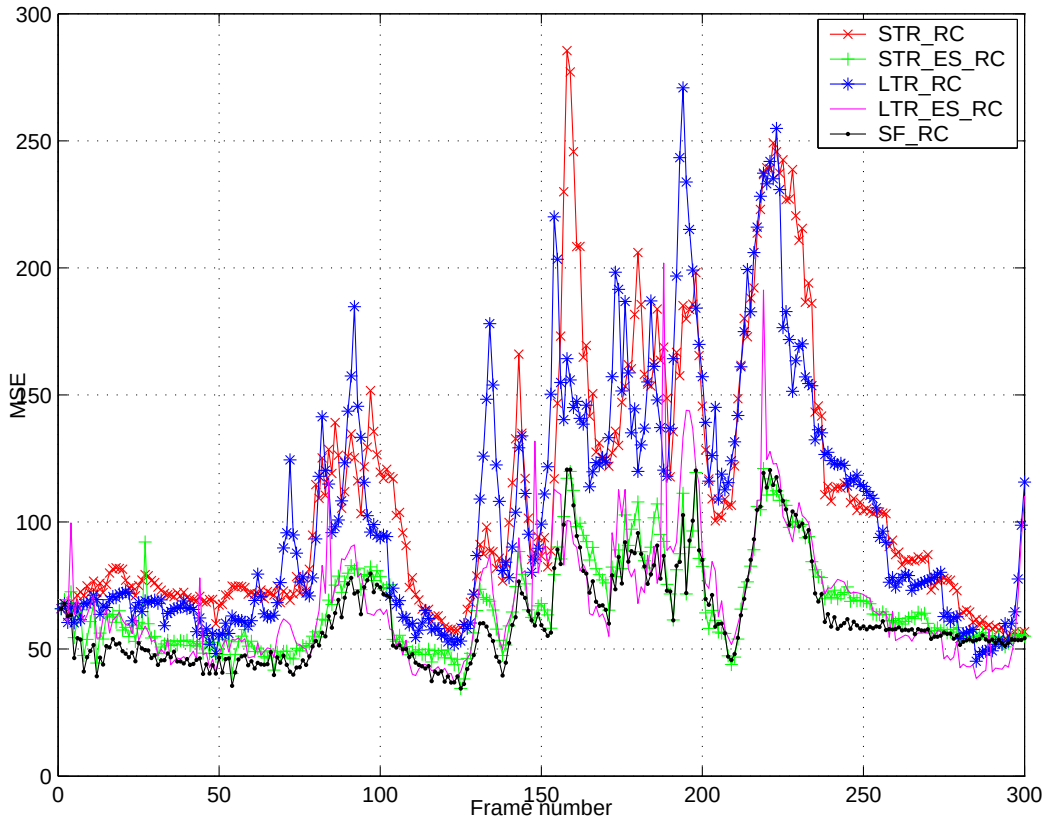


Figure 3.7: MSE variation with frame number for Foreman in multiplexed video streams.

Figs. 3.7 and 3.8 show the MSE versus frame number for both the multiplexed video streams. The five curves in each figure represent different methods of multiplexing video streams with rate control. For clarity, we only plot five methods for multiplexing that involve rate control and LTR frame selection. The curve at the bottom represents the lowest (best) MSE and the curve at the top represents the highest (worst) MSE. As can be seen, SF_RC squeezes the Akiyo video and gives many bits to the Foreman video, producing a large MSE increase when compared with STR_RC. The STR_ES_RC performs close to SF_RC in Foreman but it produces much better results for Akiyo. The MSE variation of LTR_RC

is similar to STR_RC for Foreman, meaning that the dual-frame video coding does not improve the result by a large amount individually. Even though the difference between STR_RC and LTR_RC for Akiyo is not clear from the figure, LTR_RC produces a large PSNR increase compared to STR_RC. The performance of LTR_ES_RC is close to SF_RC for Foreman but it does much better than SF_RC in Akiyo. The effect of LTR frames is not clearly visible in the figure for Foreman due to frequent small increments in the LTR frame quality but the MSE drops in Akiyo show the presence of high-quality LTR frames. Since the entire video is of high quality, the quality fluctuations are not perceptually noticeable when viewing the video, but overall the high quality LTR frames increase the quality of the entire video stream.

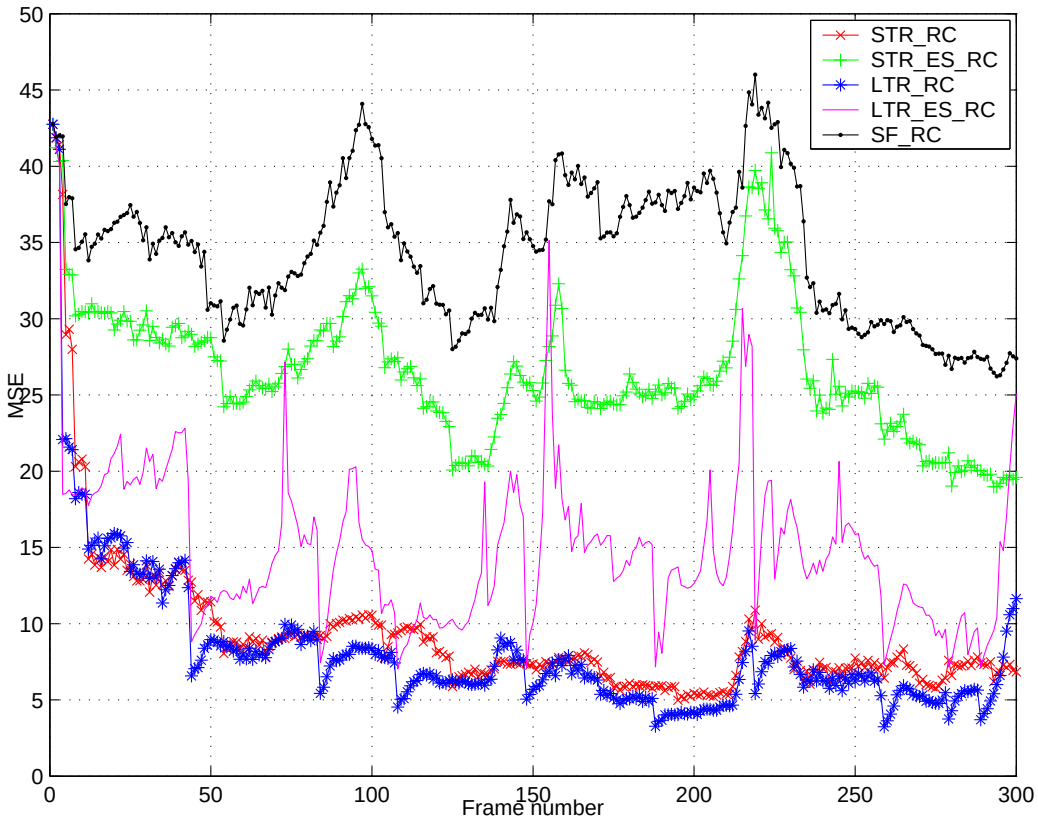


Figure 3.8: MSE variation with frame number for Akiyo in multiplexed video streams.

A similar result is shown in Table 3.2 where four video streams are mul-

Table 3.2: MSE and PSNR for multiplexing four video streams.

	STR _EB	STR _ES	SF _EB	eLTR _EB	eLTR _ES	LTR _EB	LTR _ES	STR _RC	STR _ES_RC	SF _RC	eLTR _RC	eLTR _ES_RC	LTR _RC	LTR _ES_RC
Carphone (MSE)	85.9	61.5	56.3	81.0	63.4	78.8	61.7	81.1	58.0	52.8	81.7	61.7	81.3	60.6
Carphone (PSNR)	28.79	30.24	30.63	29.05	30.11	29.16	30.23	29.04	30.49	30.90	29.01	30.23	29.03	30.31
Grandma (MSE)	26.3	38.8	47.7	16.8	21.2	17.5	21.3	16.9	35.6	44.2	14.5	20.9	14.6	21.0
Grandma (PSNR)	33.93	32.24	31.35	35.87	34.87	35.71	34.85	35.86	32.62	31.68	36.51	34.92	36.49	34.91
Coastguard (MSE)	144.8	99.6	93.0	141.2	108.6	134.7	103.9	125.0	96.5	86.4	124.3	104.4	126.2	98.8
Coastguard (PSNR)	26.52	28.15	28.45	26.63	27.77	26.84	27.96	27.16	28.29	28.76	27.19	27.95	27.12	28.18
Akiyo (MSE)	20.8	30.4	39.8	10.4	13.5	10.8	13.8	9.2	27.8	34.9	7.4	13.4	7.4	13.7
Akiyo (PSNR)	34.95	33.31	32.13	37.98	36.83	37.81	36.74	38.50	33.68	32.70	39.43	36.85	39.43	36.77
Average MSE	69.4	57.6	59.2	62.3	51.7	60.4	50.2	58.0	54.5	54.6	57.0	50.1	57.4	48.5
PSNR (Avg MSE)	29.71	30.53	30.41	30.18	31.00	30.32	31.13	30.50	30.77	30.76	30.57	31.13	30.54	31.27

tiplexed together at a combined bit-rate of 120 kbps. Carphone and Coastguard are of higher motion than Grandma and Akiyo. Note that the MSE and its corresponding PSNR for Akiyo are slightly different in Tables 3.1 and 3.2 for the cases where the video streams are encoded separately using dual-frame video coding (eLTR_EB, LTR_EB, eLTR_RC, LTR_RC). Depending on the number of video streams to be multiplexed, the starting LTR frame number is different. The remaining LTR frames are dependent on the starting LTR frame number. So, the overall performance is slightly different. Methods involving equal slope improve the performance of Carphone and Coastguard at the expense of Grandma and Akiyo. Use of LTR frames improves the quality of all the videos. Again, all rate control methods perform better than the corresponding ones without rate control. The performance of multiplexing methods improves by finding the LTR location using the motion activity detection. STR_RC performs worst among all the rate control methods and LTR_ES_RC performs the best and is about 0.77 dB better than STR_RC. In this case, the performances of SF_RC and STR_ES_RC are quite similar. Although we expect STR_ES_RC to outperform SF_RC because STR_ES_RC uses equal slope allocation which would be optimal on a per frame basis if the RD curves were continuous, the actual performances are quite similar due to the discrete RD curve, which means STR_ES_RC chooses a discrete-valued operating point that is close to but usually not equal to the continuous-valued optimal one. In general, video quality is improved by using dual-frame video coding with equal slope allocation and rate control.

The observed MSE reduction will be less if videos with similar motion levels are multiplexed together. In such a case, all the videos will gain by using dual-frame video coding and rate control. But the advantage of using equal slope allocation will be limited since the complexity of the videos is similar, so there will not be a huge MSE reduction for one video at the expense of a small MSE increase for some other video. For very high motion videos, even the dual-frame video coding method with high-quality LTR frames fails to achieve large MSE reductions.

3.6 Conclusions

In this chapter, we proposed and compared various methods for allocating bit-rate for multiple video streams using dual-frame coding. We considered the three scenarios of (a) no delay constraint, (b) separate encoder output buffer constraints, and (c) a joint delay buffer for all the multiplexed video streams. The main contributions of this chapter are as follows:

1. Firstly, by using the activity measurement algorithm to detect when the LTR frame is becoming obsolete, we developed a simple algorithm for adaptive selection of LTR frame location, and this was shown to improve the performance of dual-frame video coding by reducing the MSE of almost all videos compared to evenly spaced LTRs.

2. Secondly, with regard to the multiplexing problem, we have made two main contributions:

   - The consideration of RD properties for performing bit-rate allocation by the equal slope technique is well established. This technique allocates more bits to a video that is going through high motion by taking bits from low motion videos, resulting in a large MSE reduction for high motion videos with a small increase in MSE for low motion videos. Our contribution was to combine this approach with dual-frame coding with high-quality LTR frames, in which the LTR frames are allocated bits based on motion activity, and other frames are allocated bits using the equal slope technique.
   - The buffer-constrained rate control method which works well for individual video streams was modified for a multiplexing scenario, in that the LTR frames in various streams can be slightly delayed or advanced in their locations in order to avoid having them occur at the same time, thereby overflowing the buffer.

In summary, we proposed multiplexing video streams using the triple advantages of (a) dual-frame coding with high-quality LTR frames, (b) modified rate

control which accommodates high-quality LTR frames, and (c) equal slope bit allocation modified to accommodate high-quality LTR frames. This new method was shown to outperform existing methods for multiplexing video streams, including the superframe method equipped with rate control.

3.7 Acknowledgement

Chapter 3 of this dissertation contains material that appears in M. Tiwari, T. Groves, and P. Cosman, "Delay constrained multiplexing of video streams using dual-frame video coding", *IEEE Transactions on Image Processing*, Vol. 19, No. 4, pp. 1022-1035, April 2010. I was the primary author and the co-authors Dr. Pamela Cosman and Dr. Theodore Groves directed and supervised the research which forms the basis for Chapter 3.

Chapter 4

Competitive equilibrium bit-rate allocation for multiple videos

4.1 Introduction

For some existing methods of transmitting multiple video streams [25, 27, 28, 43], improving the overall or average quality is the goal, similar to the method discussed in Chapter 3. Overall quality improvement can be achieved by exploiting the relative complexity of different video streams at every moment. However, not all video streams may benefit from the multiplexing process. Generally, the quality of high complexity videos improves at the expense of reduced quality of low complexity videos.

None of the methods discussed in Section 1.3 for transmission of multiple video streams aims to improve the quality of *all* the users individually. If the alternative is an equal distribution of the total available resource, a user may see his video quality degrade if he participates in any one of the resource allocation methods described above. Thus, it is reasonable to expect that a user might choose not to participate in a specific resource allocation process if he is unsure of his video quality improvement. The user might decide he will be better off acting individually and receive a fixed share of the resource.

In this chapter, the goal is that no video stream will suffer quality degra-

dation by participating in the multiplexing process, compared to independent encoding. As we will see later, while the method does not offer a guarantee that no video will suffer a quality degradation from multiplexing compared to individual encoding, in practice a quality degradation is extremely unlikely, and did not occur at all in our experimental cases. We use a competitive equilibrium approach for bit-rate allocation among various video users based on their video complexity. A competitive equilibrium consists of an allocation and a price vector at which (a) each user's allocated consumption vector maximizes his utility given a budget constraint defined by the equilibrium prices and his initial wealth, and (b) at the equilibrium prices, the aggregate supply of each resource that was endowed initially to the users equals the aggregate demand for it.

For video multiplexing, we define the utility in terms of mean squared error (MSE), and the resource at any time-slot is the available bit-rate. If the video complexity is high then more of the resource is required to attain some level of utility compared to the amount required for achieving the same level of utility for a less complex video. The equilibrium price is the rate of exchange between current bit-rate and expected future bit-rate. As will be discussed later, it reflects the number of current bits a user must give up to get one expected bit some time in the future or equivalently, how many current bits a user can get by giving up one expected bit some time in the future. The price varies with the relative video complexity between the current time-slot and future time-slots. If the average complexity of all the videos in the current time-slot is greater than the estimated average complexity in the future, then the bits at the current time-slot are more valuable and the price will be greater, and vice versa if the average complexity of all the videos in the current time-slot is less than the estimated average complexity of the videos in the future.

The general equilibrium approach views the economy as a closed and interrelated system in which the equilibrium values for all the variables are determined simultaneously. Our method for implementation of competitive equilibrium selects an expected efficient, or Pareto optimal, allocation of bit-rate for multiple videos. At a Pareto optimal solution, there is no alternative way of allocating resources

that makes some users better off without making some other users worse off. By computing the expected competitive allocation in the Edgeworth box, a common tool in economics for equilibrium analysis, we find a point where all users, in expectation, perform better or at least as well as what they could achieve independently. This method exploits gains in quality that can be achieved by trading bits across time rather than merely reallocating bits among the video streams at any time, as is done with current methods of multiplexing. In our preliminary work [49], the competitive equilibrium at any time-slot was solved by reducing the entire video sequence to two equal length time-slots, one for the current time-slot and one for the average of all the remaining time-slots. In this chapter, we trade the bits for the current time-slot against all the remaining time-slots (with the same expected rate-distortion (RD) curve) and this gives more flexibility in trading.

Figure 4.1 shows a general block diagram for joint bit-rate allocation for multiple video users. Each user passes RD information to a central controller. The central controller computes the competitive equilibrium for the users simultaneously and sends the allocated bit-rate information to the corresponding encoder. Each encoder uses this information to encode his video. Encoded bitstreams are multiplexed and transmitted through a shared channel to the decoder. At the decoder, the bitstreams are demultiplexed and sent to the corresponding decoder to decompress the bitstream and display. We operate the competitive equilibrium bit-rate allocation at the level of a group of pictures (GOP). However it can be operated at other granularities.

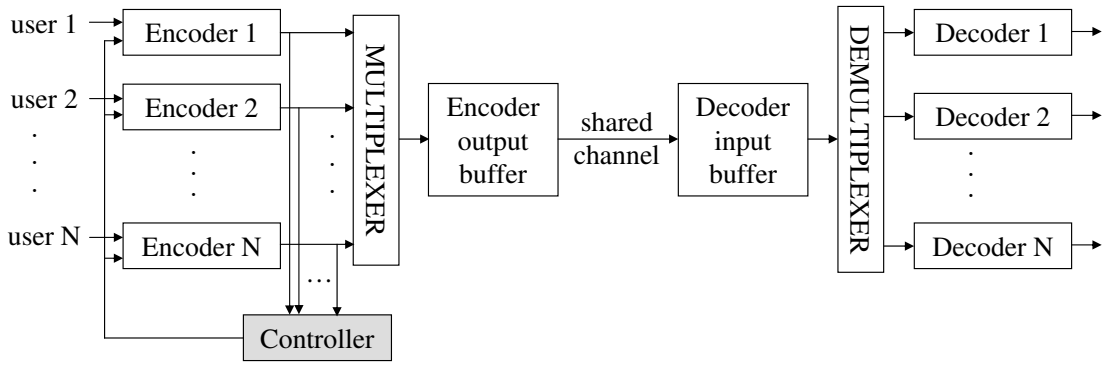


Figure 4.1: Bit-rate allocation for multiple video streams using a central controller.

In this work, we ignore any incentive the users might have to communicate false information in order to acquire additional resources. For example, a user with low complexity video in the current time-slot and high complexity in the next time-slot might overstate his current complexity relative to the next time-slot, thereby lowering the relative bit-rate price in the next time-slot, and thus enabling more favorable allocations in the current and next time-slots. The mechanism we discuss here is susceptible to this type of strategic manipulation.

However, such incentive issues are not relevant for applications in which separate users share the same objective (i.e., form a team [50], for example, as in a military scenario). Also, as is extensively discussed in the economic literature, the advantages to such strategic manipulation become vanishingly small as the number of users being multiplexed increases. The larger the number of users, the less any individual user can affect the computed competitive equilibrium prices. Therefore, we assume all the users inform the controller of their true video RD characteristics. The controller then uses the competitive equilibrium approach to determine the bit-rate allocation.

The rest of the chapter is organized as follows: Section 4.2 describes the Edgeworth box for illustrating competitive equilibria. Section 4.3 describes competitive equilibrium bit-rate allocation methods for multiple video streams. Results are given in Section 4.4, and Section 4.5 concludes the chapter.

4.2 Edgeworth Box for Competitive Equilibrium

In this section, we briefly describe a competitive equilibrium and its Edgeworth box representation. Interested readers are encouraged to read [51] for further details.

The *Edgeworth box* [52] is a graphical tool for exhibiting Pareto optimal allocations and illustrating a competitive (Walrasian) equilibrium in a pure exchange economy [51], in which no production is possible and the commodities that are ultimately consumed are those that individual users possess as initial endowments. The users trade these endowments among themselves in a market for mutual ad-

vantage. For the competitive equilibrium analysis, we start with an example using two users who exchange quantities of two goods with each other for their mutual advantage. This simple case is amenable for graphical analysis using the tool known as an Edgeworth Box. Later, for multiplexing many video streams, we will apply the theory of competitive equilibrium for exchanging bit-rate to improve the quality of all the video streams.

