Modeling and optimization techniques for digital video communication

Pankaj Batra

Submitted in partial fulfillment of the

Requirements for the degree of

Doctor of Philosophy

In the Graduate School of Arts and Sciences

Columbia University

2004

ABSTRACT

Modeling and optimization techniques for digital video communication

Pankaj Batra

This dissertation investigates the modeling and optimization of several problems that arise during compression or adaptation of both frame-based and object-based (structured) multimedia content. In particular, we address:

(a) The optimal dynamic rate shaping (DRS) of Markov-1 sources.

We present the first extensive experimental study on the various DRS algorithms (causally optimal, memoryless, and rate-based) both in their constrained and generalized forms. The study proves the computational viability of the DRS approach to transcoding and identifies a range of rate shaping ratios for which it is better than requantization, both complexity-wise as well as in performance. We substantiate the almost-optimal experimental performance of the memoryless algorithm by analyzing the behavior of the DRS problem assuming a first order Markovian source. By deriving the statistical and rate-distortion characteristics of different components of the inter-frame rate shaping problem, we offer an explanation as to why the set of optimal breakpoint values for any frame is somewhat invariant to the accumulated motion compensated shaping error from past frames. This result is significant as it opens up the way to construct much simpler memoryless algorithms that give minimal penalty in achieved quality, not just for this, but possibly other types of algorithms. Of equal, if not more, importance is the very first use of matrix perturbation theory for tracking the spectral behavior of the auto-correlation matrix of the source signal and the motion residual it yields.

(b) The modeling of object based audio-visual communication systems, such as MPEG-4, both at the server (proxy) side for transcoding purposes and at the client end for adaptive play-out control.

For the server-side problem, we suggest a way of modeling object-based scalability and some functionalities that it brings with it. Our goal in this study is to find algorithms that aid in semi-automating the selective addition/dropping of objects from a scene to provide content scalability. The 'multiple choice knapsack problem (MCKP)' is used to model objects encoded at different target bit rates. We discuss several efficient ways to choose objects optimally at the transcoded rate, and build on the above integer problem to impose logical inter-dependencies among objects. Experimental results are presented validating the efficiency and performance of these linear programming based algorithms.

For the client-side problem, we discuss the use of an early-tardy penalization framework for implicitly modeling limited client resources and for scheduling of play-out under relaxed temporal constraints. This model and suggested algorithms would be useful to control the selective playout of content that a low-end client (for instance a PDA) does not have enough resources to consume. The problem is studied by classifying it into the 'composition' and 'stream-based' model. Our main contribution in this work is in introducing a hitherto un-noticed way of formulating the playout problem, and in giving an analysis of optimal algorithms and simplifying heuristics to solve it.

(c) The best possible intra and inter-frame quantization.

We use the MCKP formulation to study the intra-frame bit allocation problem as well, and suggest how to use the linear relaxation instead of the Lagrangian relaxation, dynamic programming or the marginal analysis approach. This method leads to a provable reduction in algorithmic complexity, is slightly more generic than the other ones and provides an alternate linear-programming duality based approach for a vast array of problems formulated in an Operational Rate-Distortion context. For the inter-frame quantization problem, we study the incremental improvement obtained by relaxing a so-called causality assumption and incorporate for dependencies explicitly in the model. As a way to solve the problem approximately, we discuss the usage of linear and Lagrange relaxations followed by

randomized rounding. Our experimental results show an improvement of about 0.5 dB over the causally optimal solution, thus justifying the usage of the causality assumption as a very practical one for reducing problem complexity for a marginal loss of quality.

The unifying thread in all the above studies is the extensive use of known mathematical programming tools for modeling and analysis. Although some of these have been fairly well known in other research communities, the application and introduction of each of these to video compression researchers is novel. Further, all the studies conducted led to either an improvement over the best known algorithm or offered substantial insight into the problem structure and ways to simplify it.

Contents

List of Figures

Chapter 2.

Chapter 3.

Chapter 4.

Chapter 5.

Chapter 6.

Appendices.

List of Tables

Chapter 2.

Chapter 3.

Chapter 4.

Chapter 5.

Chapter 6.

Acknowledgements

This work has benefited from interactions with a number of people over the course of past few years. First and foremost, I wish to express my deep gratitude to my advisor Prof. Alexandros Eleftheriadis for his continuing technical guidance and friendship. Alex had always been available for sound advice and much needed confidence at times when it mattered most. I will always be grateful to him for encouraging me to pursue my multi-disciplinary research interests. It was a pleasure to hear and learn from his criticism, suggestions, and a very practical way of looking at problems – hopefully some of this has eroded on to me.

Special thanks are due to the other members of the thesis committee, Professors Jay Sethuraman, Dan Ellis, Dan Rubenstein and Dr. Amy Reibman. I am thankful for their constructive suggestions to improve this work, for accommodating for the defense in their tight schedules, and most of all for their signatures!

I wish to thank the many Professors in the IEOR Department at Columbia for letting me sit in their classes. That interaction has given me much wider spectrum and has affected many portions of this thesis.