Consider two users ($i = 1, 2$) and two goods ($j = 1, 2$). User i's consumption vector is $x_i = (x_i^1, x_i^2)$, i.e., user i's consumption of good j is $x_i^j \geq 0$. Each user i is initially endowed with an amount $c_i^j \geq 0$ of good j. The total endowment of good j in the economy is denoted by $c^j = c_1^j + c_2^j$, assumed to be strictly positive. An allocation $x \in \mathbf{R}_+^4$ is an assignment of a non-negative consumption vector to each user: $x = \{x_1, x_2\} = \{(x_1^1, x_1^2), (x_2^1, x_2^2)\}$. We say that an allocation is *nonwasteful* and *feasible* if $x_1^j + x_2^j = c^j$ for all goods j (the total consumption of each good is equal to the economy's aggregate endowment of it).

In the Edgeworth box, user 1's quantities are measured with the southwest corner as the origin (O_1), shown in Figure 4.2. User 2's quantities are measured using the northeast corner as the origin (O_2). For both the users, the horizontal dimension measures quantities of good 1 and the vertical dimension measures quantities of good 2 from their respective origins. The width and height of the box are c^1 and c^2, the economy's total endowment of goods 1 and 2. The initial endowment point is given by $c = \{(c_1^1, c_2^1), (c_1^2, c_2^2)\}$. Any point x in the box represents a division of the total endowment between users 1 and 2, as shown in Figure 4.2.

User i's wealth is defined by the market value of his goods endowed initially. Suppose users can buy or sell these goods in the market for prices p^1 and p^2. In general equilibrium theory, the wealth of a user is derived internally by the value of the prices. For any price system $p = (p^1, p^2)$ and initial endowment, the budget set for user i is:

$$B_i(p) = \{x_i \in \mathbf{R}_+^2 : p^1.x_i^1 + p^2.x_i^2 \leq p^1.c_i^1 + p^2.c_i^2\} \tag{4.1}$$

The budget set for user i is the set of all consumption vectors x_i which user i can afford at price p. The budget sets for two users in an Edgeworth box are shown in Figure 4.3. A line drawn through the initial endowment with a slope

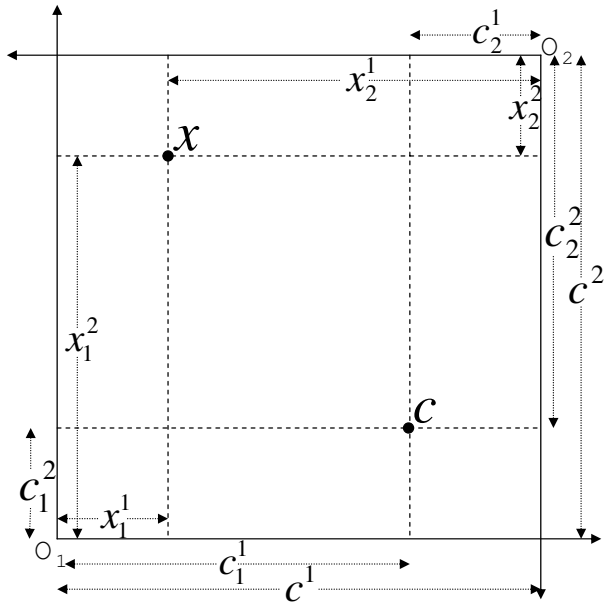


Figure 4.2: An Edgeworth box for two users and two goods.

of $-(p^1/p^2)$ is the *budget line*. User 1's budget set (denoted by $B_1(p)$) consists of all the non-negative vectors below and to the left of the budget line. The area on the other side of the budget line is the budget set for user 2 (denoted by $B_2(p)$). Any total allocation of the two goods on the budget line will be affordable at price system p to both the users simultaneously.

Given $c_i = (c_i^1, c_i^2)$, user i can calculate his utility $U_i(c_i)$, which is a measure of "goodness" or satisfaction with the consumption vector c_i. The locus of all x_i yielding the same utility $U_i(x_i) = u_i$ is called an *indifference curve* of user i. The family of all indifference curves is the collection of level sets of the utility function $U_i(x_i)$, as shown in Figure 4.4. As we move away from his origin, the utility associated with successive indifference curves for user i increases, as more of both goods increases utility. In Figure 4.4, considering the indifference curves for user 1, the utility for user 1 that is associated with the indifference curve that passes through $c1$ is higher than the utility associated with the indifference curve through c. The same is true for user 2. For the purpose of explaining the Edgeworth box, we assume that these curves are convex. For the problem of allocating bit-rate among multiple video streams, we will discuss the validity of this assumption in

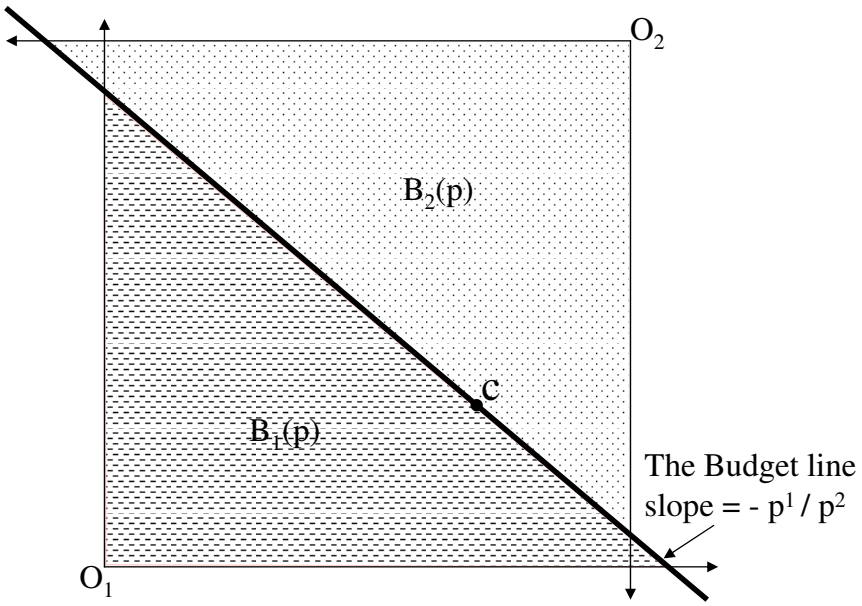


Figure 4.3: Budget sets in an Edgeworth box for two users and two goods.

the next section.

Consider the family of indifference curves for each user, consisting of all indifference curves through every allocation. Under our assumptions of convexity and smoothness, each indifference curve for a user will be tangential to one indifference curve of the other user at some point. The points where the indifference curves for both users are tangential to each other are *Pareto optimal* allocations. At these allocations, it is not possible to increase the utility of one user without decreasing the utility of the other user. The set of all Pareto optimal allocations is the *Pareto set*. It is a curve that connects all the Pareto optimal allocation points in the Edgeworth box from one origin to the other origin. The part of the Pareto set where both users do at least as well as at their initial endowments is called the *contract curve* (Figure 4.5). The contract curve lies between the indifference curves for both the users passing through the initial endowment. Free and unfettered bargaining between the users will result in some point on the contract curve as these are the only points at which both users do at least as well as at their initial endowments and for which there is no alternative further trade that can make both users better off [51]. This is shown in Figure 4.5.

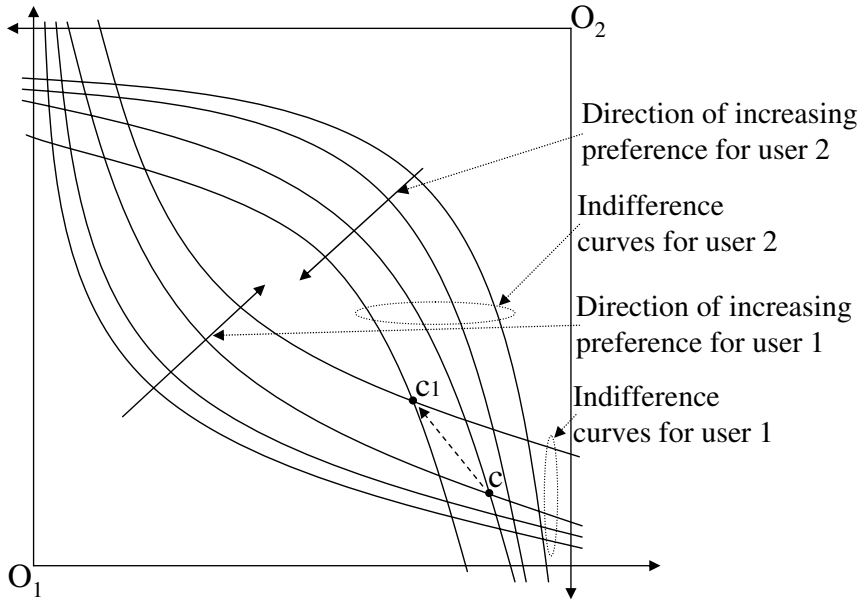


Figure 4.4: Preferences in the Edgeworth box.

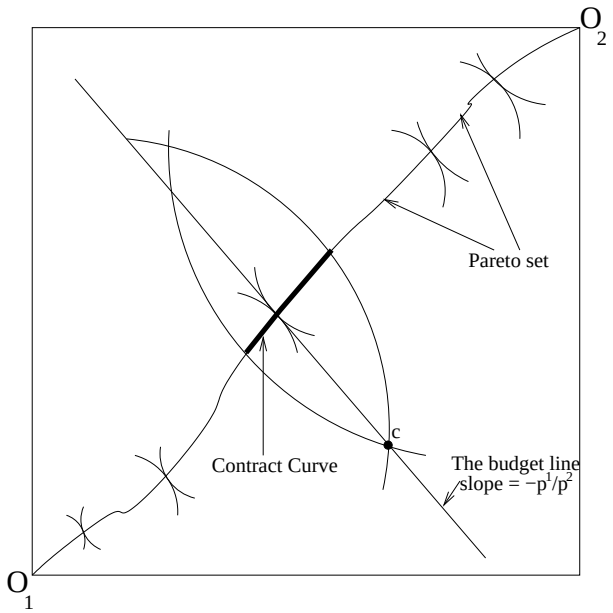


Figure 4.5: The Pareto set and the contract curve in the Edgeworth box.

Given a price vector p, a user demands his most preferred allocation in his budget set. The most preferred point is the point where the budget line is tangential to one of his preference curves. As p is varied, the budget line pivots around the initial endowment point c, and the quantity demanded by a user will be a set of points where different budget lines are tangent to different preference curves. The locus of the user i's optimal choice given current price (which defines current wealth) is known as the *offer curve* (OC_i) and is shown in Figure 4.6. The offer curve always passes through the endowment point because, for any p, the initial endowment is affordable for the user and the tangent to the indifference curve through the endowment point defines a price p at which the endowment is the most preferred allocation in the corresponding budget set.

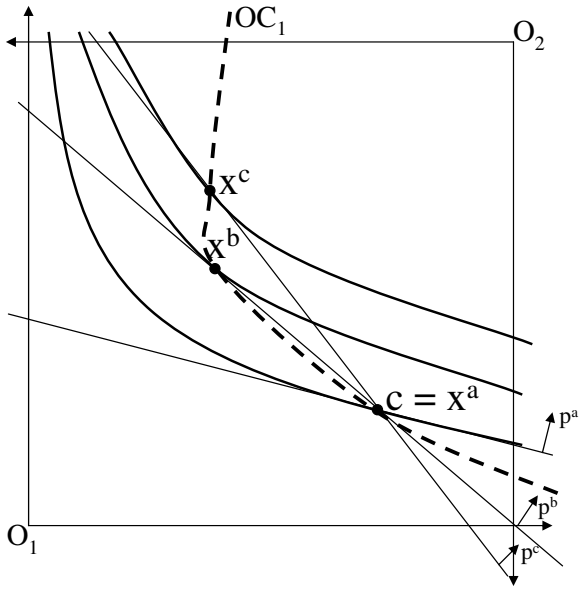


Figure 4.6: Offer curve for user 1. Budget lines p^a, p^b, and p^c are tangential to various indifference curves at x^a, x^b, and x^c respectively.

A *competitive equilibrium* for an Edgeworth box economy is a price vector p^* and an allocation $x^* = \{x_1^*, x_2^*\}$ such that

$$U_i(x_i^*) \geq U_i(x_i') \qquad \forall\ x_i' \in B_i(p^*), \quad \forall\ i = 1, 2 \tag{4.2}$$

and

$$\sum_{i=1}^{2} x_i^{j*} = c^j \quad \forall \; j = 1, 2 \tag{4.3}$$

At an equilibrium, each user i's demanded bundle at price vector p^* is x_i^* and each user's net demand for a good is exactly matched by the other's net supply. The intersection of a budget line and contract curve, where the budget line is also tangential to the indifference curve for both the users on the contract curve, defines a competitive equilibrium. At this point, the offer curves for both the users intersect at a point on the contract curve (Figure 4.7). At an equilibrium point, both users are better off compared to their initial endowment. Under our assumptions, at least one competitive equilibrium will exist for every initial endowment allocation. More details about the Edgeworth box and competitive equilibrium can be found in [51].

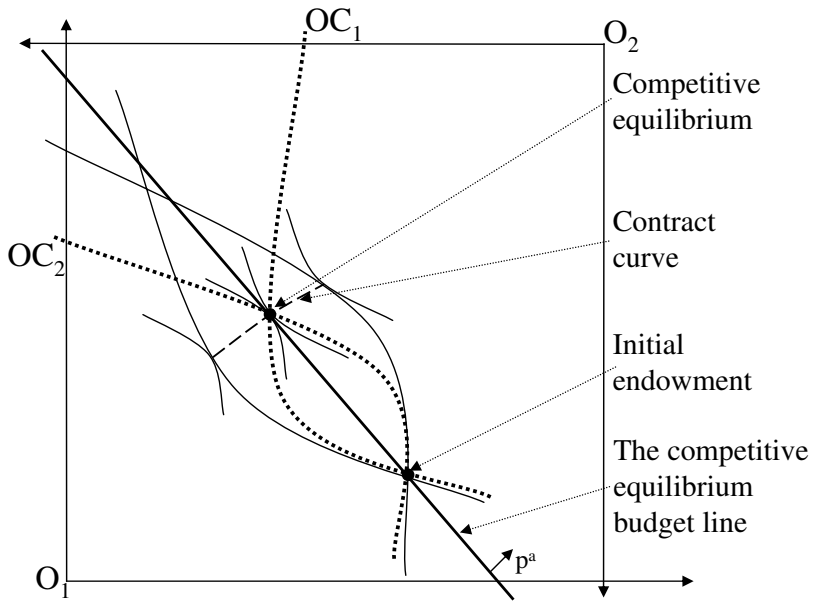


Figure 4.7: A competitive equilibrium allocation. The offer curves for both the users intersect at the competitive equilibrium allocation. One of the indifference curves for both the users is tangential to a budget line at this allocation.

4.3 Competitive equilibrium approach for video multiplexing

In this section, we explain the competitive equilibrium approach for bit-rate allocation among various video streams. Suppose there are N video users. The video stream of each user is divided into T time-slots (TS). In this work, we consider one TS to be one GOP, but a TS can be larger or smaller than a GOP. We will use the terms GOP and TS interchangeably. We assume that the videos are synchronized at the GOP level. Such synchronization can be achieved by a small amount of buffering of the input videos at the expense of a small amount of delay. For different GOP sizes, the synchronization can be achieved by the formation of a 'Super GOP' as described in [25], where a super GOP is the least common multiple (LCM) of GOPs for all the users. The problem of synchronization does not exist when a TS is of a frame size.

We extend the concept of the Edgeworth box from two users to N users and from two goods to T TS. The quantity of each good is represented by the number of bits available in each TS. A user i in TS t is initially endowed with c_i^t bits. Therefore, the total number of bits available in TS t is

$$c^t = \sum_{i=1}^{N} c_i^t \tag{4.4}$$

The users compete to receive bits from the pool of c^t bits in TS t. Let $U_i(x_i^1, x_i^2, ..., x_i^T)$ be the utility for user i. Therefore, the optimization problem for user i is given by

$$\max_{\{x_i^t\}} U_i(x_i^1, x_i^2, ..., x_i^T) \quad s.t. \quad \sum_{t=1}^{T} p^t.x_i^t \;=\; \sum_{t=1}^{T} p^t.c_i^t \tag{4.5}$$

and the constraint over all users is

$$\sum_{i=1}^{N} x_i^t = c^t \qquad \forall t = 1 \; to \; T \tag{4.6}$$

where $(p^1, p^2, ..., p^T)$ are the competitive equilibrium prices. The solution $\{x_i^{t*}\}$ obtained from Eq. 4.5 and Eq. 4.6 is the competitive equilibrium bit-rate allocation.

The above method would work for archived videos where the utility function over all TS is available in advance. However, for real-time applications, a limited amount of video is available and it is generally not possible to achieve a global competitive equilibrium solution. Thus, we reduce the problem to a sequence of problems, each of which solves for a competitive equilibrium for the current TS and a representative of all future TS. For video streams, we define the utility to be the negative of MSE. We generate the RD curve for each TS by calculating the MSE at different bit-rates. Note that the complexity of generating the RD curve can be reduced by using the method described in [46].

Suppose that, in a two user system, user 1 and user 2 each has an initial endowment of 500 bits in TS 1 and in TS 2. Therefore, a total of 1000 bits are available in each TS. If the RD curves for the two users are such that giving 600 bits to user 1 in TS 1 and 400 in TS 2 (and vice versa for user 2) produces a more favorable total MSE than the equal initial endowment, then the Edgeworth box approach would favor this allocation over the initial one.

While trading across TS is the basic idea behind our approach, often adjacent TS have similar RD curves. Therefore, little benefit can be gained by trading bits between adjacent TS for any two users. One would like to trade between the current encoding TS and some other TS widely separated in time. But, since the specific RD curve for a distant TS is typically not known in a real-time application, we consider trades between the current encoding TS and an *expected* or *representative* RD curve for all the future TS.

Specifically, for our method, in each TS for each user, the central controller will reoptimize the decision for the current and all future TS using estimated values for future RD curves. Assuming future TS are identical in expectation (the future environment is perceived as stationary), then each TS's decision problem is just an optimization problem with two decisions only - the allocation x_i^t for the current TS and $\bar{x}_i^t$, the common allocation for each of the remaining $(T-t)$ TS. We start at the first TS and sequentially process each TS in the same manner. Using the estimated utility for the future TS, the optimization problem for user i becomes

$$\max_{x_i^t, \bar{x}_i^t} U_i(x_i^t, \bar{x}_i^t)$$
$$s.t. \quad p^t.x_i^t + (T-t).\bar{p}^t.\bar{x}_i^t \;=\; p^t.c_i^t + (T-t).\bar{p}^t.\bar{c}_i^t \tag{4.7}$$

where p^t is the equilibrium price for the current TS and $\bar{p}^t$ is the estimated equilibrium price of the remaining TS. $\bar{c}_i^t$ is the average initial endowment for user i for TS $t+1$ to T.

As given in [45], the RD curve for user i in TS t is fitted by

$$D_i^t(R_i^t) = a_i^t + \frac{b_i^t}{R_i^t + d_i^t} \tag{4.8}$$

where R_i^t is the number of bits and D_i^t is the MSE distortion for TS t in video stream i. We use the unconstrained nonlinear minimization approach to find a_i^t, b_i^t, and d_i^t, the coefficients for generating this curve-fitting model. Other curve-fitting models are available in the literature [46]. Note that the model used in Eq. 4.8 is convex. The convexity of the RD curves is an empirical observation. Frequently in the previous literature, convexity is either empirically observed or is assumed to hold. All the videos investigated in this chapter exhibit this property. Were it not to be the case, the computed competitive equilibrium solution in the Edgeworth box would not necessarily be an efficient solution for bit-rate allocation.