Throughout my stay in school, I benefited from (far too many) internships at nearby research labs. I thank Dr. Li-Fung Chang and Dr. M-T Sun for hosting me at Bellcore during my Masters, and for letting me use the wireless channel simulator for the experiments reported in Chapter 4. A couple of years later, I swam across the Hudson and spent a summer at David Sarnoff Research Center. That brief stay got me interested in MPEG-4 and subsequently led to the work in Chapters 2 and 3. I thank all the members of that group, in particular Dr. Iraj Sodagar, Dr. Ya-Qin Zhang, Dr. Tihao Chiang, and Dr. Sassan Pejhan, for a most enjoyable summer. In the summer of 1999, I visited Bell Labs to work on 3D audio - an area completely new to me. Although that project had somewhat mixed results, I hope to work in the area sometime in future. A "Two famous papers" paper that I read during that stay will always be an inspiration. In early 2000, I visited AT&T Research for a short-term project on MPEG-4 systems. I thank Bob Schmidt for clarifying several doubts about the IM1 player and for his

help with the implementation. If all this was not enough, I decided to swim the Hudson yet again, and beware, this time against the tide - and so began my first job at SMARTS. I have learnt a lot about real-world problems while at SMARTS and still continue to enjoy this very fast paced company.

Finally, my love goes to my parents, sister and brother in law for years of understanding, love, and faith. They are undoubtedly the cruise ship all this swimming happens on!

CHAPTER 1

Introduction

Contents

1.1 Historical Background

The last decade has seen a dramatic growth in the prevalence and widespread acceptance of multimedia services and related electronic consumer gadgets. This has been possible largely due to the interoperability achieved by adhering to standardized compression techniques. Most notable among these are the MPEG-1 and MPEG-2 developed by ISO and the H.26x series of standards of the ITU. While the first two triggered an explosion in digital television, DVD, Video CD, video recording, and broadcasting industries, the last one catalyzed the use of voice

and video over IP.

MPEG-1, MPEG-2, H.261 and H.263 had done their job for the bit rates and fairly limited level of interactivity for which they had been designed. However, as these standards were getting frozen, the Internet revolution took over and offered people alternative modes for receiving and interacting with audio-visual content. This led the ISO, in early 1993, to begin the MPEG-4 project that was to come up with a compression standard which glued together object based audio-visual content, while concurrently providing state-of-the-art compression efficiency ranging from very low to fairly high bit rates supporting a wide array of applications from mobile video telephony to high-end ones like professional video editing. Such ambitious goals did indeed at times diffuse the focus and direction of the project. However, it also led to the development of fairly new concepts of shape encoding, synthetic compression, facial and body animation, and scene description for defining the spatio-temporal layout and ways to interact with dynamic aspects of content along with a more flexible way of multiplexing and synchronizing it.

While standards make technologies interoperable and accessible to the general pubic, they also leave considerable room for optimization, system modeling and improvement. This thesis deals with several optimization problems addressing: (a) the best possible intra as well as inter-frame video compression efficiency achievable, (b) the optimal dynamic rate shaping of sources with a known distribution, and (c) the modeling of object based audio-visual communication systems such as MPEG-4, both at the server (proxy) side as well as the client end for adaptive play-out control. The unifying thread in all these studies will be the extensive use of known mathematical programming tools for modeling and analysis. Although some of these tools have been fairly well known in other research communities, their application and introduction to video compression researchers in novel. These tools include, (a) the use of linear relaxations instead of the Lagrangian, dynamic programming or marginal analysis approach for solving the somewhat traditional bit-allocation problem, (b) the use of rounding algorithms to approximate integer programming solutions, (c) the use of an early-tardy penalization framework for implicitly modeling client side resources and for scheduling of play-out control, and (d) the use of matrix perturbation theory for tracking the spectral behavior of the auto-correlation matrix of

an AR(1) signal and the motion residual signal it yields. Each of the above leads to either an improvement over the best known algorithm or offers substantial novel insight into the problem structure and ways to simplify it. We discuss these issues in more detail in the following section and subsequent chapters.

The material contained in this thesis appears in the following papers: [11][12][13][14][15] [16][17][45][46].

1.2 Outline and contributions of the thesis

1.2.1 Modeling MPEG-4 Systems and contributions to operational R-D theory

MPEG-4 is the first visual coding standard that allows coding of scenes as a collection of individual audio-visual objects. A large portion of this thesis deals with abstracting and building models for analyzing problems related to MPEG-4 systems. We start in Chapter 2 with formulations for modeling object-based scalability and some functionalities that it brings with it. Our goal in this study is to find algorithms that aid in semi-automating the selective addition/dropping of objects from a scene to provide content scalability. Objects are assumed to have been pre-assigned preferences (priorities) and weights (bit-rates), and our objective thereafter is to find the optimal set of objects to composed the scene. We begin with a simplistic model for object-based scalability using the *"knapsack problem"* – a problem for which the optimal object set can be found using known schemes such as dynamic programming, the branch and bound method and approximation algorithms. The above formulation is then generalized to model authoring or multiplexing of scalable objects (e.g., objects encoded at various target bit-rates) using the *"multiple choice knapsack problem"*. We relate this model to several problems that arise in video coding; the most prominent of these being the bit allocation problem. Unlike previous approaches to solve the operational bit-allocation problem using Lagrangian relaxation, we suggest to use the linear programming (LP) relaxation of this problem. We show that for this problem the duality gap for Lagrange and LP relaxations is exactly the same. The LP relaxation is solved using strong duality with dual descent – a procedure that can be completed in *linear* time. We show that there can be at most two fractional variables in the optimal primal solution

and therefore this relaxation can be justified for many practical applications. This work reduces problem complexity, guarantees similar performance, is slightly more generic, and provides an alternate LP-duality based proof for earlier work by Shoham and Gersho [144].

Next we show how additional constraints may be added to impose *logical inter-dependencies* among objects in a presentation and discuss how object aggregation can be exploited in reducing problem complexity. The marginal analysis approach of Fox is suggested as a method to do re-allocation with *incremental inputs*. It helps in efficiently re-optimizing the allocation when a system has user interactivity, appearing or disappearing objects, time driven events etc. (features that are all an integral part of MPEG-4 systems). Finally, we present a brief experimental study to analyze the performance of some of the algorithms discussed above. Using rate-distortion data collected from encoded images, we compare the complexity and solution qualities of exact methods and LP relaxations, both for problems with and without extra dependencies.