Using Eq. 4.8, the utility function is represented by

$$U_i(x_i^t, \bar{x}_i^t) = -(a_i^t + \frac{b_i^t}{x_i^t + d_i^t}) - (T-t).(\bar{a}_i^t + \frac{\bar{b}_i^t}{\bar{x}_i^t + \bar{d}_i^t}) \tag{4.9}$$

that is, the negative sum of the MSE in TS t and the estimated weighted MSE for the remaining $(T-t)$ TS. Then the indifference curve through the initial endowment at TS t can be derived as

$$\begin{aligned} -(a_i^t + \frac{b_i^t}{x_i^t + d_i^t}) - (T-t).(\bar{a}_i^t + \frac{\bar{b}_i^t}{\bar{x}_i^t + \bar{d}_i^t}) = \\ -(a_i^t + \frac{b_i^t}{c_i^t + d_i^t}) - (T-t).(\bar{a}_i^t + \frac{\bar{b}_i^t}{\bar{c}_i^t + \bar{d}_i^t}) \end{aligned} \tag{4.10}$$

for different combinations of x_i^t and $\bar{x}_i^t$. Since the RD curves for both the current TS and the average for the future TS are convex, these indifference curves are also convex in nature.

A competitive equilibrium is found by solving

$$\max_{x_i^t, \bar{x}_i^t} -(a_i^t + \frac{b_i^t}{x_i^t + d_i^t}) - (T-t).(\bar{a}_i^t + \frac{\bar{b}_i^t}{\bar{x}_i^t + \bar{d}_i^t})$$

$$s.t. \;\; p^t.x_i^t + (T-t).\bar{p}^t.\bar{x}_i^t = p^t.c_i^t + (T-t).\bar{p}^t.\bar{c}_i^t, \quad \forall\, i = 1\ to\ N \tag{4.11}$$

The Lagrangian expression for user i is

$$\begin{aligned} L_i = &-(a_i^t + \frac{b_i^t}{x_i^t + d_i^t}) - (T-t).(\bar{a}_i^t + \frac{\bar{b}_i^t}{\bar{x}_i^t + \bar{d}_i^t}) + \\ &\lambda_i(p^t.c_i^t + (T-t).\bar{p}^t.\bar{c}_i^t - p^t.x_i^t - (T-t).\bar{p}^t.\bar{x}_i^t) \end{aligned} \tag{4.12}$$

The constraints on the total available bits in each TS are given by

$$\sum_{i=1}^{N} x_i^t = c^t \tag{4.13}$$

$$\sum_{i=1}^{N} \bar{x}_i^t = \bar{c}^t \tag{4.14}$$

By differentiating L_i with respect to x_i^1, x_i^2, and λ_i, equating the results to 0 and solving for x_i, and substituting in Eq. 4.13, we get

$$\sum_{i=1}^{N} \sqrt{\frac{b_i^t}{p^t}}.(\frac{p^t.(c_i^t + d_i^t) + (T-t).\bar{p}^t.(\bar{c}_i^t + \bar{d}_i^t)}{\sqrt{p^t.b_i^t} + (T-t).\sqrt{\bar{p}^t.\bar{b}_i^t}}) - \sum_{i=1}^{N} d_i^t = c^t \tag{4.15}$$

To determine the competitive equilibrium, we need to find the equilibrium prices, p^t and $\bar{p}^t$ that solve Eq. 4.15. Since the solution of Eq. 4.15 is homogeneous of degree 0 in prices, we only need to find an equilibrium price ratio $p^t/\bar{p}^t$. Therefore, without loss of generality, we may take $\bar{p}^t = 1$ and solve Eq. 4.15 numerically for p^t. With p^t, we find x_i^t and $\bar{x}_i^t$ which comprise a competitive equilibrium.

To predict the future average RD function, we consider several alternatives that differ in the information assumed to be held by the user at the time he makes the forecast. In all cases, the user will use the competitive equilibrium approach to calculate the allocated bit-rate for the current TS with respect to the forecasted average RD function for the future. Suppose we have information about the average RD function for a video user i over *all* TS (1 to T). Then we can use such information to calculate the bit-rate demand at a competitive equilibrium

for a user in the current TS by trading the bits with the average RD function for all the TS. We call this method of bit-rate allocation **ALL_TS** which assumes knowledge of the average RD function over all the TS for a user. The average RD curve in ALL_TS is approximated by averaging the individual coefficients (a, b, and d) separately for a user over all TS. The coefficients generated by actually averaging the RD curves for all the TS of a video are extremely close to the average coefficients.

A user is assumed to always know the actual RD function for the current TS t and all past TS (1 to $t-1$). Given the average RD function for all TS, he can calculate the average RD function for the *remaining* TS (**REM_TS**). With the information on the current RD function and average RD function for the remaining TS, the central controller uses Eq. 4.15 to calculate the competitive equilibrium price and bit-rate allocation for the current TS. The average RD curve in REM_TS is approximated by averaging the individual coefficients (a, b, and d) separately for a user over all the remaining TS. Both ALL_TS and REM_TS are *ex ante* approximation models where we assume some information about the future video in advance.

Suppose a user has no knowledge about future TS (*ex post*) (as is the real-time case) but assumes that the video is a stationary process at the GOP level. Future TS properties (for example, complexity) can be estimated by looking at the past. We calculate the bit-rate demand for the current TS using the average of all *previous* TS (**PRE_TS**) as the estimate of future TS. This method would be expected to work well for long videos but may not work for short videos if the previous TS are very different from the future TS. If averaged over a sufficiently long interval, the complexity for most video streams can be assumed to be almost stationary. The assumption about stationarity is important for PRE_TS to work well. The average RD curve in PRE_TS is approximated by averaging the individual coefficients (a, b, and d) separately for a user over all the past TS.

To approximate an upper bound on video quality improvement, we consider a method in which each user has full information about his RD curves in all TS, and proposes to divide the bits among all the TS based on his relative complexity. This

can only be done for archived videos where the coefficients of the RD curves are calculated offline. Each video stream uses this criterion for bit allocation among its TS independently. Since the total number of bits in each TS is given by the initial endowment, we normalize the number of bits allocated to each video stream by the total available bits for a TS (**FUL_TS**). Note that, for this method, each user attempts to allocate bits across TS but does not trade with other users. FUL_TS is, of course, not the real upper bound. The real upper bound would be given by computing the competitive equilibrium for all TS simultaneously (Eq. 4.5 and Eq. 4.6), an extremely large computational problem if there are many users and many TS.

To compare the improvement in video quality of competitive equilibrium bit-rate allocation using the various multiplexing schemes, we consider the equal bit-rate allocation for each user in every TS (**EQL_TS**). Here, each user in every TS receives an equal number of bits to encode his video. Note, for a TS of GOP length, the rate control algorithms used in conjunction with most of the video standards strive to achieve equal bit-rate allocation for all GOPs, similar to EQL_TS. We use equal bit-rate allocation as an initial endowment for our competitive equilibrium allocation.

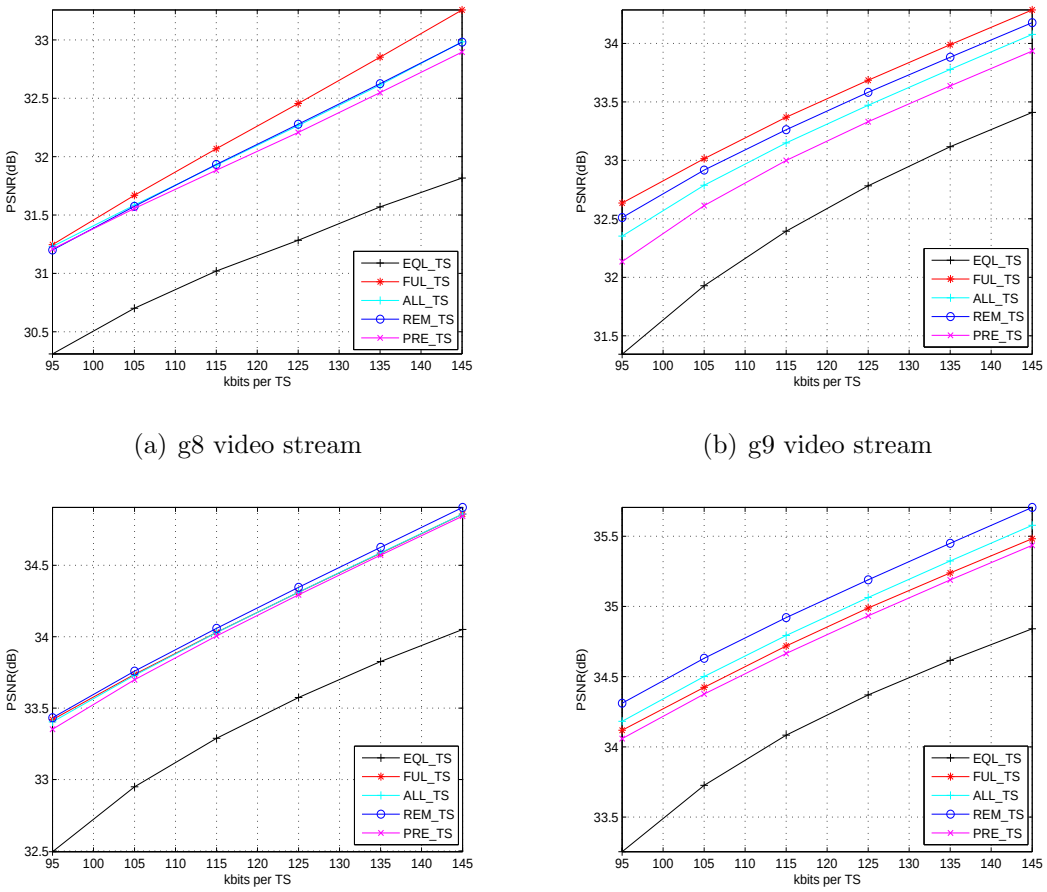
4.4 Results

The simulation was performed using the baseline profile of H.264/AVC [37] reference software JM 11.0 [38]. The GOP size is 15 frames (I-P-P-P). The frames inside a TS are encoded using H.264 rate control [41]. The test video sequences were taken from travel documentaries at a resolution of 352×240 pixels (SIF) and at 30 frames per second. We chose 12 test sequences (denoted by g1 to g12) and each sequence was 250 seconds (7500 frames) in length. The coding parameters such as resolution, GOP size or structure can be changed for any appropriate application as our multiplexing method is independent of such parameters. We considered a lossless channel for transmitting multiple video streams.

Each video sequence contained various types of scenes with varying camera

motions such as zooming and panning. The high motion scenes included dancing, bike racing, and a vegetable market. The low motion scenes showed buildings, maps, sculptures, scenery, etc. Other types of scenes included a flying airplane, showing flowers, people talking, farming, cooking, children playing chess, etc. The videos also had scenes with varying spatial content such as a crowded market, bird's eye view of a city, sky, still water, etc. Each video sequence contained many types of scenes and motions.

Figure 4.8 shows the results of multiplexing four video streams. The five curves in each plot represent the various bit-rate allocation methods for multiplexing video streams as described previously. Each plot shows PSNR versus bit-rate (ranging from 95-145 kbits per TS (190-290 kbps) per user). We calculate the MSE of each frame and average across all frames of a video, then convert it to PSNR. The performance of EQL_TS is worst in all the videos. This is the method used in most video standards for GOP level rate control. For archived videos, the RD curves for all TS are available and we see that FUL_TS performs better than the other methods for most of the videos. The PSNR gain over EQL_TS varies from 0.62-0.87 dB for g11 to 0.94-1.44 dB for g8. However, this method cannot be used for real-time video multiplexing. The bit-rate allocation of EQL_TS is considered to be the initial endowment for the competitive equilibrium bit-rate allocation methods (ALL_TS, REM_TS, and PRE_TS). If we consider that a user knows the average RD curve for all the TS, then the competitive equilibrium bit-rate allocation method, ALL_TS, is used to improve the video quality of all the video streams individually. We found that ALL_TS performs 0.67-1.00 dB for g9 to 0.88-1.17 dB for g8 better than EQL_TS. If a user knows the average RD curve for future TS, this information can be used to improve the video quality, as shown by REM_TS. This method finds a competitive equilibrium point for the current TS when compared to its average RD of the remaining TS. This method improves the quality of each video stream from 0.77-0.94 dB for g10 to 0.87-1.17 dB for g8 over the EQL_TS method. As this allocation method uses the knowledge of the average RD curve for all the future TS, in general, its performance is slightly better than the ALL_TS method.



(a) g8 video stream

(b) g9 video stream

(c) g10 video stream

(d) g11 video stream

Figure 4.8: PSNR variation with bit-rate for four multiplexed video streams.

Finally, we assume that we have no prior knowledge about the video and we estimate the future RD curves by looking at the previous TS. Again we compute the competitive equilibrium for the current TS and the estimated average RD information for future TS based on the average of the previous TS (PRE_TS curve in the figure). This method improves the PSNR from 0.56-0.81 dB for g11 to 0.86-1.08 dB for g8 over EQL_TS. All the competitive equilibrium bit-rate allocation methods improve the quality of all the video streams. We see that, even with absolutely no knowledge about the future video RD characteristics in PRE_TS, we are able to improve the quality of all the video streams by calculating a competitive equilibrium bit-rate allocation. The PSNR improvement over EQL_TS using

this allocation method is in the vicinity of other competitive equilibrium bit-rate allocation methods described in this chapter.

Our competitive equilibrium bit-rate allocation method aims at improving the video quality of each user. If we calculate the MSE averaged across all the users, then our method may perform worse than the methods for minimizing the MSE across all the videos [25, 28, 43]. For example, consider the case for the four videos used in Figure 4.8 at 95 kbits per TS per user. If we maximize the quality averaged over all the videos at each TS using MINAVE from [28] applied to time-domain RD curves, then we achieve 32.91 dB as the average PSNR. On the other hand, the average video quality for the ALL_TS, REM_TS, and PRE_TS methods is 32.65 dB, 32.71 dB, and 32.55 dB, respectively. Clearly, our multiplexing method does not minimize the average distortion. For improving the video quality for each user individually, we incur some performance penalty when compared to the method that explicitly has as its goal the minimization of average distortion over all videos.

Estimates of MSE for future TS by our various estimation methods are shown in Figure 4.9 for the g11 video sequence at a bit-rate of 100 kbits per TS. On the x-axis is the TS index and on the y-axis is the MSE. The four curves show the actual MSE and three types of estimated MSE for the future TS. We encode each TS at the given bit-rate and this is shown as the 'Actual' curve in the figure. When the RD curve is averaged over all the TS (ALL_AVG), then the MSE remains constant. When the RD curve is averaged over the remaining video at any TS, then the calculated MSE is the same as ALL_AVG initially, then deviates, and finally converges with the actual MSE of the last TS. When the average RD curve is estimated from the past TS (PRE_AVG), then the calculated MSE starts with the actual MSE at the beginning and then converges to the ALL_AVG at the end. We compare the actual MSE variation at any TS with the MSE of these averages. The important conclusion that can be derived from this figure is the low variation of all the types of averaging methods compared to the variation in the actual MSE at any TS. When a user calculates his bit-rate requirement for the current TS, then the user actually calculates the usefulness of the bits currently compared to the future. If the video is less complex in the current TS than the

average of the future, then the user demands fewer than the average bits for the current TS in anticipation that he will receive more bits in the future when his complexity is expected to be higher. By trading the bits across time, a user is able to improve his video quality. At any given TS, the demands from some users are less than their average allocation of bits while other users' demands are greater. At a competitive equilibrium allocation for the current TS and the average of their future TS, the expected video quality of all the users is improved.

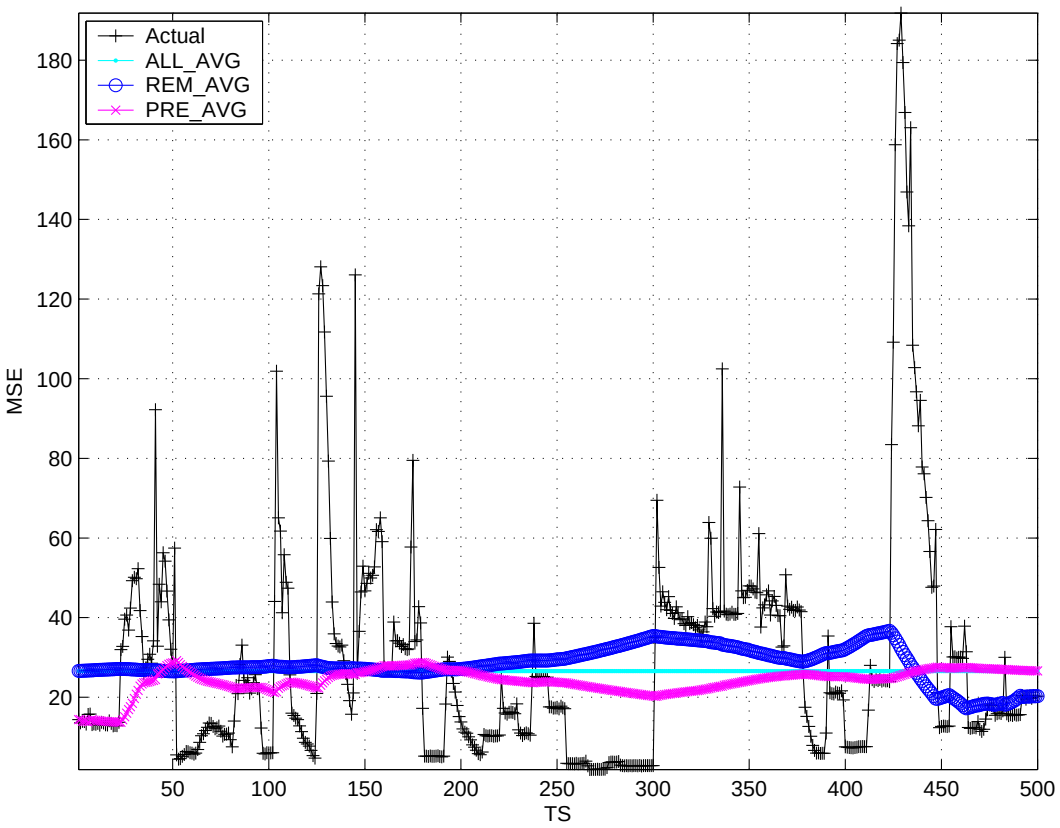


Figure 4.9: Actual MSE and estimates of MSE for future TS by our various estimation methods for g11 video sequence at 100 kbits per TS.

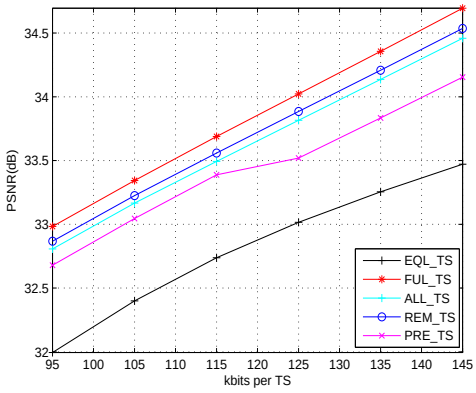
Similarly, Figure 4.10 shows the results for six video streams that are multiplexed together using the methods described above. The PSNR of all the six videos improves for a wide range of bit-rates. For the competitive equilibrium bit-rate allocation methods in these six videos, g8 and g9, in general, produce most PSNR improvement. The improvement is as high as 1.5 dB. The least PSNR improve-

ment is seen for g11 but it is still in the range of 0.60-1.08 dB above EQL_TS. A similar result is shown in Table 4.1 when ten video streams are multiplexed together at an average bit-rate of 100 kbits per TS per user. The table shows the PSNR improvement over EQL_TS by various multiplexing methods.

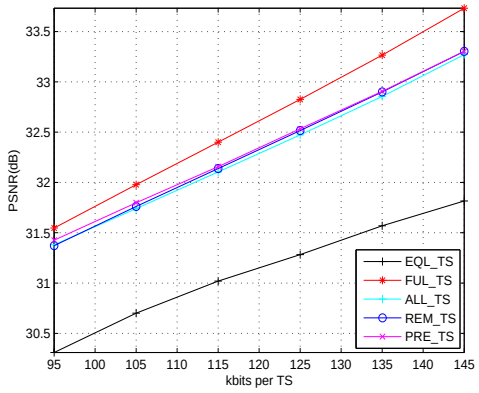
We note that the largest PSNR gain is achieved by finding the competitive equilibrium when there is a lot of fluctuation in the video motion, for example g9. Conversely, the PSNR gain is low if the motion fluctuation in a video stream is low, for example g1. Most of the video streams have significant motion fluctuation and scene changes, so multiplexing them by computing the competitive equilibrium improves their quality substantially.

The performance of PRE_TS depends on the accuracy of the estimation of future TS from past TS. Suppose we have a video whose complexity is monotonically decreasing. Therefore, it would be desirable if the bit-rate demanded in the current TS would be higher than the bit-rate demanded in any future TS. The competitive equilibrium bit-rate allocation will compare the RD function for the current TS and the estimated RD function for the future TS. However, using PRE_TS, the estimated RD curve for the future TS is based on the past TS. The estimate for the future will consistently be higher than the actual future complexity and will be also higher than the actual complexity of the current TS. Therefore, the bit-rate allocation for the current TS at competitive equilibrium for this video will be less than the average allocation for the future TS, which is not the desired result. In contrast, in the REM_TS method, the allocation for the current TS will be more than the average allocation for the future since fewer bits are actually required for the future TS. In such a case, REM_TS and FUL_TS will perform much better than EQL_TS but it is possible that EQL_TS might perform better than PRE_TS. However, in real video examples, we rarely encounter such pathological cases, and our multiplexing methods were found to improve the quality of all of the video streams studied.

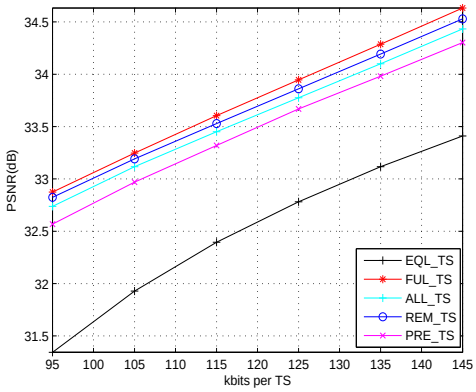
As can be seen from Figures 4.8 and 4.10, all the video streams gain from the multiplexing process. The multiplexing method using the competitive equilibrium borrows bits from a low motion TS of a video and gives these bits to another



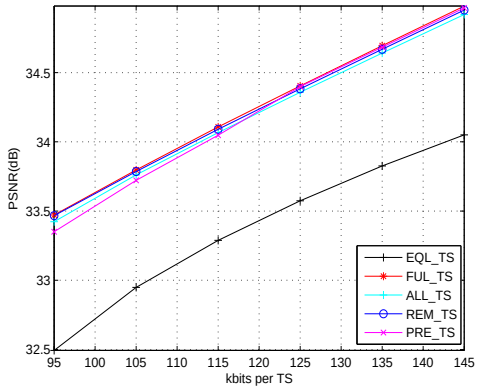
(a) g7 video stream



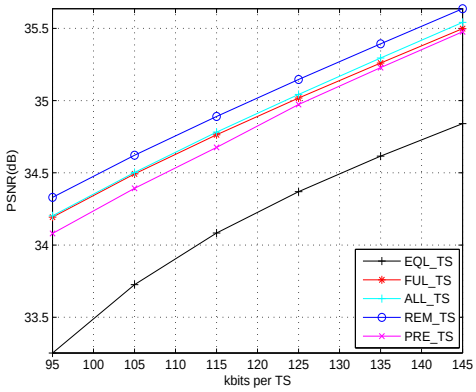
(b) g8 video stream



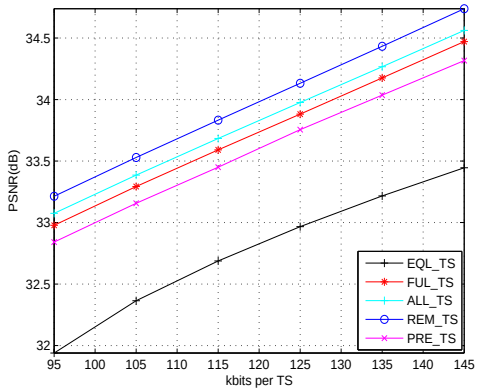
(c) g9 video stream



(d) g10 video stream



(e) g11 video stream



(f) g12 video stream

Figure 4.10: PSNR variation with bit-rate for six multiplexed video streams.

Table 4.1: PSNR (dB) of EQL_TS and improvement over EQL_TS by using various multiplexing methods when 10 video streams are multiplexed together at an average bit-rate of 100 kbits per TS per user

	g1	g2	g5	g6	g7	g8	g9	g10	g11	g12
EQL_TS	34.18	35.91	34.45	34.15	32.21	30.51	31.63	32.72	33.47	32.14
	Improvement over EQL_TS									
FUL_TS	0.35	1.77	0.49	0.74	1.18	1.42	1.52	0.98	0.91	1.06
ALL_TS	0.48	0.76	0.62	0.75	1.08	1.17	1.45	0.96	0.99	1.08
REM_TS	0.48	0.87	0.65	0.87	1.09	1.14	1.41	0.93	1.06	1.29
PRE_TS	0.32	0.77	0.44	0.63	0.97	1.03	1.39	0.91	0.85	0.87

video in the same TS with the expectation of getting bits back later when the need arises. Thus, the multiplexing method exchanges bits between video streams as well as across the TS. This leads to another observation that the quality fluctuation for each video stream is slightly reduced compared to EQL_TS because the high motion TS get more bits than the low motion TS instead of getting the same number of bits for all TS. Figure 4.11 shows the PSNR fluctuation for g9 for all the multiplexing methods. The EQL_TS method has the highest PSNR fluctuation (20.31-44.83 dB) and FUL_TS has the lowest (24.29-42.50 dB). Among the competitive equilibrium bit-rate allocation methods, REM_TS has minimum PSNR fluctuation (22.15-42.50 dB) compared to ALL_TS or PRE_TS. The quality fluctuation can further be reduced by imposing a constraint on maximum and minimum video quality. However, any method of reducing the quality variance comes at the cost of reduction in overall quality as can be seen in [28]. In our method, reduction in the quality variance is achieved by trading the bits across time. Further work can be done in reducing the quality variance for a video while maintaining the overall video quality. Perceptually, we see a huge improvement in the subjective quality by our multiplexing method compared to EQL_TS.

Depending on the prices at every TS, the videos might receive unequal total numbers of bits in the multiplexing process, but all the videos benefit from these multiplexing methods. By changing the encoding technique inside a GOP (e.g., using multiple reference frame prediction or using hierarchical B-frames), along with these multiplexing methods, the overall video quality can be expected to further improve. The PSNR gains are negligible if the videos have similar complexity at every TS. In such a case, the bit-rate requirements for all the videos are similar for the current TS and future TS and so very little trading will take place. Similarly, if each individual video has nearly identical complexity in each one of its TS compared to any other TS, then, even if the videos differ hugely in complexity compared to one another, the competitive equilibrium bit-rate allocation would result in negligible PSNR improvement over EQL_TS. This is because the bit-rate requirement for each video at the current TS is nearly the same as that for the future TS; no user would be willing to trade bits for the current TS with respect to

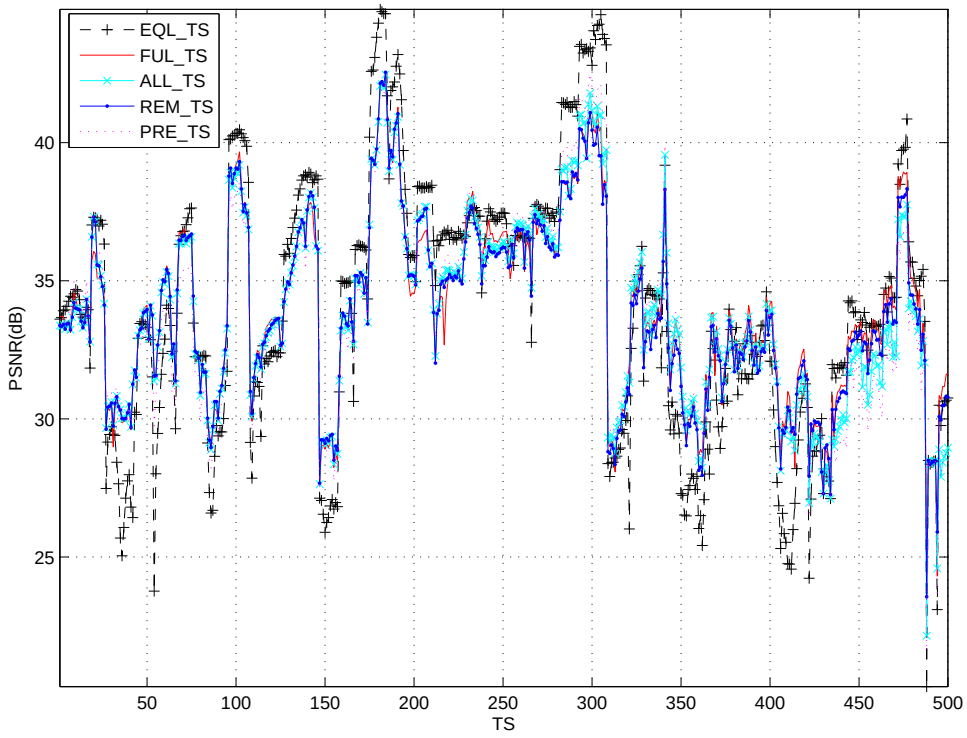


Figure 4.11: Variation of PSNR with TS for g9 video at 100 kbits per TS.

the future TS. Then, all the videos receive almost the same number of bits at each TS. Thus the competitive equilibrium bit-rate allocation would result in negligible PSNR improvement.

The utility function for this chapter was defined in terms of MSE. However, the proposed competitive equilibrium bit-rate allocation method can be applied to any other utility function provided that it is convex. The users would have to communicate their individual utility information instead of RD function and the interpretation of the results would vary depending on the utility function of the users.

4.4.1 Video users with variable start and end times

The results presented above are for N users who are simultaneously transmitting their videos. All the users are present during all TS. This is the same condition used in previous work on video multiplexing [25, 27, 28, 30, 31, 33, 43]. We now consider video streams with different start and end times, so there are different numbers of users involved in the multiplexing at different times.

Let t_i and T_i be the start and end times for user i. The utility function for user i at TS t such that $t_i \leq t \leq T_i$ is given by

$$U_i(x_i^t, \bar{x}_i^t) = -(a_i^t + \frac{b_i^t}{x_i^t + d_i^t}) - (T_i - t).(\bar{a}_i^t + \frac{\bar{b}_i^t}{\bar{x}_i^t + \bar{d}_i^t}) \tag{4.16}$$

A competitive equilibrium at TS t is found by solving

$$\begin{aligned} \max_{x_i^t, \bar{x}_i^t} & -(a_i^t + \frac{b_i^t}{x_i^t + d_i^t}) - (T_i - t).(\bar{a}_i^t + \frac{\bar{b}_i^t}{\bar{x}_i^t + \bar{d}_i^t}) \\ s.t. \;\; & p^t.x_i^t + (T_i - t).\bar{p}^t.\bar{x}_i^t = p^t.c_i^t + (T_i - t).\bar{p}^t.\bar{c}_i^t, \\ & \forall\, i = 1\, to\, N,\; i:\; t_i \leq t \leq T_i \end{aligned} \tag{4.17}$$

The constraint on the total available bits in TS t is given by

$$\sum_{\substack{i=1 \\ i:t_i \leq t \leq T_i}}^{N} x_i^t = c^t \tag{4.18}$$

The equilibrium prices can be found by solving

$$\sum_{\substack{i=1 \\ i:t_i \leq t \leq T_i}}^{N} \sqrt{\frac{b_i^t}{p^t}}.(\frac{p^t.(c_i^t + d_i^t) + (T_i - t).\bar{p}^t.(\bar{c}_i^t + \bar{d}_i^t)}{\sqrt{p^t.b_i^t} + (T_i - t).\sqrt{\bar{p}^t.\bar{b}_i^t}}) - \sum_{\substack{i=1 \\ i:t_i \leq t \leq T_i}}^{N} d_i^t = c^t \qquad (4.19)$$

Thus, the competitive equilibrium bit-rate allocation can be calculated using these prices.

There are two scenarios for the system involving users with different start and end times. In the first, we consider that each user is offered some fixed dedicated bit-rate but then chooses to add their allocation into the communal pool and join a competitive equilibrium allocation process. The total bit-rate at any TS therefore scales up with the number of users in that TS. In the second scenario, users also start and end their video transmission at different times, but the shared channel has a bit-rate that may be constant or variable over time, but in any case does not scale with the number of users. These scenarios are briefly discussed below.

Suppose we have a system where a user is assigned a specific bit-rate when he enters. There are two options available for such users: (a) the user can transmit his video at his given bit-rate, and (b) the user can collaborate with other users, adding his allocation into the communal pool and achieving a competitive equilibrium bit-rate allocation, with the expectation that he will be better off in terms of video quality by doing so. The overall bit-rate depends on the number of users present at any TS and their initial endowments. Our competitive equilibrium bit-rate allocation method was easily extended to such systems to improve the quality of all the users. Using the simulation results for such systems, it was found that the quality improvement using the competitive equilibrium bit-rate allocation depends on the amount of time that the users overlap with each other. As one would expect, the case of large overlaps among large numbers of users produces higher quality improvement. As the number of such users increases in the system for the competitive equilibrium allocation, the performance improvement for collaborating (compared to retaining one's own equal initial endowment) increases since more trading takes place between the users for mutual advantage. The results are largely the same as the case of same start and end times. Current users of the system

know that any additional users who join in will bring their own equal allocation with them for the communal pool, so there is, on the average, a slight improvement when new users join (because of the advantages of having more people involved in trades), and there is a slight disadvantage when people leave.

Table 4.2: PSNR (dB) improvement over EQL_TS by using various multiplexing methods when 10 video streams with different start and end times are multiplexed together at a constant bit-rate of 500 kbits per TS.

	g1	g2	g5	g6	g7	g8	g9	g10	g11	g12
FUL_TS	0.16	0.96	0.39	0.83	1.14	1.13	1.46	1.04	0.98	0.92
ALL_TS	0.49	0.22	0.52	0.94	0.86	0.88	1.32	1.05	1.18	1.01
REM_TS	0.46	0.08	0.63	1.07	0.76	0.45	1.28	1.03	1.20	1.25
PRE_TS	0.34	0.58	0.36	0.79	0.79	1.04	1.20	1.07	0.95	0.73

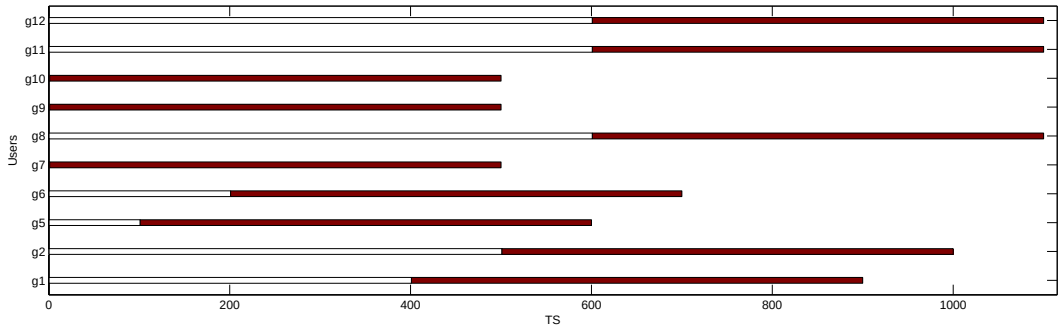


Figure 4.12: Start and end times of 10 video users.

We now consider the second case, where the shared channel has a bit-rate that does not depend on the number of users present. This may be a Constant Bit-rate (CBR) channel or a Variable Bit-rate (VBR) channel. In either case, users enter and leave the system at different times, and the average bit-rate at any TS depends on both the total bit-rate at that TS and on the number of users present. Therefore, the initial endowment to each user varies, and is performed at that TS. In such a case, the competitive equilibrium bit-rate allocation depends on the number of users present at any TS for trading and the total available bit-rate at that TS. If there are many users present, since the total bit-rate does not scale up with the numbers of users, everyone's quality will be worse on the average compared to if there were fewer users, regardless of whether equal allocation or competitive

equilibrium allocation is used. However, as in the previous case, the improvement for collaborating compared to equal allocation generally increases with the number of users present at any TS. However, if there are many users present, the quality of most of them (unless they have a very low complexity TS) is lower than their expected quality for the future, and users would then be unwilling to give away bits in that TS in exchange for future bits. In such a case, little or no trading will take place between the users. Table 4.2 shows the results for multiplexing 10 video streams with different start and end times for a shared constant bit-rate of 500 kbits per TS. The start and end times for each user are given in Figure 4.12. At some TS, there are as many as six video streams, and at other times, there are as few as three. To estimate the average video quality for future TS, we assume that the users know the average bit-rate. At any TS, the competitive equilibrium is achieved for those users who are present at that TS. Even with different start and end times, all users improve their video quality compared to EQL_TS, as shown in Table 4.2. Figure 4.12 and Table 4.2 provide one example of differing start and end times. The results, in general, depend on the distributions of start times and end times, as well as on what users know about these distributions when they forecast their future bit-rate demands.

We note also that our competitive equilibrium model for bit-rate allocation is intended to be an approximation to a 'large' system in which fluctuations in the number of users on the shared channel at any one time will be 'close' to some average number. In that case, variable starting and ending times present no problem because the number of bits allocated to each user initially can be considered to be constant per TS. For all these different scenarios, the computations we make are exactly the same - each user is trading off bits between the current TS and an average of all future TS (which may be different for different users, depending on the length of their video and when they started).

4.5 Conclusion

We discussed various methods for multiplexing video streams for improving the quality of each individual video. Note that because this technique is applicable to both ad hoc networks that employ cluster heads and cellular architectures, it is relevant to both military and commercial scenarios. We considered the competitive equilibrium approach for allocating bit-rate among various video streams. Using an Edgeworth box solution, we graphically showed the process of bit-rate allocation at a competitive equilibrium. A central controller collects rate-distortion information from all the users. The central controller performs the competitive equilibrium calculation for a bit-rate allocation to all the users simultaneously. The final bit-rate allocation is a Pareto optimal solution and all the users do at least as well as they would with an individual allocation. The bit-rate allocation information is sent to the video users, and the users use this information to encode their video streams. We proposed three different methods for estimating the future RD information for a video stream. The estimation of future RD information was used to trade bits for the current TS with respect to the expected bit-rate requirement in the future. All the estimation methods work well for the competitive equilibrium allocation. The results show PSNR improvement for all the video streams. Comparing the decoded videos after multiplexing, we found that the subjective quality was improved by using the competitive equilibrium bit-rate allocation when compared with EQL_TS. Typically, the video quality improvement is clearly visible in high motion parts of a video stream where more bits are allocated in the competitive equilibrium bit-rate allocation methods. Generally, the PSNR improvement depends on the accuracy of estimating the RD information for future TS. The PSNR improvement is greater for the videos with higher motion fluctuation, even though their estimation of the future TS is almost stationary over time. Such videos have varying demand for the current TS with respect to the almost constant demand for the future TS, and so are willing to trade away bits for now in anticipation of gaining bits at some future TS or vice versa.

4.6 Acknowledgements

The authors would like to thank Prof. Marco Tagliasacchi from Politecnico di Milano for providing the data of [28] to compare the results.

Chapter 4 of this dissertation contains material that appears in M. Tiwari, T. Groves, and P. Cosman, "Competitive equilibrium bitrate allocation for multiple video streams", *IEEE Transactions on Image Processing*, Vol. 19, No. 4, pp. 1009-1021, April 2010. I was the primary author and the co-authors Dr. Pamela Cosman and Dr. Theodore Groves directed and supervised the research which forms the basis for Chapter 4.

Chapter 5

Bit-rate allocation for multiple video streams using a pricing-based mechanism

5.1 Introduction

In Chapter 4, we proposed a joint bit-rate allocation scheme based on competitive equilibrium theory that improves the quality of all videos. The quality improvement was achieved by reallocating the bits for each video from those Time-Slots (TS) when a reduction in bit-rate hurts little to other TS when increased bit-rate increases quality the most. The method in Chapter 4 allocated bits among video streams at each TS and within a video stream across TS. This is possible if there are many videos, some of whose quality can be improved by reducing their allocation in one TS for an increased allocation in some other (later) TS, and other videos in the same TS whose quality could be improved by the reverse exchange. The method described in Chapter 4 is a centralized bit-rate allocation method where all the users send their true Rate-Distortion (RD) information to a central controller who, in turn, decides the bit-rate allocated to each user at each TS.