We close this chapter by suggesting that approximation algorithms for the multiple choice knapsack problem, although a theoretical nicety for the most part, can be used to quantify the complexity vs. quality tradeoff at the encoder in a tunable and universal way.

Chapter 2 deals with modeling issues related to MPEG-4 systems. These may aid in transcoding (structured) content once it has been authored. Most of these algorithms are likely to reside at the content server or an intermediate proxy. The next chapter explores flexibilities that exist at the client side. We will present models and algorithms for adaptive playout control and discuss an alternative (implicit) way of modeling resource constraints.

1.2.2 Playout control in MPEG-4 based systems

In Chapter 3, we investigate algorithms to schedule the playout or retrieval of audio-visual objects under relaxed temporal restrictions. These algorithms control the selective playout of content that the client-side (for instance a PDA) does not have enough resources to consume. Unlike Chapter 2, we present an implicit way of modeling resource contention and use an early-tardy (ET) penalization framework to analyze the problem. In an ET framework, events that

happen both before and after their ideal due times get penalized. For the playout problem, the earliness penalty comes from decoding and buffering early objects; and the lateness penalty from the loss in correlation that results by skipping some video object instances. We discuss the justification for using an ET framework for the playout scenario and present ways to estimate parameters of this model.

Two sub-parts are used to analyze the ET problem. The first (called composition problem) involves scheduling the decoding and playout of objects having a common display time, e.g., while doing a pull-based display in a web browser. The second (called streaming problem) discusses scheduling issues with isochronous media. We present a brute force optimal scheme and several heuristics to solve the composition model. These heuristics use a preliminary fast sub-optimal scheme to get seed solutions and subsequently use a local neighborhood search procedure to improve the seed solution by giving it a minor perturbation. The streaming problem leads to a non-work-conserving scheduling discipline for transmission or play-out control. A method to optimally insert idle time in the bitstream, or equivalently, a way to add optimal slippage in displaying frames is given. The algorithm can be viewed as a way to reduce potential burstiness in traffic at a minimal cost. Our main contributions in this work are in introducing a hitherto un-noticed way of formulating the playout problem, usage of this framework, and analysis of heuristics.

1.2.3 Content-based video transmission over wireless channels.

As a byproduct of our study in Chapter 3, we find how spatio-temporal variance in visual content affects our scheduling decisions. Chapter 4 explores the flexibilities one gets due to visual cues in further detail. We present content-based approaches for visual segmentation and resource allocation for transmission over wireless channels. Unlike earlier chapters in this thesis, most of the work discussed in Chapter 4 predates the MPEG-4 and later MPEG-7 standards. While we use a frame-based codec (H.263), nothing prevents us from using similar ideas in object-based standards.

We discuss content dependent FEC and ARQ schemes to do adaptive resource allocation for video transmission over wireless channels. The material in this chapter may be best interpreted in the context of the MPEG-7 standard, by which the visual information will soon be supplemented with metadata representative of its content. FEC based schemes are used to provide class-dependent error robustness and a modified ARQ technique for giving constrained delay and loss. We use frame-type (extent of intra coding), scene changes, and motion-based procedures to provide finer level of control for data segmentation in addition to standard headers and data-type (motion vectors, low and high frequency DCTs) based segmentation. An experimental simulation platform is used to test the objective (SNR) and subjective effectiveness of proposed algorithms. The FEC schemes improve both objective and subjective video quality significantly. An experimental study on the usage of selective repeat ARQ for one-way real time video applications is presented. We discuss the constraints of using ARQ under display deadlines, limited buffering requirements and small initial startups. The proposed ARQ scheme greatly reduces the packet errors, when used along with decoder buffer control and source interleaving. The common theme integrating the study of our FEC and ARQ algorithms is the use of visual content in deciding the resource allocation.

1.2.4 Dependent quantization and its extensions

Early in Chapter 2 of this thesis we discuss how to represent logical inter-dependencies among objects that compose a presentation. Although this can be done easily, the resulting integer problem could become extremely difficult to solve. As a way around it, Chapter 2 proposes to use LP relaxations and after solving this relaxation we freeze the variables that satisfy integrality in the optimal LP solution at their respective values. This is followed by an exhaustive search for the best solution over fractionally assigned variables. The limited experiments that we conduct in Chapter 2 gave very encouraging results for the dependency structure considered. However, as an approach, this is certainly not generic, and may never even lead to a feasible integral solution.

In Chapter 5, we revisit this issue by restricting our attention to a special dependency structure. We present multiple choice knapsack problem (MCKP)-like formulations to model predictive

inter-dependencies in motion-compensated coders, a problem that has been researched by earlier authors also. Our formulation lets us relax a so called causality assumption made by these authors and allows us to study the incremental improvement thus obtained. A rounding procedure is used to generate heuristic solutions after solving linear relaxation of the proposed formulation. This approach has less complexity than earlier work in the area based on the usage of dynamic programming. Experimental results show an improvement of about 0.5 dB over the causally optimal solution. From a practical standpoint, this validates that the causality assumption is a very justified one to reduce problem complexity for a marginal loss of optimality.