Implementation of these schemes requires communication of specific infor-

mation about individual videos at every TS, namely, their RD curve, or the rate at which quality increases as more bits are received. This complicated information must be communicated accurately. For some applications, this may be problematic. Various decentralized algorithms were discussed in Section 1.3 for joint bit-rate allocation for multiple video streams. A pricing mechanism for multiuser resource allocation in wireless multimedia applications was discussed in Section 1.3. But these methods also result in reduction of video quality for some users, compared to their video quality if all the resource (bit-rate) has been allocated initially.

Rate control using pricing mechanisms has been extensively studied for communication networks [53–59]. Using a pricing mechanism in [53], the proportional fairness criterion was implemented. Such methods are specifically designed for networking problems where link prices, routing, and proportional fairness are of importance in the medium access control layer. These algorithms, however, do not take into account the characteristics of the source. In particular, the dynamics of the video stream will have little effect on the performance of these algorithms. Therefore, applying such algorithms for video transmission may not result in improving the quality of all the video streams. In this chapter, we specifically use a pricing-based bit-rate allocation mechanism for video transmission where multiple video users compete for the resources simultaneously to improve their video quality.

Compared to the method discussed in Chapter 4, we propose a scheme which requires simpler information to be exchanged. This scheme is modeled on price-guided procedures discussed in the economics literature [60] that are characterized as decentralized, as various video transmitters (hereafter users) only communicate their bit-rate demands in response to the bit-rate price announced by a bit-rate allocator (hereafter the allocator) in a TS. By contrast, in a centralized procedure (e.g. [61]), each user communicates the private information that is necessary for the bit-rate calculation (e.g., rate-distortion curves) and the allocator decides on an allocation for all the users. In our decentralized procedure, the allocator adjusts the user's demands to equalize the aggregate allocation to the available supply and announces the price for the next TS. With this price-guided allocation scheme,

instead of using bits at a constant rate, users will increase their demand in TS during which their videos are more complex (e.g., high motion) and reduce their demand in TS of low complexity. Permitting the amount of bit-rate used in each TS to vary increases the efficiency of each user's total bit-rate use by giving more of the resource when it is most valuable (in terms of lowering MSE) and less when it is less valuable. The use of a price to guide users' choices of demand reflects the relative scarcity of available bit-rate in each TS. When all users request more bits than the average, scarcity is greater and the price is higher, thus moderating the demands. Our simulation results show that each user benefits from this price-based decentralized bit-rate allocation mechanism compared to the equal bit-rate allocation to all the users. The performance of this algorithm is comparable to the centralized bit-rate allocation introduced in Chapter 4 where all users send their RD curves to the allocator. The equilibrium price is not achieved in this price-based decentralized allocation method because only one iteration for price adjustment is made. However, the simulations show that the bit-rate price in this method closely follows the equilibrium price.

The rest of the chapter is organized as follows: Section 5.2 provides the general description of the pricing-based decentralized bit-rate allocation process for individual users. Section 5.3 discusses various aspects of the bit-rate allocation process in detail. Simulation results are given in Section 5.4 and Section 5.5 concludes the chapter.

5.2 Pricing-based decentralized bit allocation

Suppose there are N video users sharing the available bit-rate. The video stream of each user is divided into TS. In this work, we consider one TS to be one group of pictures (GOP), but a TS can be larger or smaller than a GOP, similar to the procedure discussed in Section 4.3.

For user n, the video stream starts at TS t_n and ends at TS T_n. The entire system time is set on the basis of the start and end time of all the video streams. The system time starts at T_s when the first user enters the system ($T_s = min(t_n)$)

and the system time ends at T_e when the last user exits the system ($T_e = max(T_n)$). This is shown in Fig. 5.1 for three users. The current TS in the system is referred to as TS t which varies from T_s to T_e. Without loss of generality, we assume the time axis is such that $T_s = 1$. Therefore, the system time starts at TS 1 and ends at TS T, where $T = (T_e - T_s + 1)$. Note that, when we refer to TS t for user n, we assume that $t_n \leq t \leq T_n$.

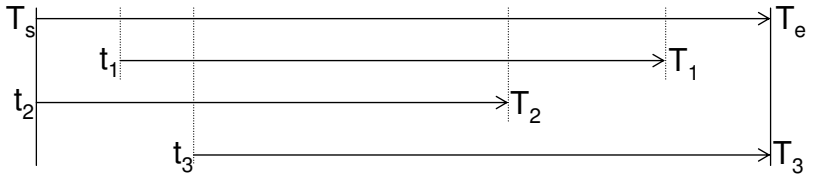


Figure 5.1: Start and end times for three video users.

The utility of user n at TS t, denoted by $U_{n,t}(x_{n,t})$, is taken to be the negative of his MSE at a bit-rate of $x_{n,t}$, given by the RD curve for that TS. Although for the purpose of direct simulations of the algorithm introduced in this chapter, we have taken user's utilities to be directly given by the negative distortion (MSE) by the RD curve, our results will hold qualitatively for any sufficiently smooth monotonic decreasing convex utility function of distortion. A user's goal is to maximize his utility, or, equivalently, minimize his total MSE, given his resources, across all TS.

At time t, let $M_{n,t}$ be the available money for user n and p_t be the bit-rate price. Upon entering the system, each user is allocated a specific amount of credit - called 'money' - when he starts sending his video. The initial allocation is based on the (expected) length of time of his video and the (expected) average channel bit-rate (r_n) over this time span, valued at the (normalized or expected) average price ($\bar{p}_{t_n}$). The initial allocation of money for user n is:

$$M_{n,t_n} \;=\; (T_n - t_n + 1).r_n.\bar{p}_{t_n},\ \forall\ n \;=\; 1\ to\ N \tag{5.1}$$

The utility optimization problem for user n over all his TS is given by

$$\max_{\{x_{n,t}\}} \sum_{t=t_n}^{T_n} U_{n,t}(x_{n,t}) \tag{5.2}$$

subject to the constraint

$$\sum_{t=t_n}^{T_n} p_t.x_{n,t} \leq M_{n,t_n} \tag{5.3}$$

The constraint in Eq. 5.3 requires the money spent over all TS to be less than or equal to the total allocated money. Solving Eq. 5.2 under the constraint of Eq. 5.3 gives the optimal demand for each user in all TS. The bit-rate prices (p_t) in Eq. 5.3 are ideally the equilibrium prices which would equate the total demand in each TS with the total supply. However, for a real-time problem, the RD functions for future TS are unknown as are the future prices which will depend on all video users' (unknown) RD functions in the future. To address this informational limitation, we consider a sequential process, as we did in Chapter 4. In each TS, a user will reoptimize his decision for the current and all future TS using expected values for future prices and RD functions. If the future TS are identical in expectation (for example, the future environment is perceived as stationary), then the decision problem in each TS is just an optimization problem with two decisions only - the demand, $x_{n,t}$, for the current TS and $\bar{x}_{n,t}$, the common demand for each of the remaining $(T_n - t)$ TS. Given the price at TS t and an estimated price $\bar{p}_t$ for the future average RD function, the new optimization problem becomes

$$\begin{aligned} \max_{x_{n,t},\bar{x}_{n,t}} \quad & [U_{n,t}(x_{n,t}) \ + \ (T_n - t).\bar{U}_{n,t}(\bar{x}_{n,t})] \\ s.t. \quad & p_t.x_{n,t} \ + \ (T_n - t).\bar{p}_t.\bar{x}_{n,t} \ \leq \ M_{n,t} \end{aligned} \tag{5.4}$$

where $\bar{U}_{n,t}(\bar{x}_{n,t})$ is the future average utility for user n at TS t.

User n at TS t, thus, makes a demand of $x^*_{n,t}$ bits, where $x^*_{n,t}$ is the solution of Eq. 5.4. This information is sent to the allocator. Based on the demand from each user, the allocator makes the decision on the number of bits to be allocated to each user.

5.2.1 Outline of the pricing-based decentralized bit-rate allocation algorithm

The outline of our pricing-based decentralized rate allocation algorithm is given in this section. The details of the algorithm are discussed in the next section.

1. **Initial conditions**: Initially, the allocator announces a bit-rate price (p_1) at TS 1. Without loss of generality, we set $p_1 = 1$. Each user is allocated an initial amount of money as given by Eq. 5.1 at the start of his video.

2. **Bit-rate demand**: Each user knows the bit-rate price (p_t) and their own utility function ($U_{n,t}(x_{n,t})$) for the current TS. Each user also estimates an average future price ($\bar{p}_t$) and future average utility function ($\bar{U}_{n,t}(\bar{x}_{n,t})$). The estimation of future average price is explained in Section 5.3.3, and methods for estimating average utility functions for the remaining future TS are given in Section 5.3.1. Using Eq. 5.4, a user calculates his bit-rate demand, $x^*_{n,t}$, which is then transmitted to the allocator. Details of this calculation are in Section 5.3.2.

3. **Bit-rate and price adjustment**: As the sum of the demands from all the users might not equal the available supply, the allocator needs to normalize the demands to the available supply. Possible approaches are given in Section 5.3.4. The allocator also determines the new bit-rate price for the next TS based on the difference between supply and demand for the current TS. Details of this calculation are also in Section 5.3.4.

4. **Available money**: When the allocation is received by the users, they encode their video at the allocated bit-rate and transmit through the shared channel. The users then recalculate their total available money for the next TS by subtracting the amount of money just spent in the current TS. Details of this calculation can be found in Section 5.3.5.

5. Steps 2-4 are repeated for each TS until all the videos are transmitted.

5.3 Bit-rate allocation for multiple video streams

In this section, we discuss in detail the steps involved in the pricing-based decentralized bit-rate allocation.

5.3.1 Estimating average RD functions for the future

Eq. 5.4 for bit-rate allocation for multiple video streams requires the estimation of average RD characteristics for future TS. We considered several estimation approaches. They differ in the amount of information a user holds at the time of making the forecast. These methods were described in Section 4.3 and are briefly discussed below.

Let $D_{n,t}(r_{n,t})$ denote the MSE distortion at rate $r_{n,t}$. We approximate the RD curve using

$$D_{n,t}(r_{n,t}) \;=\; a_{n,t} \;+\; \frac{b_{n,t}}{r_{n,t}+d_{n,t}} \tag{5.5}$$

where $a_{n,t}$, $b_{n,t}$, and $d_{n,t}$ are curve fitting coefficients for user n at TS t and are determined numerically. This curve-fitting model is widely used for the RD curves of video streams [31, 33, 45]. Other curve-fitting models with reduced complexity can be used, for example [46]. The utility of a user is defined by the negative sum of his MSE at any TS ($U_{n,t}(x_{n,t}) = -D_{n,t}(x_{n,t})$). We examine the following two methods to estimate the future average RD curves:

1. **PRE**: The average future RD curve is estimated by averaging the past RD curves. Specifically, we take the average of the individual curve fitting coefficients (a, b, and d) separately for a user over all past TS (t_n to $t-1$, for user n). This is an *ex post* model where users have no information about the future TS, as in the real-time case. We assume that the video is a stationary process at the GOP level. This means, on average, future TS properties such as complexity can be estimated by looking at the past video. Generally, if averaged over a sufficiently long period of time, the complexity of most video streams can be assumed to be almost stationary. Therefore, after a sufficiently long period of time, the average RD function of past TS will be a good approximation model for the average RD function for future TS. Estimation of the average RD function for the future from the past TS was observed to work well for all the videos used for our simulation.

   We use Eq. 5.4 to calculate the bit-rate demanded by each user in this allocation method. This method would be expected to work well for long videos

but may not work as well for short videos if the previous TS are very different from the future TS.

2. **REM**: This method assumes each user knows the approximate average RD function for his video over the *remaining* TS ($t+1$ to T_n, for user n). This curve is obtained by averaging the individual curve fitting coefficients (a, b, and d) separately for a user over all the remaining TS. Empirically averaging the actual curves and applying a standard curve fitting technique, the estimated coefficients are extremely close to those from averaging the coefficients individually. Since REM requires an average of RD curves for all the remaining TS for a user, it changes with every TS. REM is an *ex ante* approximation model where we assume this average information about the future video is known in advance. This assumption would hold for archival video.

   REM is used in Eq. 5.4 to formulate the bit-rate allocation problem for each user. If the complexity for the current TS is more than the estimated average complexity for the remaining TS, then a user is willing to spend more money than average for the current TS and vice versa if the complexity of the current TS is less than the estimated average complexity for the remaining TS.

We expect REM to perform better than PRE because future video information is used in REM whereas no future information is used in PRE.

5.3.2 Bit-rate demanded by the user

Using Eq. 5.4 and Eq. 5.5, we get the user's per TS decision problem:

$$\max_{x_{n,t},\bar{x}_{n,t}} \quad -(a_{n,t}+\frac{b_{n,t}}{x_{n,t}+d_{n,t}}) \; - \; (T_n-t).(\bar{a}_{n,t}+\frac{\bar{b}_{n,t}}{\bar{x}_{n,t}+\bar{d}_{n,t}})$$
$$s.t. \quad p_t.x_{n,t} \; + \; (T_n-t).\bar{p}_t.\bar{x}_{n,t} \; \leq \; M_{n,t} \;\; \forall\, n \; = \; 1\, to\, N \tag{5.6}$$

where $\bar{a}_{n,t} + \frac{\bar{b}_{n,t}}{\bar{x}_{n,t}+\bar{d}_{n,t}}$ is the predicted average RD function for user n for all future TS ($t+1$ to T_n). At any TS, a user tries to reduce his sum of MSE for the current TS and estimated MSE for all the future TS given the bit-rate price for the current TS and the expected bit-rate price for all future TS.

We solve Eq. 5.6 using a Lagrange multiplier approach [47] for each user separately. All the users calculate their bit-rate demand for the current TS. The bit-rate demand for user n in TS t is given by

$$x^*_{n,t} = \sqrt{\frac{b_{n,t}}{p_t}}.\frac{M_{n,t} + p_t.d_{n,t} + (T_n - t).\bar{p}_t.\bar{d}_{n,t}}{\sqrt{p_t.b_{n,t}} + (T_n - t).\sqrt{\bar{p}_t.\bar{b}_{n,t}}} - d_{n,t}, \quad \forall\, n = 1\, to\, N \tag{5.7}$$

$x^*_{n,t}$ is the only information that is conveyed to the allocator by the user.

5.3.3 Normalization of $\bar{p}_t$

The parameter ($\bar{p}_t$) represents the average price at all future TS beyond current time t. This parameter is required to determine the bit-rate demand for the current TS for each user with respect to the average demand at future TS.

Under our stationarity assumption about the future, *i.e.*, the future RD functions and bit-rate supply are assumed to be distributed identically for every TS, we can reduce the user's T_n TS problem to a sequence of two TS problems, given by Eq. 5.4. Additionally, by Eq. 5.1 and Eq. 5.4, a user's budget is homogeneous of degree zero in prices so that a user's optimal demand in the current TS ($x^*_{n,t}$) and average future demand ($\bar{x}_{n,t}$) depend only on the price ratio, $p_t/\bar{p}_t$. Thus, without loss of generality, we may normalize the future average price $\bar{p}_t$ to unity, $\bar{p}_t = 1$.

5.3.4 Allocation and price adjustment by the allocator

The market clearing price or an equilibrium price is defined as the price at which the total demand is equal to the total supply. Since the price p_t will in general not be the equilibrium price, the sum of bit-rate demanded by all the users may differ from the total available bit-rate at any TS. Excess demand is defined to be the difference between total bit-rate demand and total bit-rate supply (R_t). It will generally be either strictly positive or strictly negative. The allocator has the following options to equalize the total demand and total bit-rate supply:

1. **Normalized demand**: The allocator may normalize the individual demands to balance the total demand and supply:

$$\hat{x}_{n,t} = x^*_{n,t} . \frac{R_t}{\sum_{n=1}^{N} x^*_{n,t}} \tag{5.8}$$

The normalized allocations ($\hat{x}_{n,t}$) are sent back to the users who encode their videos using the allocated bit-rate. The price for the next TS is adjusted by the allocator based on the current excess bit-rate demanded by all the users

$$p_{t+1} = p_t + \alpha_p . (\frac{\sum_{n=1}^{N} x^*_{n,t} - R_t}{R_t}), \; p_1 = 1 \tag{5.9}$$

where the price adjustment parameter, α_p, is a design choice to regulate the price variation, as will be discussed in Section 5.4.1.

If aggregate demands are similar from one TS to the next (e.g., if aggregate demand in the current TS is a good predictor of aggregate demand in the next TS, as would be the case if the video streams followed a random walk), then the price rule that sets the next TS's price by adjusting the current price proportionately to current excess demand can be expected to be a reasonably efficient rule. In fact, video streams are generally characterized by scenes of varying amount of motion with abrupt breaks at scene changes. Within a scene, a random walk assumption is generally reasonable. Thus, if most of the time, most videos are within a scene, then the aggregate demand each TS will be a reasonably good predictor of demand at the next TS, and a price adjustment rule based on excess demand will provide the appropriate signal about the relative scarcity of bit-rate available next TS. As long as only a few videos experience an abrupt break at the same time, current demand will be a good predictor for the demand at the next TS.

2. **Iterative pricing**: The price at each TS could, in principle, be iterated until the market clearing price is achieved. The final price achieved by these iterations would be the equilibrium price for that TS.

The initial price for the first iteration of TS t is taken to be the final price achieved at TS $t-1$ (that is, $p_t^{(1)} = p_{t-1}^{(final)}$). For the iterative pricing at each

TS, the initial price is announced by the allocator and the users calculate their demand which is given by Eq. 5.4. If the total demand is not equal to the total supply, then the allocator adjusts the price using Eq. 5.10 and the new price is sent back to the users to recalculate their demand. This process is iterated until total demand is equal to total supply, at which point the market clearing (or equilibrium) price is achieved.

At the i^{th} iteration, the bit-rate price for the next iteration is set as

$$p_t^{(i+1)} = p_t^{(i)} + \delta_p.(\frac{\sum_{n=1}^{N} x_{n,t}^{*(i)} - R_t}{R_t}), \tag{5.10}$$

where δ_p is the iterative price adjustment parameter and $x_{n,t}^{*(i)}$ is the bit-rate demand by user n at the i^{th} iteration in TS t.

Under standard conditions, the bit-rate price would be expected to converge to the market clearing price. This is the final bit-rate price for TS t and is also used as the initial price for the next TS. At this price, the actual allocation given by the allocator to user n is $\hat{x}_{n,t} = \lim_{i\to\infty} x_{n,t}^{*(i)}$, and normalization is not needed.

Note that the number of iterations depends on δ_p. If δ_p is too large, then it is possible that the price sequence will fluctuate about the market clearing price without ever converging. If δ_p is very small, then convergence to the market clearing price may eventually be achieved but it might take an arbitrarily large number of iterations to converge. Therefore, δ_p would need to be either chosen carefully initially to avoid both extremes or changed adaptively depending on the excess demand. An iterative pricing method was used in [33] that also discusses these convergence issues.

Ideally, in each TS, several iterations of price and demand messages would be exchanged between the allocator and the users, as given by Eq. 5.10. However, in a real-time process, iteration over the price to achieve the competitive equilibrium can become a bottleneck. As we will show later in our simulation, the bit-rate price calculated by the allocator without any iterations follows the competitive equilibrium price closely, and iteration over

price within a TS produces little improvement in the video quality of the users.

3. **Delay buffer**: If iterative pricing is used, each user would be allocated exactly the number of bits demanded. However, if the iterative pricing within a TS is not used (that is, there is only a single exchange of price and demand information between the allocator and the users in each TS), the user will generally not receive the exact bit-rate demanded. This problem can also be solved by using a delay buffer. The total demand may not equal total supply, but the buffer is drained at the rate of total supply. Any extra bit-rate demanded will be stored in the delay buffer which will be drained during those periods when the total demand is less than the total supply. The bit-rate price will still vary as given in Eq. 5.9. In this method, we assume that the delay buffer is arbitrarily large, as needed.

4. **Limited delay buffer and normalized demand**: In any practical scenario, the size of the delay buffer is limited and pre-defined. Therefore, any excess demand in a TS can be accommodated in the delay buffer as long as the delay buffer does not overflow. If, however, the available space in the delay buffer is smaller than the excess bit-rate demanded by the users, then the user's demands are normalized in accordance with the available space in the delay buffer. The demand is not normalized if there is enough space in the delay buffer to accommodate the excess demand.

   Suppose B_t is the available space in the buffer at TS t. In case of demand normalization due to imminent buffer overflow (i.e., $B_t \leq \sum_{n=1}^{N} x^*_{n,t} - R_t$), the normalization is given by

$$\hat{x}_{n,t} = x^*_{n,t} - \frac{x^*_{n,t}}{\sum_{n=1}^{N} x^*_{n,t}}.(\sum_{n=1}^{N} x^*_{n,t} - (R_t + B_t)) = x^*_{n,t}.\frac{(R_t + B_t)}{\sum_{n=1}^{N} x^*_{n,t}} \qquad (5.11)$$

   However, to reduce the chance of buffer overflow and underflow, we add an extra parameter to the price adjustment function to take into account the

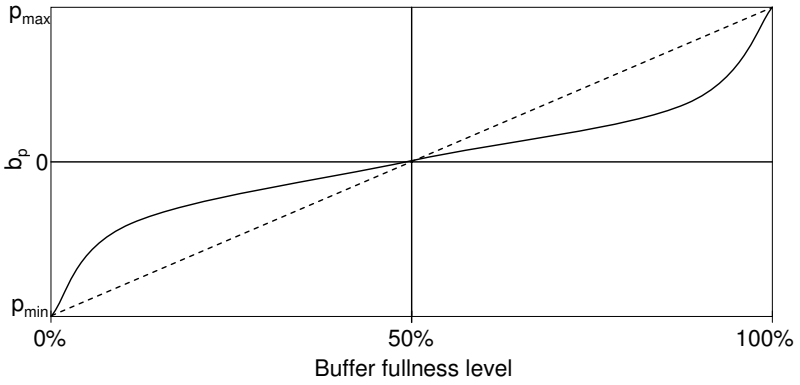


Figure 5.2: Additional price adjustment with the buffer fullness level.

level of buffer fullness. We modify Eq. 5.9 for the limited delay buffer case as

$$p_{t+1} = p_t + \alpha_p.(\frac{\sum_{n=1}^{N} x^*_{n,t} - R_t}{R_t}) + b_p(B_l),\ p_1 = 1 \qquad (5.12)$$

where $b_p(B_l)$ is the buffer level price adjustment factor which is a function of buffer fullness level (B_l).

In Fig. 5.2, we show two examples of varying b_p with B_l. In the first example, b_p varies linearly with B_l, as shown by the dotted line. When the buffer fullness is 50%, then b_p is 0, which means that no adjustment is made to the bit-rate price based on the buffer fullness. When the buffer fullness exceeds 50% then b_p is larger than zero and increased linearly, and when the buffer fullness is less than 50% then b_p is smaller than zero and decreased linearly. The second example shows a monotonic variation of b_p with B_l. Here, if the buffer fullness is around 50% then the price variation is low. As the buffer fullness approaches 100%, b_p grows rapidly to avoid buffer overflow. On the other hand, if the buffer fullness approaches 0%, b_p is reduced rapidly to avoid buffer underflow. In case of buffer overflow, we normalize the demand as given in Eq. 5.11. In case of an empty buffer, we proportionally increase the demand such that the available bit-rate is fully utilized. In both the cases, the idea is to increase the bit-rate price when the total demand is more than the total supply to discourage the users from demanding more

and to decrease the bit-rate price when the total demand is less than the total supply to encourage the users to demand more.

5.3.5 Wealth adjustment by the user

The users send their bit-rate demands ($x^*_{n,t}$) to the allocator. The allocator sends back the actual bit-rate allocation ($\hat{x}_{n,t}$) as discussed previously. Then, users encode their video streams at the allocated bit-rates and transmit over the shared channel. A user is charged for his allocated bit-rate at the current price. Therefore, users reduce their remaining wealth as given by

$$M_{n,t+1} \;=\; M_{n,t} \;-\; p_t.\hat{x}_{n,t} \;\;\forall\; n \;=\; 1 \;to\; N \tag{5.13}$$

where $\hat{x}_{n,t}$ is the actual allocated bit-rate for user n at TS t.

The wealth of a user is reduced at each TS until he transmits all of his video or runs out of money. If a user calculates his optimal demand as given in Eq. 5.7 then the user will always preserve money for the future until all of his video is transmitted.

5.3.6 Sophisticated users

By honest reporting of demand, we mean reporting the $x^*_{n,t}$ which results from Eq. 5.7. A sophisticated user is one who either over or under reports his true demand (Eq. 5.7) in order to affect the price at the next TS to his advantage. An ordinary or normal user is a price taker who bases demand solely on Eq. 5.7 and does not take into account how his current demand may affect the future price. As a general proposition, a sophisticated user could potentially benefit by departing from the honest reporting of his bit-rate demand at any TS. By either exaggerating or understating his demand, he can alter both his allocation and expenditure in the current TS and also influence the price he will face next TS. Thus he could potentially gain more utility at the next TS than he would sacrifice at the current TS by not demanding his $x^*_{n,t}$ bits this TS.

For example, by demanding less than $x^*_{n,t}$ bits in the current TS, a user would receive less bit-rate than otherwise, thus lowering his utility this TS. How-

ever, in addition to preserving more money to spend for bit-rate next TS, the price will be lower than otherwise because the excess demand will be less. Hence, he may be able to acquire more bit-rate next TS. It is not guaranteed that he will experience a net benefit since every other user will also face the lower price and hence demand more as well. Whether such a strategy actually would pay off in higher utility next TS to offset the loss in utility in the current TS would depend crucially on the demands of other users. If the aggregate demand of the other users next TS is highly price sensitive, then lowering demand this period to lower bit-rate price next TS would be less likely to pay off than if the other users' aggregate demand next period is less price sensitive.

Conversely, a sophisticated user might be able to benefit by overstating his demand in the current TS if the increase in price next TS causes other users to demand sufficiently less bit-rate, so that the sophisticated user will not suffer as big a loss in bit-rate next TS as he would have expected by demanding more currently.

Although there exists, in general, some deviation from reporting a user's demand in any TS, given by Eq. 5.7, that would benefit him, knowing even the direction of the deviation (that is, should the user demand more or less) requires him to know more about the other users aggregate demand than may be plausible to assume. Additionally, under our model assumptions and specifications, it is a standard economic theory result [51] that the potential utility increase which any sophisticated user would be able to achieve becomes vanishingly small as the total number of users increases. That is, as the number of other users increases, a sophisticated user would be less able to manipulate the next period's price and hence would have less advantage possible from doing so. For this chapter we have followed in the tradition of competitive analysis and assumed all users ignore any potential influence that their current TS demand will have on bit-rate price at the next TS. As included in our results, we did verify that users who try simple deviations in demand from Eq. 5.7 invariably did worse in overall quality compared to when they followed Eq. 5.7.

5.4 Results

Our simulation results were generated using H.264/AVC [37] reference software JM 11.0 [38] with the baseline profile. The test videos were taken from a 72 minute travel documentary. The frame rate of each video is 30 frames per second. Each test video is 250 seconds long (total 7500 frames) at a resolution of 352×240 pixels (SIF). We chose 12 such test streams (denoted g1 to g12, the same videos as used in Chapter 4). The GOP (TS) size is 15 frames (I-P-P-P) and the frames in a GOP are encoded using H.264 rate control [41]. The coding parameters such as length of video, resolution, GOP size or structure can be changed according to the requirement of an application as our multiplexing method is independent of such parameters. The decentralized rate allocation method for multiple video streams can be used for any GOP size or structure, frame rate, video length or resolution. We considered a lossless channel for transmitting multiple video streams. The quality of a video is reported in terms of average Peak Signal-to-Noise Ratio (PSNR). We first calculate the frame level MSE for any video stream. Then the MSE is averaged across all frames of a video and finally converted to PSNR for reporting the results.

An upper bound on the video quality can be approximated by using the exact RD function for all the users at all TS. Suppose each user is endowed with some initial wealth. The users are assumed to allocate their wealth at each TS depending on the video complexity. We call this method **FULL**. This method can only be used for archival videos where the RD coefficients are calculated off-line. Each user uses this bit-rate allocation criterion among his TS independently. The bit-rate at each TS is adjusted like other methods, as discussed in Section 5.3.4. Note that we assume a constant price at all TS. Therefore, it is not a real upper bound. The real upper bound can be calculated by solving Eq. 5.2 and Eq. 5.3 (for all TS and for all users simultaneously) and computing the market clearing price such that the total demand is equal to the total supply, which is an extremely large computational problem, and the complexity grows with the numbers of users and TS.

In this chapter, we compare our multiplexing methods using the pricing-

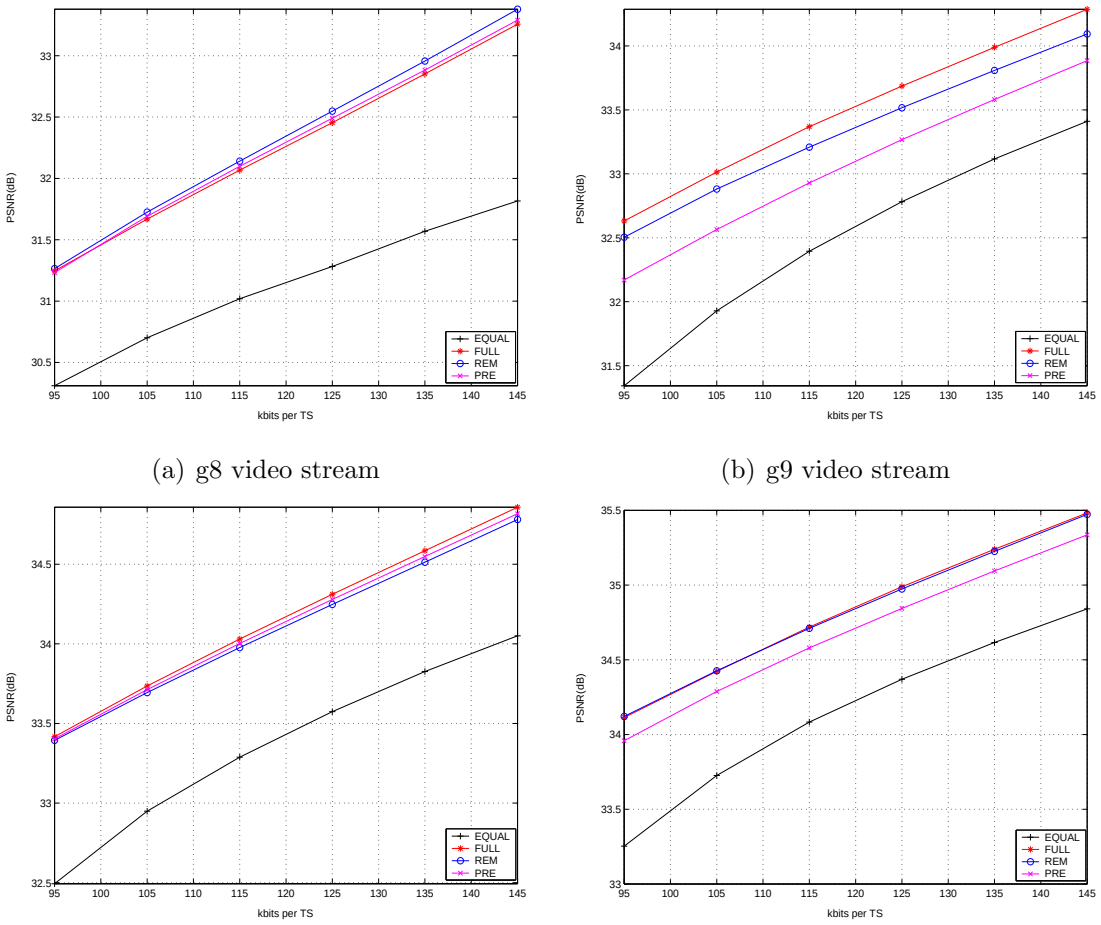
based decentralized bit-rate allocation to the constant rate allocation, **EQUAL**. Here, each TS in a video receives an equal number of bits to encode that segment of video. Note that the rate control algorithms, such as [41], used in conjunction with most current video standards strive to achieve equal rate allocation for all GOPs, similar to EQUAL when a TS is of GOP length. This seems to be an appropriate comparison and is analogous to current multiplexing practices [25,28,31,33] where the results are compared with the equal bit-rate allocation to all users. In addition, we also compare our method with the MINAVE method described in [28] which improves the average video quality by allocating the bit-rate to videos based on their relative complexity.

5.4.1 Constant rate and number of users at all TS

We start with the scenario of bit-rate allocation for multiple video streams where we consider a Constant Bit-Rate (CBR) Channel and all the users are present at all TS. All the videos start at the first TS, and last for 500 TS. There is no delay buffer, and the bit-rate demanded by the users is normalized to equal the total available supply at each TS as given in Eq. 5.8. The price adjustment parameter, α_p, was kept at 0.1. The price fluctuation increases if α_p is large, resulting in a large fluctuation in the bit-rate demanded by the users. The price adjustment can not track the excess demand properly if α_p is very small. For our simulations, α_p was not optimized for any set of video streams and thus it might be possible to improve the performance of these multiplexing methods by carefully tuning this parameter, depending on what is known about the videos being transmitted.

The PSNR results for four video streams when multiplexed together using our pricing-based decentralized bit-rate allocation mechanism are shown in Fig. 5.3. On the x-axis is the operating bit-rate (kbits per TS) and on the y-axis is the video quality given by PSNR (dB). There are four curves in each plot, one for each bit-rate allocation method. EQUAL is our baseline case where each video stream receives an equal share of available bits at each TS. From the figure, we see that all the other bit-rate allocation methods outperform EQUAL for all videos.

In the FULL method, each user knows his RD characteristics for all the



(a) g8 video stream

(b) g9 video stream

(c) g10 video stream

(d) g11 video stream

Figure 5.3: PSNR performance versus bit-rate for four multiplexed video streams. All the video streams exist at all TS and the bit-rate demand is normalized by the total available supply.

TS in advance. Depending on the video complexity at any TS, a user will make a bit-rate demand at that TS. The demands are normalized based on the total available bit-rate at that TS. We see that FULL almost always performs the best among all methods. The improvement of FULL over EQUAL varies from 0.62-0.86 dB for g11 to 0.94-1.44 dB for g8. This quality improvement is attributed to not only the varying number of bits at each TS but also the advance knowledge of the video characteristics.

Instead of precise knowledge of RD characteristics at each TS, suppose the users only know the average RD characteristics for the remaining TS. Then the REM bit-rate allocation method is shown to provide video quality improvement for all the videos individually. Here, each user calculates his bit-rate demand at any TS compared to his average demand for the remaining TS. If the video is more complex in the current TS compared to the expected average video complexity for the remaining TS, then the user will spend more wealth for the current TS compared to the average amount of wealth spent in the remaining TS. On the other hand, if the video is less complex than the average complexity for his video, then he will spend less than his average, and save his wealth for the TS when he might need to spend more. The results in Fig. 5.3 show that the quality of all the videos is improved individually. The video quality improvement for all the users with REM varies from 0.60-0.87 dB for g11 to 0.95-1.56 dB for g8. All the video streams benefit from this multiplexing method and the amount of PSNR improvement depends on the video characteristics. Generally, the PSNR improvement is more for the videos whose complexity varies substantially over time.

In real-time video transmission scenarios with no input buffering of raw frames other than the current GOP, generally the users have no knowledge of their future video. The users know only their video at the current TS and all the past TS. We use PRE bit-rate allocation where a user calculates his demand for the current TS compared to the estimated average demand for the future TS, where the average demand for the future TS is estimated as being equal to the average demand for all the past TS. With no knowledge of how the video characteristics

might look in the future, PRE bit-rate allocation still improves the quality of all the videos individually. The quality improvement over EQUAL varies from 0.46-0.83 dB for g9 to 0.92-1.47 dB for g8. Note that this method performs worse than FULL and REM because of the lack of any knowledge about the video characteristics for future TS. However, PRE is realistic for real-time applications.

In general, the pricing-based decentralized rate allocation method for multiple video streams improves the video quality for all the videos. The MINAVE method [28] has a different objective: to reduce the *average* MSE across all the users. For example, MINAVE applied to time-domain RD curve produces average quality of 32.91 dB for the four video streams used in Fig. 5.3 while the average PSNR values for REM and PRE are 32.68 dB and 32.56 dB. The MINAVE method does better in minimizing the *average* MSE for all the users, however one of the users experiences worse PSNR compared to EQUAL, whereas the other users experience better PSNR. In contrast, with our method, all four users are better off compared to EQUAL.

5.4.2 Comparison with centralized bit-rate allocation

Video quality improvement for six video streams when multiplexed together using price-based bit-rate allocation is shown in Fig. 5.4. These results are similar to the previous case of multiplexing using four video streams. However, due to the increase in the number of video streams, the effect of normalization is less than that with four videos. Therefore, the bit-rate allocation to each user is closer to their actual demand. For FULL, the improvement for g8 is from 1.23-1.91 dB, much higher than 0.94-1.44 dB for the same video in the case of multiplexing four video streams. Similarly, REM improves the video quality from 0.62-0.92 dB for g11 to 1.21-1.98 dB for g8, and PRE improves the video quality from 0.50-0.77 dB for g11 to 1.07-1.74 dB for g8. Using the price-based decentralized bit-rate allocation, all the video streams improve their video quality compared to EQUAL allocation. The video quality improvement for a video stream increases with the increase in the number of videos multiplexed together.

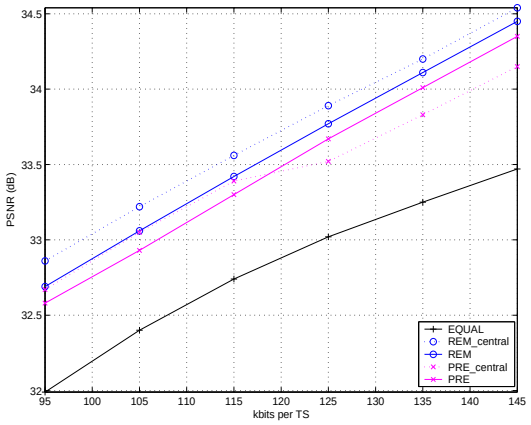
Fig. 5.4 also compares the results between price-based decentralized bit-rate

allocation and centralized competitive equilibrium bit-rate allocation [61]. Generally, centralized bit-rate allocation performs better than decentralized bit-rate allocation because more information is used. In the centralized bit-rate allocation, all the users send their RD functions, and a central allocator computes the appropriate bit-rate allocation for all the users simultaneously. In the decentralized allocation, the allocator has no information about the RD functions of the users; only the bit-rate demand is conveyed to the allocator. However, as can be seen from Fig. 5.4, the improvement from using centralized allocation (REM_central and PRE_central) is only around 0.2-0.3 dB over price-based decentralized allocation for g7, g11, and g12 videos. The performances for the two methods are comparable for g9 and g10 videos. The decentralized allocation performs slightly better than the centralized allocation for g8.

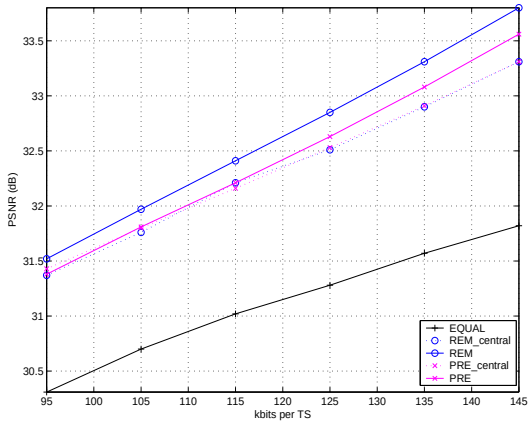
In general, the centralized bit-rate allocation performs slightly better than the decentralized allocation. However, our price-based decentralized bit-rate allocation method has the advantage of reducing the amount of private information shared by the users and removing the huge computational burden imposed on the allocator in the centralized approach. The computational complexity grows exponentially with the number of users in the centralized allocation. In the proposed method, the computational complexity is very small. The computation performed by the allocator is a simple normalization. The calculation performed by each of the users is independent of the number of users.