1.2.5 Optimal dynamic rate shaping of Markov-1 sources

Yet another problem where the affects of dependency are known to have negligible affect on the optimal point selection is dynamic rate shaping. In this problem, one attempts to selectively drop a sequence of DCT coefficients so as to meet a rate constraint while minimizing the overall distortion due to shaping. Even if we make the causality assumption for this problem and optimize assuming given optimal decisions from the past, one needs to keep track of motion residual signal trickling from earlier frames. However, our earlier experimental results suggested that the optimal breakpoint stays pretty much the same even if we drop the accumulated motion compensated shaping error. Note that this memoryless assumption, on top of the causality assumption, greatly simplifies the practical use of (almost optimal) dynamic rate shaping. A desire to better understand this somewhat non-intuitive behavior led us to rigorously investigate the optimal DRS problem with a known source distribution.

In Chapter 6 we derive the statistical and rate distortion characteristics of the different components of the inter-frame DRS problem assuming an AR(1) source. By using results from Matrix Perturbation Theory along with the above facts, we show that the spectral characteristics of the source signal alone and the source along with the motion residual signal track each other. Therefore, the set of optimal breakpoints for any frame is somewhat invariant to the accumulated motion compensated shaping error from past frames and may be very reasonably approximated using the current frame shaping error alone. This result is very significant since it opens up the

way to construct much simpler memoryless algorithms that give minimal penalty in achieved picture quality.

CHAPTER 2

Modeling and efficient optimization for object-based scalability and some related problems

---

Contents

2.1 Introduction

MPEG-4 [76][77][146], the latest in the series of multimedia compression standards being developed by the ISO, is the first object-based visual coding standard. Unlike the H.261/3/3+ and the MPEG-1/2 standards, which were intended primarily for compression, in MPEG-4, scenes are coded as an aggregation of synthetic and natural audio-visual objects, text and graphics. These are supplemented with a flexible scene description. This brings in a lot more flexibility in the way content is created, distributed, and consumed. At the authoring end, users will soon have tools to create rich multimedia presentations on their low-end PCs. Functionalities similar to those in text-document creation and editing will be supported for images, audio and video. On the transmission side, the compression standard is much more robust to channel errors. At the consumption end, users will be able to interact with the scene and change its behavior. MPEG-4 features such as scene updates, routes, events etc. [7][147] provide the ability to change content quality by changing fields of some nodes, or by adding or removing locally stored objects from the scene. This also helps in filtering content so that it fits the end-terminal's capabilities and adapts well to changing network conditions. To complete the loop, it is also expected that there will be a back channel signaling capability (an upstream channel) between the client and server to do content manipulation at the source.

Not only does MPEG-4 provide tools for coding arbitrarily shaped objects, it also has the facility to flexibly compose the scene and to interact with it. The main strength of the standard comes from this object-oriented approach that lets a user interact with audio visual content in ways

more creative than traditional VCR modes like stop, rewind, play etc. The composition information for the scene is carried as a separate elementary stream called Binary Format for Scenes (BIFS) [147]. BIFS follows a Virtual Reality Modeling Language (VRML) [78] like syntax. It describes a scene as a hierarchy of nodes. Audio-visual objects are leafs of this tree and higher up in the tree are transforms that control rendering of objects in space and time. BIFS commands can be used to add/delete objects, to activate or de-activate enhancement information for scaleable coding, or to change attribute values of some objects without actually interacting with the compressed object. This way, multimedia content can be made to adjust itself to suit the transmission rate and quality. Below, we give a brief outline of supported features in MPEG-4 that are relevant in the context of this work. These are:

- ***Tools for encoding objects.*** Objects could be still images (textures), video (arbitrarily shaped or frame based), synthetic or natural audio, speech, 2D and 3D graphics, text, face and body animation, etc.
- ***A facility to flexibly compose the scene.*** As mentioned earlier, the composition information for a scene is carried as a separate elementary stream called BIFS.
- ***Local user interaction with audio-visual content.*** BIFS-Commands can be used to add/delete objects, to activate or de-activate enhancement information for scaleable coding, or to change attribute values of some objects without actually interacting with the compressed object. In addition BIFS-Anim may also be used to continuously animate a scene.
- ***Remote interaction with the server.*** An ability to interact with the server using back-channel signaling will be supported.
- ***MPEG-J.*** The work done under MPEG-Java (MPEG-J) provides a set of platform independent application program interfaces (APIs) to be used for terminal control, processing, and resource management. These APIs include – scene graph, network, decoder, devices, and system capabilities API.

2.1.1 Results

MPEG-4 provides a structured mode of content representation, distribution and consumption. Thus, the more traditional forms of spatio-temporal scalability [76][77][146], bit allocation

[115][131][140][144], and rate control [37][115][140], are likely to be soon supplemented by object-based ones. In this work, we present formulations for modeling object-based scalability and some functionalities that it brings with it. We assume that objects have been compressed beforehand using scalable or non-scalable formats. The primary focus thereafter will be on optimal object selection and bitrate distribution among the objects. We discuss optimal and asymptotically optimal algorithms and heuristics for admission control of objects given their nominal operational R-D point. We also discuss ways for authoring using scalable objects that have been constant bit-rate (CBR) encoded at various candidate target bit rates. Furthermore, we present generalizations for imposing inter-dependencies among objects, re-optimizing with incremental (user) inputs and suggest some simple linearization schemes.

We start with an overly simplistic formulation in Section 2.2.2 and model object based scalability as a *"knapsack problem (KP)",* an extensively researched problem in Operations Research (OR) literature. This is used to model a value-added service in which richer content is made available for higher price or with an increase in resources. For sake of completeness, we outline some known ways to do optimal object selection (or, admission control) based on dynamic programming and the branch and bound method. Multimedia documents nowadays may consist of a large number (easily around 100-1000s) of objects. Therefore, in order to reduce the complexity involved in finding the optimal object set for large problems, some heuristics are suggested.