5.4.3 Effect of delay buffer

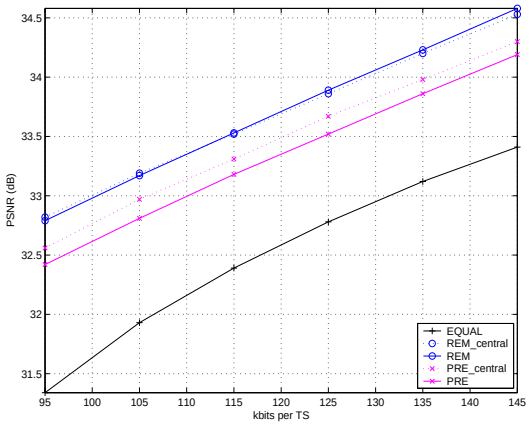
In deriving the previous results, there was no delay buffer to store the demanded bit-rate that is in excess of total available bit-rate at any TS. However, if we have a delay buffer to store the extra bits which can be transmitted at a later TS, then performance can be improved dramatically. Fig. 5.5 shows the effect of a delay buffer on two out of the six video streams that are multiplexed together in Fig. 5.4. On the x-axis is the size of the delay buffer (in terms of average bit-rate per TS per user) and on the y-axis is the PSNR improvement over EQUAL. The videos have been multiplexed together at an average bit-rate of 100 kbits



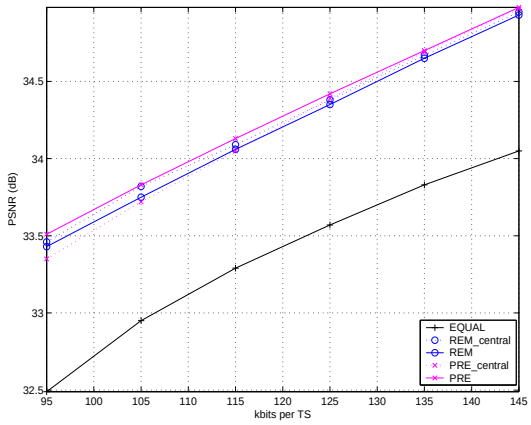
(a) g7 video stream



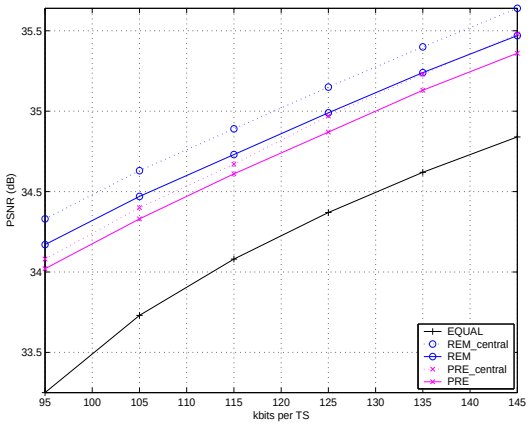
(b) g8 video stream



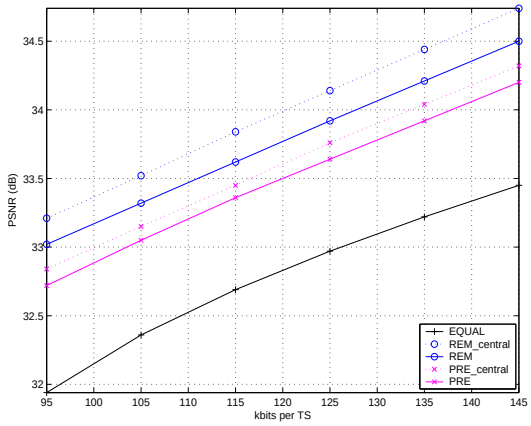
(c) g9 video stream



(d) g10 video stream



(e) g11 video stream



(f) g12 video stream

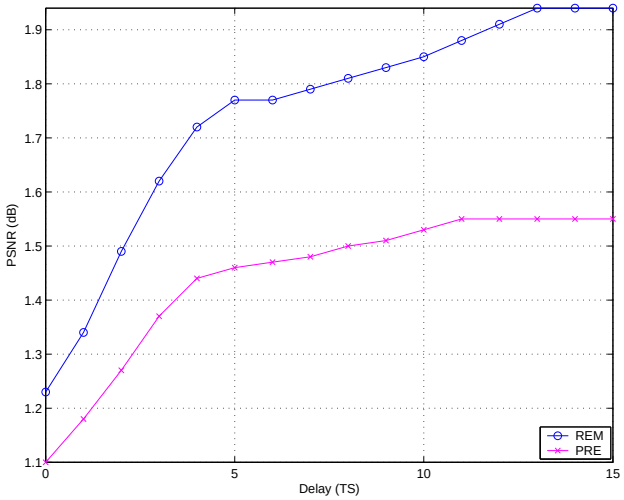
Figure 5.4: PSNR performance versus bit-rate for six multiplexed video streams: Comparison of the proposed method with the centralized bit-rate allocation method.

per TS per user. The two curves in each plot show the result for the price-based decentralized bit-rate allocation methods. A delay buffer of 0 TS represents no delay buffer; all demands are normalized by the total available bit-rate at each TS. The video quality improves as the size of the delay buffer increases. At any TS, buffer overflow is prevented by normalizing the demand when the buffer is full, as given in Section 5.2. For g8, the video quality improves from 1.10 dB at no delay to 1.55 dB for delay buffer size of 1100 kbits per TS for PRE and from 1.23 dB at no delay buffer to 1.94 dB at a delay buffer size of 1300 kbits per TS for REM. The video quality improvement saturates at some delay buffer size because at such high delay all the demands are always accommodated in the buffer and there is no need for normalization. After that, any further increase in the delay buffer size does not increase quality.

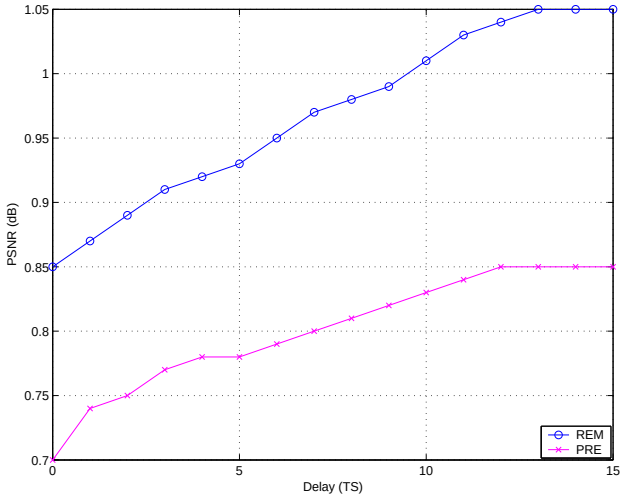
5.4.4 Iterative pricing

Although our method for allocating bit-rate is based on classical iterative price-guided procedures, we truncate the iterations before convergence to an equilibrium (between demand and supply) is reached. In fact, we only allow one round of price and demand information to be exchanged prior to allocating bit-rate to the users. However the price that is sent to the users is not an arbitrary or "average" price; it is an adjusted price from the previous TS where the adjustment is proportional to the excess demand reported in the previous TS. Whether or not this is a fortuitous or clever choice of the price to announce depends on how similar the excess demand (at any price) in the current TS is to the excess demand in the previous TS. Fortunately, as an empirical observation, it seems that for the various collections of videos used in our simulations, complexity levels in successive GOPs are correlated, and so, successive aggregate excess demand functions are quite similar. Thus, even though the current TS price is adjusted from the previous TS price proportionately to that TS excess demand, in general the price is adjusted in the right direction and over time the sequence of adjusted prices tracks the sequence of equilibrium prices quite closely.

For multiplexing four video streams, Fig. 5.6 compares the case when the



(a) g8 video stream



(b) g11 video stream

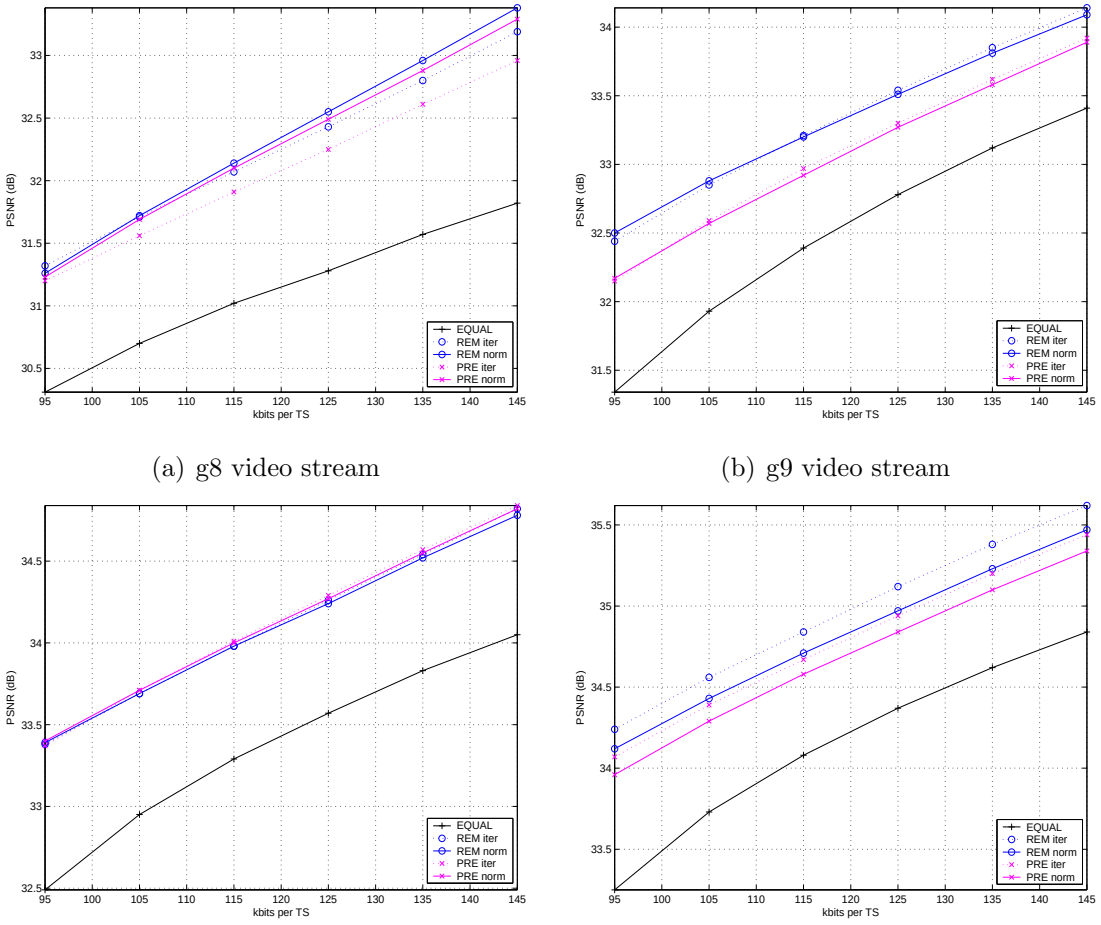
Figure 5.5: PSNR improvement with delay buffer size for two video streams from six multiplexed video streams at 100 kbits per TS streams.

bit-rate price is iterated, against the case when the price is broadcast only once. REM and PRE represent the normalized demand case when the price is announced only once; these are the same as Fig. 5.3. 'REM iter' and 'PRE iter' represent the video quality achieved by iterating the price to obtain the equilibrium price for the REM and PRE methods. On the x-axis is the bit-rate and on the y-axis is the video quality achieved using these methods. The solid curves represent the video quality achieved through normalization and the dotted curves represent the video quality achieved by obtaining the equilibrium prices through price iteration. We find that, for g9 and g11, the iterative pricing produces marginally better video quality than the normalization procedure, and the trends are opposite for g8, while there is negligible difference between these two procedures for g10. In general, both methods for calculating the prices produce almost the same video quality. However, iterating over price to achieve the equilibrium involves sending messages back and forth between the users and the allocator. This is a time consuming process and may not be suited for time critical applications.

5.4.5 Sophisticated users

In our discussion so far, we have assumed that users do not attempt to deviate from their true demand. All the users are concerned for the current TS with respect to the price adjustment process irrespective of what other users are demanding. As we have argued earlier, it is not possible for a user to influence the prices without having the knowledge of the video characteristics in the future for all the users. Since the users are unaware of the videos of other users as well as their own video in the future, we skip the case of users changing the bit-rate prices for their own benefit.

However, a user may just act selfishly and may demand more every TS. Table 5.1 shows the PSNR for the case when (a) all the users honestly report their bit-rate demand, (b) g9 user exaggerate his demand, and (c) g10 user exaggerate his demand. The PSNR values are generated when these four videos are multiplexed together at 100 kbits per TS per user. For this simulation, we assume that the sophisticated user will inflate their demand by 20% in each TS in order to extract



(a) g8 video stream

(b) g9 video stream

(c) g10 video stream

(d) g11 video stream

Figure 5.6: PSNR performance versus bit-rate for four multiplexed video streams for comparing the effect of iterative pricing ($\delta_p = 0.05$) and demand normalization.

more bit-rate from the allocator.

Table 5.1: PSNR performance (dB) of users when four video streams are multiplexed together at 100 kbits per TS per user and when some users deviate from their true demand.

	g8 video user	g9 video user	g10 video user	g11 video user
All users make true demand				
REM	31.50	32.70	33.55	34.27
PRE	31.47	32.37	33.56	34.13
g9 user deviate from true demand (+20% each TS)				
REM	31.51	32.52	33.56	34.24
PRE	31.50	32.12	33.60	34.11
g10 video deviate from true demand (+20% each TS)				
REM	31.50	32.71	33.40	34.27
PRE	31.53	32.43	33.23	34.11
g9 user deviate from true demand (+20%/-20% at alternate TS)				
REM	31.49	32.55	33.54	34.27
PRE	31.45	32.27	33.55	34.13
g10 user deviate from true demand (+20%/-20% at alternate TS)				
REM	31.50	32.69	33.45	34.27
PRE	31.46	32.36	33.47	34.13

From Table 5.1, it can be observe that when a user deviates from the bit-rate demand given in Eq. 5.4 and inflates demand, then he is the one who gets hurt in the process. When the g9 user changes his demand, his PSNR drops by approximately 0.20-0.25 dB. Similarly, when the g10 video user changes his demand, his video quality drops by 0.15-0.25 dB. It can be observed that when one user deviate from his bit-rate demand, other users are still able to maintain their video quality. When a user inflates his demand, that user eventually ends up asking for less with time because his wealth reduces at a faster level. When his demand decreases, other users end up getting more bit-rate at lower price. So, in general, users will not inflate their demand, and will match their demands correctly to complexity based on Eq. 5.4.

In another case of users trying to change the demand for their profit, suppose a user inflates and deflates his demand at alternate TS. From the Table 5.1, it can be seen that such users still suffers from changing their bit-rate demand while the other users are mildly affected by his actions. These are bad strategy for users

since they are changing the demand without having control on the bit-rate prices and thus end up hurting themselves.

5.4.6 Constant bit-rate and variable number of users

To this point in this chapter, we have assumed that all users start sending their video at the same time, and all users have the same video length. However, in any practical scenario, users may start sending their videos at different times, and the video lengths may not be the same. Fig. 5.7 shows an example profile of eight video users with different start times and video lengths. The start times for users are randomly chosen uniformly between TS 0 and 200, and the video lengths are also randomly chosen uniformly between 300 and 500 TS.

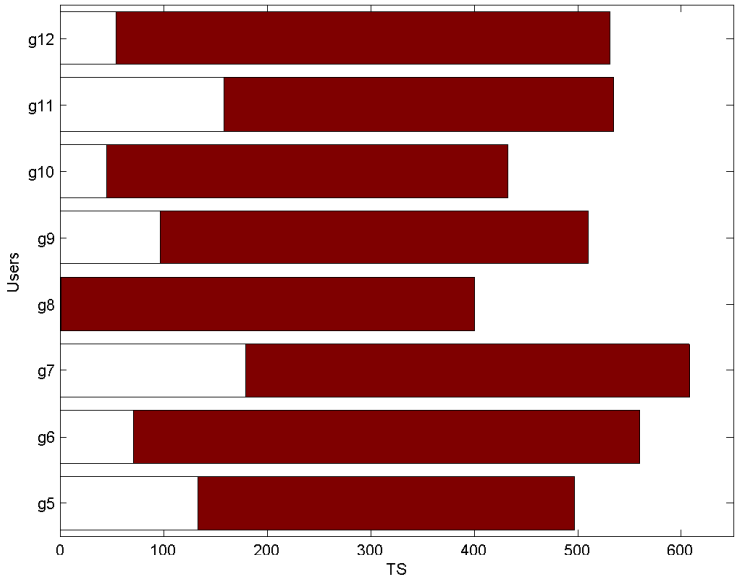


Figure 5.7: Start and end times of 8 video users.

We simulated our price-based decentralized bit-rate allocation method for the videos with the user profile given in Fig. 5.7. The video quality improvement for these video streams is given in Fig. 5.8 for the CBR channel (independent of the number of users at any TS). Again, all the video streams benefit from our multiplexing method. In general, quality improvement increases with the number of users who are participating in the multiplexing process, as shown previously.

However, the video quality improvement also depends on the time when the users overlap with each other.

In a system with a large number of users, it may be assumed that there is nearly a constant number of users at any TS. In such systems, users may enter or leave the system at any time. So, changes in the number of users at any TS with respect to the total number of users will make little difference to other users. With different start times and video lengths, we have shown that all the users still benefit from the multiplexing process. As the number of users increases, the system behaves more like the case of a constant number of users at all TS.

5.5 Conclusion

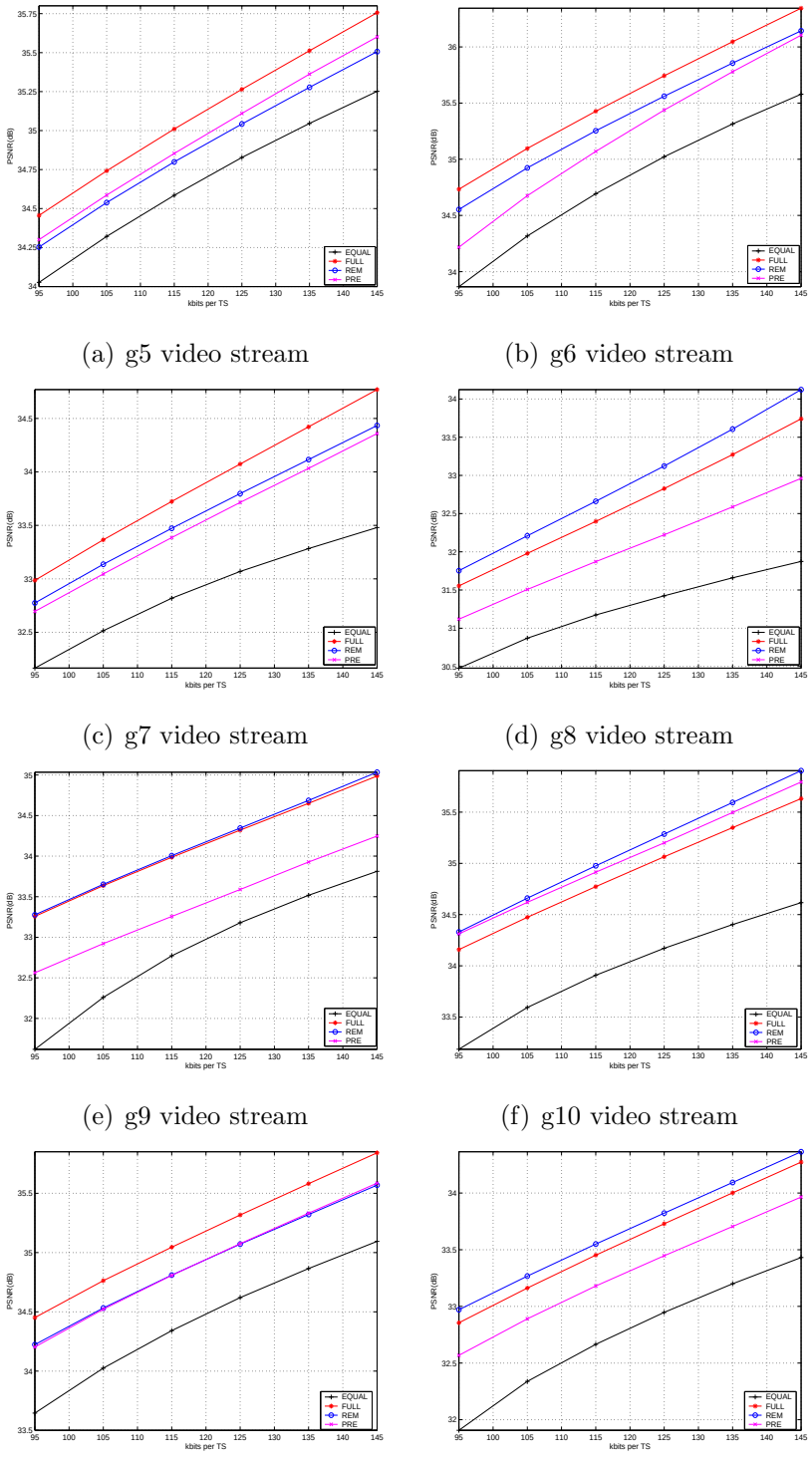
We have demonstrated various methods of price-based decentralized bit-rate allocation among multiple video streams. A video user independently calculates his bit-rate demand for the current TS based on current price, available money, and relative video complexity for the current TS compared to the estimated average complexity for future TS. The bit-rate demanded by each user is sent to the allocator who normalizes the total demand and sends the bit-rate price for the next TS based on the total demand and total available bit-rate.