In Section 2.3, we generalize the simplistic formulation of Section 2.2.2 to model authoring or multiplexing using scalable objects (e.g., objects encoded at various target bit-rates). This is done using the so-called "*multiple choice knapsack problem (MCKP)"*. This problem may also be solved optimally using a dynamic programming approach. Unfortunately this avenue is impractical due to a pseudo-polynomial space and time complexity. Instead, the primary focus in this chapter will be on solving the linear programming (LP) relaxation of the above problem. Almost all earlier work on the bit-allocation problem (which can be considered a special case of MCKP) is based on Lagrangian relaxation (LR). Unlike Lagrange relaxations (where certain hard constraints are relaxed into the objective function), in an LP relaxation we relax integrality constraints on the variables. Note that integrality constraints are retained in Lagrange

relaxations. Since both LP and LR are relaxations, both give an infeasible optimal point that is an upper bound on the objective function; but each can be used to find a heuristic solution by a trivial rounding operation. We further go on to prove that both these relaxations lead to exactly the same duality gaps (or approximation errors) for the problem under consideration. However, an LP relaxation leads to a somewhat faster *(linear time)* algorithm that exploits strong duality with dual descent. This gives a slight computational improvement and an alternate proof of earlier work by Shoham and Gersho [144].

In Section 2.4, we give modifications to enforce *inter-dependencies* among objects in a presentation. In particular, we explain how logical implications like *if-then* and *either-or* constraints among objects can be imposed. We discuss how simple transformations can be used to linearize certain forms of non-linearities. In Section 2.5, we suggest another transformation that exploits object aggregation and helps in reducing problem complexity. The marginal analysis approach of Fox [56] is suggested as a way to do *online* re-allocation in Section 2.6. It is efficient to re-optimize the allocation when the system has provisions for user interactivity, scene updates, on-the-fly changes like object transcoding, or other time driven events.

In summary, the main contributions of this work are the following:

1. We present a framework useful for modeling object-based scalability. It is easily extensible to incorporate several practical enhancements such as logical inter-dependencies among objects, simple non-linearities in the model, online implementations, and simplifications obtained by object aggregation.
2. The MCKP formulation used for modeling scalable objects leads to a much faster (linear time) algorithm for the much researched bit-allocation problem. The solution procedure is based on solving the LP relaxation using dual descent. This work reduces problem complexity, guarantees similar performance, is slightly more generic, and provides an alternate LP-duality based proof for earlier work by Shoham and Gersho [144].
3. We briefly suggest that approximation algorithms may be used to quantify complexity vs. quality tradeoff at the encoder in a tunable way. These algorithms lead to lossy universal quantization (compression) schemes.

For publications related to this chapter, see [15].

2.1.2 Related work

MCKP is similar to a number of problems researchers have studied in the video coding literature. Most of these have been formulated in an operational rate-distortion framework. The allocation of bits to a discrete set of quantizers is the most prominent one among these. In this problem, we try to assign quantizers to independent sets of macroblocks (or source frames) in order to minimize the total distortion without violating a bitrate constraint. This problem has been attacked in two ways. The first approach is based on building a stochastic model for the R-D behavior of quantizers. Such a model helps to predict the average distortion (averaged over source ensembles) for encoding at a given rate. This regression step is normally done off-line. Once a stable model has been built, optimization is done assuming similar R-D behavior. Typically, non-linear programming tools (e.g., steepest descent or quasi-Newton methods) are used to find the optimal allocation after converting the problem to an unconstrained one. Note that the bottleneck complexity of the first approach comes from the model building stage. On the contrary, the second avenue to address this problem skips the offline model building step and the optimization is done directly on operational (gathered) data points. Therefore, there are no residual errors that would have been introduced during the modeling step. This approach, however, requires a lot of data to be continuously gathered. Moreover, a further (combinatorial) hardness of the second approach arises from the fact that quantization matrices may be scaled up or down using only a *discrete* set of values. We will focus primarily on the latter approach in this work. However, nothing prevents using this approach in conjunction with a predictive model that helps to ease the burden of gathering data points.

There has been a lot of work in, and generalizations of, the bit-allocation problem. With these, a wealth of new applications has emerged applying these results. Some recent ones include R-D optimal shape coding [140], optimal breakpoint selection for data-partitioning [45], server retrieval scheduling [117], selecting best wavelet packet [129], jointly R-D optimal segmentation, optimal displacement vector field (DVF) and displaced frame difference (DFD) encoding, quad-tree decomposition and scanning etc. We refer the interested reader to [140] for

an extensive survey. Despite a flood of recent applications, some of the original work in this problem dates as far back as the 1960s [71]. To the best of our knowledge, the primary contributions are due to the following:

- Segall [141] – This work assumed a convex and functional R-D curve, for example an exponential one. Since continuous functions were assumed, this method is closer to traditional rate distortion theory as opposed to an operational one.
- Fox [56] – Fox's solution was based on a so-called marginal analysis method applied on *distinct* and *discrete* quantizer functions. Although this method generally gives good error gaps, it assumes convexity of the R-D function; or at the very least requires that one identifies the convex hull for each quantizer set.
- Trushkin [158] – Presented how to use dynamic programming to solve this problem. Although optimal, it was noted to have an impractically high complexity. For the sake of completeness, a similar dynamic program is outlined in Appendix A.2.
- Shoham and Gersho [144] – Considered an arbitrary set of quantizers and made no assumptions on convexity or monotonicity. They suggest that convexity assumptions are usually violated due to non-linearities in coding standards and through the use of small block sizes that do not get us close to convex R-D bounds. The analysis was based on Lagrangian relaxation as applied to integer programs using Everett's result and its extensions [52][53] [55].
- Ramchandran, Ortega, Vetterli [115][131] – Suggest allocation schemes with dependencies among the quantizers. An optimal buffer control strategy was also found using dynamic programming. However, we believe that complexity considerations in using dynamic programming were not rigorously addressed in these papers.
- Schuster and Katsaggelos [140] – Review developments in operational R-D theory, and suggest recent applications to shape coding etc.