We examined several variations:

1. We considered the case of an output delay buffer where, instead of normalizing the demands at every TS, the allocator stores the excess demand in the buffer and strives to allocate the actual bit-rate demanded by each user. This further improves the video quality since the bit-rate demand is met at almost all the time-slots.

2. We showed that the performance of our method using a single iteration for bit-rate price determination performs close to the case when the equilibrium price is achieved iteratively.

3. We examined the case where users start at different times and have different video lengths. All users improve their video quality by making bit-rate



(a) g5 video stream

(b) g6 video stream

(c) g7 video stream

(d) g8 video stream

(e) g9 video stream

(f) g10 video stream

(g) g11 video stream

(h) g12 video stream

Figure 5.8: PSNR performance versus bit-rate for eight multiplexed video streams with different start and end times.

demands based on the bit-rate price; the trends are consistent with the case where all users start and end at the same time.

In comparison with the existing multiplexing methods in the literature [25, 27, 28, 30], our price-based decentralized bit-rate allocation method improves the video quality of each user individually whereas the previous methods, focussing on improving the average quality only, caused some users to improve at the expense of others.

In comparison to the centralized bit-rate allocation [61], the proposed method makes the following contributions:

1. The burden of RD information exchange in the centralized bit-rate allocation has been reduced to transmitting only the bit-rate demand by the users.

2. The computational burden that appears in centralized allocation increases exponentially with the number of users. In our proposed method, the computational burden is small and is shifted to individual users and it is independent of the number of users. Yet we show that our proposed algorithm achieves similar video quality improvement.

5.6 Acknowledgements

Chapter 5 of this dissertation contains material that appears in M. Tiwari, T. Groves, and P. Cosman, "Bit-rate allocation for multiple video streams using a pricing-based mechanism", *IEEE Transactions on Image Processing*, in review since February 2010. I was the primary author and the co-authors Dr. Pamela Cosman and Dr. Theodore Groves directed and supervised the research which forms the basis for Chapter 5.

Chapter 6

Conclusions

In this dissertation, we first investigated several improvements in dual-frame video coding to improve the compression efficiency. We also studied three different ways for bit-rate allocation among multiple video streams. In the first approach, our aim was to improve the average video quality of all the video streams using dual-frame video coding. In the other two approaches, the goal was to improve the video quality of all the users by joint bit-rate allocation compared to the video quality when encoded independently.

In Chapter 2, we used simulated annealing to find the best location for LTR frames in a video sequence for dual-frame video coding and discussed rate control for dual-frame video coding with high quality LTR frames.

1. We proposed simulated annealing to find good locations for high quality long-term reference frames. This improved the quality of the entire video stream compared to evenly spaced high quality LTR frames.

2. The process of LTR frame selection was further performed on a constrained lookahead window size in a long video sequence for real-time video transmission. Compared to the case where the entire video sequence was considered, this reduced delay and computational complexity while the PSNR improvement was comparable to the improvement when the entire video sequence was considered.

3. The number of bits, and therefore the quality level, to be assigned to an LTR frame can be determined using a simple video activity measure, and this performs better than the previous work which allocated a fixed number of bits to the high-quality LTR frames.

4. High-quality LTR frames require more bits on average than regular frames, and the standard approaches for buffer-constrained rate control are not designed for this. We designed a rate control algorithm that uses a tighter target buffer level for most frames, and a less restrictive buffer fullness threshold at and after an LTR frame, and this approach outperformed previous methods.

In Chapter 3, we proposed and compared various methods for allocating bit-rate for multiple video streams using dual-frame coding. We considered the three scenarios of (a) no delay constraint, (b) separate encoder output buffer constraints, and (c) a joint delay buffer for all the multiplexed video streams.

1. Separate from the multiplexing problem, by using the activity measurement algorithm to detect when the LTR frame is becoming obsolete, we developed a simple algorithm for adaptive selection of LTR frame location to improve the performance of dual-frame video coding.

2. We combined the equal slope approach and dual-frame coding with high-quality LTR frames, in which the LTR frames are allocated bits based on motion activity, and other frames are allocated bits using the equal slope technique.

3. The buffer-constrained rate control method which works well for individual video streams was modified for a multiplexing scenario, in that the LTR frames in various streams can be slightly delayed or advanced in their locations to avoid having them occur at the same time, thereby overflowing the buffer.

In Chapter 4, we discussed competitive equilibrium bit-rate allocation for multiple video streams.

1. The final bit-rate allocation is a Pareto optimal solution and all the users do at least as well as they would with an individual allocation.

2. We proposed three different methods for estimating the future RD information for a video stream. The users trade bits for the current TS with respect to the estimated bit-rate requirement in the future.

3. We achieved video quality improvement for all the video streams using competitive equilibrium bit-rate allocation compared to the allocation when the video streams are allocated equal bit-rate.

In Chapter 5, we studied pricing-based decentralized bit-rate allocation for multiple video streams.

1. The burden of RD information exchange in the centralized bit-rate allocation was reduced to transmitting only the bit-rate demand by the users. The small computational burden in this method is shifted to individual users and it is independent of the number of users.

2. We considered the case of an output delay buffer where, instead of normalizing the demands at every TS, the allocator stores the excess demand in the buffer and strives to allocate the actual bit-rate demanded by each user.

3. We showed that the performance of our method using a single iteration for bit-rate price determination performs close to the case when the equilibrium price is achieved iteratively.

4. We examined the case where users start at different times and have different video lengths. All users improve their video quality by making bit-rate demands based on the bit-rate price.

6.1 Future work

There are various avenues for future work in dual-frame video coding and multiplexing. One could use the motion vectors which are computed anyway as

part of the encoding process to determine the location and quality level of the LTR frames. With a sufficient number of future frames in the buffer, a dynamic programming solution or a greedy approach can be used to obtain the LTR frame location. The performance of such methods may depend on the lookahead buffer size. In this dissertation, we assumed a lossless channel. Another avenue for future work involves cross-layer design for transmission of multiple video streams on a wireless channel. In such a case, the decision for the LTR frame location and quality may also depend on the channel conditions.

Generation of RD curves involves huge computational complexity. Further research needs to be done to reduce this complexity in order to use such multiplexing methods efficiently.

By classifying video streams in various categories based on their long term averages for estimating the RD properties for future TS, we may further improve the quality of videos by these multiplexing process, compared to the averaging method where we use RD curves of only the past TS.

In our bit-rate allocation for multiple video streams, the goal was to reduce the video distortion. The economic model used for resource allocation in our work is largely disassociated with the video quality. At a high quality, any improvement in video quality is not perceptible subjectively. Therefore, one would not like to waste his resources on achieving video quality beyond such high quality. Similarly, one would not like to use his resources to obtain a low video quality that is perceptually very bad. Between these two quality thresholds, each user values the video quality differently. Therefore, it is a very challenging task to obtain a utility function for video quality measurement. Further work needs to done in defining a transfer function that quantifies the marginal video quality improvement in terms of cost (money, or any other numeraire commodity).

In previous work, the quality of video streams was improved by trading bits among the videos and across time. However, in wireless channels, each channel behaves differently for each video user. The channel state information for each user depends on the user's location, time of the day, distance from the tower, power level, fading, multipath, and many other parameters. Moreover, for some networks,

such as cognitive radio, there is a huge fluctuation in the amount of bandwidth available to secondary users. Therefore, it is not straightforward to allocate the resources on the basis of just the video characteristics. The dimensionality of the problem increases with the addition of the channel condition. More game-theoretic models can be explored for allocating resources to multiple video users in such wireless environments.

Bibliography

[1] T. Sikora, "The MPEG-4 video standard verification model," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 1, pp. 19–31, Feb. 1997.

[2] T. Wiegand, "Joint final committee draft for joint video specification H.264," Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-D157, July 2002.

[3] I. Richardson, "H.264 and MPEG-4 video compression," *John Wiley and sons ltd.*, 2003.

[4] M. Gothe and J. Vaisey, "Improving motion compensation using multiple temporal frames," *Proceedings IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, vol. 1, pp. 157–160, May 1993.

[5] M. Budagavi and J. D. Gibson, "Multiframe block motion compensated video coding for wireless channel," *Proc. IEEE Asilomar conference on Signals, Systems and Computers*, pp. 953–957, Nov. 1996.

[6] ——, "Multiframe video coding for improved performance over wireless channels," *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 252–265, Feb. 2001.

[7] T. Wiegand, N. Färber, K. Stuhlmüller, and B. Girod, "Long-term memory motion-compensated prediction for robust video transmission," in *Proceedings IEEE International Conference on Image Processing*, vol. 2, 2000, pp. 152–155.

[8] T. Fukuhara, K. Asai, and T. Murakami, "Very low bit-rate video coding with block partitioning and adaptive selection of two time-differential frame memories," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 1, pp. 212–220, Feb. 1997.

[9] Y. Huang, B. Hsieh, T. Wang, S. Chien, S. Ma, C. Shen, and L. Chen, "Analysis and reduction of reference frames for motion estimation in MPEG-

4 AVC/JVT/H.264," *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 3, pp. 145–148, apr 2003.

[10] Y. Su and M. Sun, "Fast multiple reference frame motion estimation for H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 3, pp. 447–452, mar 2006.

[11] A. Leontaris, P. Cosman, and A. Tourapis, "Multiple reference motion compensation: a tutorial introduction and survey," *Foundations and Trends in Signal Processing*, vol. 2, no. 4, pp. 247–364, 2009.

[12] T. Wiegand, X. Zhang, and B. Girod, "Long-term memory motion-compensated prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 1, pp. 70–84, Feb. 1999.

[13] A. Leontaris and P. C. Cosman, "Video compression with intra/inter mode switching and a dual-frame buffer," in *Proc. IEEE Data Compression Conference*, Mar. 2003, pp. 63–72.

[14] V. Chellappa, P. Cosman, and G. Voelker, "Error concealment for dual-frame video coding with uneven quality assignment," *Proc. IEEE Data Compression Conference*, pp. 319–328, Mar. 2005.

[15] ——, "Dual-frame motion compensation with uneven quality assignment," *Proc. IEEE DCC*, pp. 262–271, Mar. 2004.

[16] ——, "Dual-frame motion compensation for a rate switching network," in *Proc. IEEE 37th Asilomar Conference on Signals, Systems, and Computers*, Nov. 2003, pp. 1539–1543.

[17] A. Leontaris and P. C. Cosman, "Compression efficiency and delay tradeoffs for hierarchical B-pictures and pulsed-quality frames," *IEEE Transactions on Image Processing*, vol. 16, no. 7, pp. 1726–1740, July 2007.

[18] A. Leontaris and P. Cosman, "Dual-frame video encoding with feedback," in *Proc. IEEE 37th Asilomar Conference on Signals, Systems, and Computers*, Nov. 2003, pp. 1514–1518.

[19] ——, "Video compression for lossy packet networks with mode switching and a dual-frame buffer," *IEEE Transactions on Image Processing*, vol. 13, no. 7, pp. 885–897, July 2004.

[20] M. Tiwari and P. Cosman, "Dual-frame video coding with pulsed quality and a lookahead buffer," *Proc. IEEE Data Compression Conference*, pp. 372–381, Mar. 2006.

[21] Z. Chen and K. Ngan, "Recent advances in rate control for video coding," *Signal processing: Image communication*, vol. 22, no. 1, pp. 19–38, Jan. 2007.

[22] X. Zhu, E. Setton, and B. Girod, "Rate allocation for multi-camera surveillance over an ad hoc wireless network," in *Proceedings Picture Coding Symposium (PCS)*, Dec. 2004.

[23] X. Zhu and B. Girod, "Distributed rate allocation for multi-stream video transmission over ad-hoc networks," in *Proceedings IEEE International Conference on Image Processing*, Sept. 2005, pp. 157–160.

[24] J. Kammin and M. Sakurai, "Video multiplexing for the MPEG-2 VBR encoder using a deterministic method," in *Proceedings IEEE International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution*, 2006, pp. 221–228.

[25] L. Wang and A. Vincent, "Bit allocation and constraints for joint coding of multiple video programs," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 6, pp. 949–959, Sept. 1999.

[26] J. Yang, X. Fang, and H. Xiong, "A joint rate control scheme for H.264 encoding of multiple video sequences," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 2, pp. 617– 623, May 2005.

[27] G. Su and M. Wu, "Efficient bandwidth resource allocation for low-delay multiuser video streaming," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 9, pp. 1124–1137, Sept. 2005.

[28] M. Tagliasacchi, G. Valenzise, and S. Tubaro, "Minimum variance optimal rate allocation for multiplexed H.264/AVC bitstreams," *IEEE Transactions on Image Processing*, vol. 17, no. 7, pp. 1129–1143, July 2008.

[29] L. Boroczky, A. Ngai, and E. Westermann, "Statistical multiplexing using MPEG-2 video encoders," *IBM J. Res. Develop.*, vol. 43, no. 4, pp. 511–520, July 1999.

[30] A. Fattahi, F. Fu, M. van der Schaar, and F. Paganini, "Mechanism-based resource allocation for multimedia transmission over spectrum agile wireless networks," *IEEE Journal on Selected Areas in Commnications*, vol. 25, no. 3, pp. 601–612, Apr. 2007.

[31] H. Park and M. van der Schaar, "Bargaining strategies for networked multimedia resource management," *IEEE Transactions on Signal Processing*, vol. 55, no. 7, pp. 3496–3511, July 2007.

[32] F. Fu and M. van der Schaar, "Noncollaborative resource management for wireless multimedia applications using mechanism design," *IEEE Transactions on Multimedia*, vol. 9, no. 4, pp. 851–868, June 2007.

[33] F. Fu, T. Stoenescuand, and M. van der Schaar, "A pricing mechanism for resource allocation in wireless multimedia applications," *IEEE Journal on Selected Areas in Commnications*, vol. 1, no. 2, pp. 264–279, Aug. 2007.

[34] J. Ruiz-Hidalgo and P. Salembier, "On the use of indexing metadata to improve the efficiency of video compression," *IEEE Transactions on circuits and Systems for Video Technology*, vol. 16, no. 3, pp. 410–419, Mar. 2006.

[35] S. Kirkpatrick, C. Gelatt, and M. Vechhi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[36] P. Laarhoven and E. Aarts, "Simulated annealing: Theory and applications," Mathematics and its Applications, Dordrecht, Reidel, 1987.

[37] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, July 2003.

[38] "H.264/AVC ref. software," http://iphome.hhi.de/suehring/tml.

[39] F. Tonci, C. Adsumilli, M. Carli, A. Neri, and S. Mitra, "Buffer constraints for rate-distortion optimization in mobile video communications," *Proc. IEEE International Symposium on Signals, Circuits, and Systems*, pp. 71–74, July 2005.

[40] P. Li, X. Yang, and W. Lin, "Buffer-constrained R-D model-based rate control for H.264/AVC," *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 321–324, Mar. 2005.

[41] K.-P. Lim, G. Sullivan, and T. Wiegand, "Text description of joint model reference encoding methods and decoding concealment methods," *JVT of ISO/IEC MPEG and ITU-T VCEG, JVT-K049*, Mar. 2004.

[42] M. Jiang and N. Ling, "Low-delay rate control for real-time H.264/AVC video coding," *IEEE Transactions on Multimedia*, vol. 9, no. 3, pp. 467–477, June 2006.

[43] M. Tiwari, T. Groves, and P. Cosman, "Multiplexing video streams using dual-frame video coding," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Mar. 2008, pp. 693–696.

[44] M. Kalman and B. Girod, "Optimal channel-time allocation for the transmission of multiple video streams over a shared channel," in *Proceeedings IEEE Workshop on Multimedia Signal Processing*, Oct. 2005, pp. 1–4.

[45] K. Stuhlmüller, N. Färber, M. Link, and B. Girod, "Analysis of video transmission over lossy channels," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 1012–1032, June 2000.

[46] L. Lin and A. Ortega, "Bit-rate control using piecewise approximated rate-distortion characteristics," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 4, pp. 446–459, Aug. 1998.

[47] S. Boyd and L. Vandenberghe, "Convex optimization," *Cambridge University Press*, 2004.

[48] M. Tiwari and P. Cosman, "Selection of long-term reference frames in dual-frame video coding using simulated annealing," in *IEEE Signal Processing Letters*, vol. 15, 2008, pp. 249–252.

[49] M. Tiwari, T. Groves, and P. Cosman, "Bitrate allocation for multiple video streams at competitive equilibria," in *Proceedings IEEE Asilomar Conference on Signals, Systems and Computers*, Oct. 2008.

[50] J. Marschak and R. Radner, "Economic theory of teams," *Yale University Press*, 1972.

[51] A. Mas-Colell, M. Whinston, and J. Green, "Microeconomic theory," *Oxford University Press Inc.*, 1995.

[52] F. Edgeworth, "Mathematical psychics: An essay on the application of mathematics to the moral sciences," *London: C. Kegan Paul and Co.*, 1881.

[53] F. Kelly, A. Maulloo, and D. Tan, "Rate control for communication network: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, 1998.

[54] F. Li, D. Tolley, N. Padhy, and J. Wang, "Framework for assessing the economic efficiencies of long-run network pricing models," *IEEE Transactions on Power Systems*, vol. 24, no. 4, pp. 1641–1648, Nov. 2009.

[55] A. Mohsenian-Rad, V. Wong, and V. Leung, "Two-fold pricing to guarantee individual profits and maximum social welfare in multi-hop wireless access networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 8, pp. 4110–4121, Aug. 2009.

[56] G. Zachhariadis and J. Barria, "Dynamic pricing and resource allocation using revenue management for multiservice networks," *IEEE Transactions on Network and Service Management*, vol. 5, no. 4, pp. 215–226, Dec. 2008.

[57] D. Niyato and E. Hossain, "Market-equilibrium, competitive, and cooperative pricing for spectrum sharing in cognitive radio networks: analysis and comparison," *IEEE Transactions on Wireless Communications*, vol. 7, no. 11, pp. 4273–4283, Nov. 2008.

[58] ——, "A game theoretic analysis of service competition and pricing in hetrogeneous wireless access networks," *IEEE Transactions on Wireless Communications*, vol. 7, no. 12, pp. 5150–5155, Dec. 2008.

[59] ——, "Competitive pricing for spectrum sharing in cognitive radio networks: dynamic game, inefficiency of nash equili brium, and collusion," *IEEE Journal on Selected Areas of Communications*, vol. 26, no. 1, pp. 192–202, Jan. 2008.

[60] E. Malinvaud and M. Bacharach, "Decentralized procedures for planning," *Activity Analysis in the Theory of Growth and Planning, St. Martin's Press*, pp. 170–208, 1967.

[61] M. Tiwari, T. Groves, and P. Cosman, "Competitive equilibrium bitrate allocation for multiple video streams," *IEEE Transactions on Image Processing (to appear)*, Apr. 2010.