Despite the above developments, complexity considerations in solving this problem have still not been addressed adequately. The marginal analysis approach of Fox assumes convexity of R-D curves. If convexity is not guaranteed, we have to find the convex hull of the operating points before using the algorithm. This needs a minimal $O(n\log n)$ complexity if there are n operating points. Even after this, the solution lies on the convex hull of the operating points, and so it can

be sub-optimal. Dynamic programming [158], on the other hand, gives the optimal solution and does not make any assumptions of convexity and monotonicity. However, it leads to a pseudo-polynomial time algorithm whose complexity blows up very rapidly. Finally, Shoham and Gersho's approach, although almost there, did not rigorously discuss how pruning helps reduce a worst-case search over the $O(n^2)$ singular points.

The primary focus in this work will be on solving the LP relaxation of an integer problem (instead of its Lagrangian relaxation). Using strong-duality we will solve the LP-dual problem instead. Finally, exploiting complementary slackness, we will construct the optimal solution to the primal problem. The overall complexity of the procedure will be shown to be *linear* in the size of the problem. This improves on the celebrated work of Shoham and Gersho by an *O(n)* factor.

2.2 Preliminaries

2.2.1 Mathematical background

The problem addressed in this Chapter is solved using several mathematical programming tools. Since video communication researchers may not be well versed in these techniques, this section will introduce them at an elementary level. For a more complete treatment, the interested reader may refer to [18][20][41].

2.1.1.1 Linear Programming

A linear programming problem has the following general structure:

$$\max\{c^T x \mid Ax \le b, x \ge 0\} \text{ or } \min\{d^T x \mid Bx \ge h, x \ge 0\} \text{ with } b = -h, c = -d, \text{ and } A = -B.$$
$$c : n \times 1,\ x : n \times 1,\ b : m \times 1,\ A = (a_{ij}) : m \times n.$$

It is often convenient to convert the above to an equivalent *equality constrained* problem:

$$\max\{(c^T, 0)\begin{pmatrix} x \\ \tilde{x} \end{pmatrix} \mid Ax + I\tilde{x} = b \text{ and } x, \tilde{x} \ge 0\}$$

The extra variables introduced to ensure equality are also known as *slack variables*. The feasible region *R* of either of the two problems is a *convex* polyhedron given by the intersection of *(n+m)*

half-spaces. Let $x^l (l = 1,..,s)$ be the vertices of *R*. Note that $s \le \binom{n+m}{m}$. Since the feasible region is the convex hull of these vertices, any point *x* in *R* may be written as the convex combination $\sum_{i=1}^{s} l_i x^l$ with $l_l \ge 0$ and $\sum_{i=1}^{s} l_l = 1$. The above linear programming problem, which aims to maximize $c^T x$, therefore, reduces to the determination of the vertices of *R*. One of these vertices is guaranteed to be the optimal one.

The Simplex method of G.B. Dantzig gives an algebraic characterization of these vertices. It is a generic technique that solves any linear programming problem in a finite number of steps and is based on the idea of improving the cost while moving from one vertex of the feasible region to another, and terminating only when certain optimality criteria are met. Several refinements have led to many variations of the Simplex algorithm. With the computational power available today, this method is usually regarded as fairly efficient, and is routinely used to solve problems with several thousands of variables or constraints.

Additional linear programming facts that would be useful later are as follows:

1. Basic feasible solution: Any vector *x* having exactly *m* positive components that belong to *m* linearly independent columns of the matrix *A* is called a basic feasible solution.

2. Dual problem: The problem - $\min\{b^T y \mid A^T y \ge c, y \ge 0\}$ - is called the dual problem for the problem - $\max\{c^T x \mid Ax \le b, x \ge 0\}$. The latter is also known as the primal problem.

3. Weak and strong duality: If *x* and *y* are primal and dual feasible, respectively, then $c^T x \le (A^T y)^T x = (y^T A)x \le y^T b = b^T y$. Further if x^* and y^* are feasible and $c^T x^* = b^T y^*$, then x^* and y^* are optimal for the primal and dual problems. In fact, it has also been shown that x^* is a solution of the primal problem *if and only if* there exists a dual feasible y^* such that $c^T x^* = b^T y^*$; y^* is a solution of the dual problem *if and only if* there exists a primal feasible x^* such that $c^T x^* = b^T y^*$.

2.1.1.2 Integer Programming

Integer programming addresses optimization problems with the following format:

$$\max\{c^T x \mid Ax \le b, x \ge 0, x \text{ integer}\} \text{ with } c: n\times 1,\ x: n\times 1,\ b: m\times 1,\ A=(a_{ij}): m\times n.$$

One searches for the maximum of the objective function no longer among all the points of the feasible region *R*, but only among the integer or lattice points contained in *R*. The need for formulating and solving integer problems comes from the fact that in certain applications of LP, fractional solutions tend to be practically meaningless. In the context of this Thesis, integer *indicator* variables would be used to characterize discrete selection choices in certain resource allocation problems. Enlisted below are the methods that we shall use to solve integer problems, along with the places where they get used:

- Optimal methods:
  - Dynamic programming:
    - Section 2.2.2.1 illustrates this enumerative technique for the Knapsack problem, while Appendix A.2 shows how to exploit it for MCKP.
    - Chapter 3 uses this method for solving the early-tardy playout scheduling problem.
  - Branch and bound:
    - Appendix A.1 outlines this generic (though combinatorial) technique for solving any linear integer problem.
    - In Section 2.6, we shall utilize this method to benchmark the performance of sub-optimal heuristics for the Multiple Choice Knapsack Problem.
- Approximate methods:
  - Greedy heuristics:
    - Section 2.2.2.2 illustrates the basic idea behind greed using the Knapsack problem.
    - A sequence of greedy heuristics would be used for constraint satisfaction in dependent quantization problems of Chapter 5.
  - Linear relaxation (with or without deterministic and/or randomized rounding):

- A deterministic technique for rounding the possibly fractional LP optimal solution to MCKP is suggested in this Chapter.
- Randomized techniques, which treat the fractional LP optimal points as probabilities while rounding, would be used for tackling quantization problems with dependencies in Chapter 5.

- Lagrangian relaxation (with or without deterministic and/or randomized rounding):
  - Lagrangian relaxation is perhaps the most popular optimization method amongst video communications researchers. In this Chapter we will discuss certain key differences between Linear and Lagrangian relaxations for MCKP.
  - Chapter 5 uses Lagrangian relaxation as an escape mechanism if the resource constraint becomes difficult to satisfy while rounding based solely on the LP relaxation.
  - Finally, Chapter 6 would use Lagrangian techniques once more, this time for solving the causally optimal dynamic rate shaping problem.

The following section continues our tutorial introduction to Integer Programming while presenting the techniques using a problem pertinent to and a precursor for the more realistic models starting in Section 2.3.

2.2.2 A simplistic formulation

We want to design algorithms that aid in authoring and customization of multimedia documents by semi-automating the selective addition/dropping of objects from a scene to provide content scalability. We will use MPEG-4 systems to present the case for value added services. The approach and modeling, however, are applicable in other scenarios such as with documents having structured HTML or XML content. Assume that there are n objects $O_1,..,O_n$ with *average* bit-rate/buffer requirements $s_1,..,s_n$ that may be included in the presentation. There is a pre-specified aggregate buffer or bandwidth constraint, B, that must not be exceeded. The inclusion of an object to the presentation increases the overall quality by p_i and decreases the available

resource by an amount s_i. The coefficients p_i ($i=1,..,n$) are in a common normalized unit, and we start by assuming that there are no inter-dependencies among objects, e.g., from a synchronization, compression, or inclusion point of view. We assume that objects have been pre-compressed at a constant bit-rate. Thus, s_i corresponds to the target bit-rate at which O_i is encoded, while p_i is a measure of its average quality (average peak signal to noise ratio (PSNR), a quality metric based on visual perception, a priority listing or user perceived utility). The objective is to find a subset of objects to include so as to maximize the overall quality. Since we may also view the problem as an admission control one, the coefficients p_i and s_i indicate nominal or minimal operating points for an object. Using the above assumptions and notation, the problem may be formulated as:

KP: Maximize $\sum_{i=1}^{n} p_i X_i$

Subject to: $\sum_{i=1}^{n} s_i X_i \leq B$; with $X_i \in \{0,1\}$ for $i=1,...,n$

Note that X_i is an indicator variable that is 1 if and only if O_i is included in the presentation. The above is the famous 0-1 knapsack problem (KP) that has been extensively researched in the OR literature. It is an NP-hard problem, which means that no known algorithm solves it in polynomial time for all instances of the problem. However, some generic methods such as branch and bound and dynamic programming may be used to solve KP. Unfortunately, both of these have very bad worst-case complexities (exponential for Branch and Bound and pseudo-polynomial for dynamic programming). Therefore, it is unlikely that either of these methods can be used in practice.

2.2.2.1 Finding Optimal Solution

The forward version of dynamic programming (DP) is one of the most intuitive (although somewhat impractical) ways to solve KP. Dynamic programming, originally due to Bellman [20], is an enumerative procedure for solving certain optimization problems. It solves a large (separable) problem by dividing it into a series of simpler ones (see Figure 2.1). There are $n+1$ stages in the illustrated DP algorithm. Each stage corresponds to an object added to the presentation. Each stage has at most B states. Each state corresponds to a given available bandwidth. Let us use the following notation:

$f_k(m)$ = maximum objective function value if m units of bandwidth are available and we may use only objects $1,..,k$

m = bandwidth for the current stage, $m \leq B$

The program starts from the first stage and moves forward stage by stage. In the end we find the optimal stage n decision, and the solution is traced out by backtracking the algorithm. More formally, the dynamic program is,

STEP 1: $f_1(m) = 0$ *for* $m=0,1,...,s_1-1$ [Object 1 is not included]

$= p_1$ *for* $m=s_1,...,B$ [Object 1 is included]

STEP 2: $f_k(m) = f_{k-1}(m)$ *for* $m=0,1,...,s_k-1$ [Object k is not included]

$= Max\{ f_{k-1}(m), p_k + f_{k-1}(m-s_k)\}$ *for* $m=s_k,...,B$

[Object k is not or is included, respectively]

The above program may be followed using hints given on right hand side. We proceed by calculating $f_k(1), f_k(2), f_k(3)$ … until we obtain $f_k(B)$ for all $k=1,..,n$. The optimal objective value is given by $f_n(B)$, and we can trace out the solution (i.e., X_i s) by backtracking the optimal path. Note that step 2 requires $O(B)$ time and has to be run n times, once for each stage. Thus, the overall time and space complexity of DP is $O(nB)$. This complexity is polynomial in n if B increases as a polynomial with n. Note that the term in B is not logarithmic. Hence this algorithm is said to be pseudo-polynomial as it needs time that is exponential in the binary representation of its input data. This complexity blows up very fast since the granularity in which B has to be represented is usually very small. In order to reduce complexity, we may prune out some states that can never be part of an optimal solution. Consider stage k of the algorithm with a current state S_1 and a previous state S_2. In case we find that (a) the sum of bandwidth requirements for objects in S_1 is more than that in S_2, and (b) the aggregate utility of objects in S_1 is less than or equal to that in S_2; then S_1 may never be part of an optimal object set. If a subset from first k objects includes S_1, then we could trivially do better by replacing this

subset with S_2, leaving everything else unchanged. Hence, we may prune out state S_1 from the search.

A branch and bound method is another generic method that may be used to solve KP. This procedure is fairly general and can be used to find solutions for any pure or mixed (linear) integer-programming problem. Since we will refer to it several times, we outline the procedure in Appendix A.1. It must, however, be noted that both branch and bound and DP are not practical and give solutions in reasonable time only if the number of objects is small. Due to complexity considerations, we need to resort to heuristics instead.

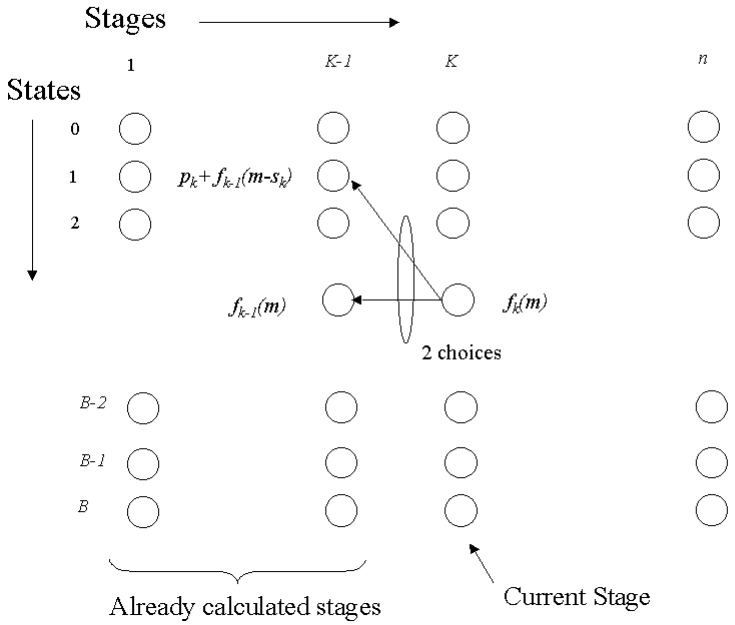


Figure 2.1 Forward version of dynamic program

2.2.2.2 Finding Approximate Solution

A greedy algorithm (GA) [110] is perhaps the most intuitive heuristic to solve this problem. GA may be summarized as follows:

STEP 1. Reorder the objects in decreasing order of p_i/s_i.

STEP 2. Add objects to the scene in this order so long as the bitrate bound (B) is not exceeded.

We admit objects in order of decreasing quality densities. Although simple, the algorithm can lead to a solution that is arbitrarily far from the optimal one for certain problem instances. We can assure that the solution is within a multiplicative factor of 2 from the optimal if we choose the better of the following two solutions: (a) That obtained using GA, and (b) that of allowing in only the critical object (i.e., the last object in GA adding which just violates bandwidth bound). The interested reader may find a proof in [110] and references therein.

2.2.2.3 Extensions

The assumptions we made in Section 2.2 are oversimplified in several aspects. In this Section, we relax some of these assumptions from a video coding/communication perspective, and discuss ways to find optimal and approximate solutions. Some of the issues we will address are:

a. A binary formulation for inclusion or exclusion of objects has been presented. Can we extend the approach to allow for including scaled or transcoded versions of objects? Can the problem be solved faster than earlier approaches under some relaxations?
b. How can we impose inter-dependencies that may exist among objects that compose a presentation? How do we deal with simple non-linearities in the objective function or in the constraints?
c. Can aggregating objects (i.e., treating them in a similar way) help in reducing complexity?
d. Can we re-optimize faster (and *online*) when the scene has on-the-fly or time driven events, user interactivity, appearing or disappearing objects, etc.?
e. How is the problem related to some know problems in video coding literature, and can some of them be solved using the approaches given here?

2.3 Modeling scalable objects

2.3.1 Formulation

At the cost of poorer quality, smaller spatial size or worse temporal continuity, an image sequence can be encoded at various target bit rates [76][77][140][146][157]. Several forms of scalability may be used to compress video at the desired target bit rate [37][149] or to give layered representations for it [77][157]. The typical motivation for scalability is to have a base

layer that gives a coarse signal representation; and as more bandwidth becomes available, progressive enhancements may be added. Many online schemes for extracting scaled or transcoded versions from a pre-compressed video stream are also known. Examples of these include schemes like dynamic rate shaping [45], requantization, frame dropping, etc. In addition, many compression schemes (e.g., wavelet or sub-band based) are inherently scalable. Therefore, while aggregating objects during authoring or multiplexing, it may be worthwhile to have some additional freedom with their bit-rates and quality perception at chosen bit-rates. We incorporate this into the model by using a generalization of the knapsack problem, called the "*Multiple Choice Knapsack Problem (MCKP)*". Unlike the simple problem, in MCKP, we have some disjoint *sets* of objects. Objects in each of these sets have utilities and weights associated with them. At most one object from each set may be added to the presentation. As before, the idea is to select one version from each set so as to maximize the overall quality, while not exceeding the bitrate bound. In our context, this is similar to having candidate sets of transcoded versions of audio-visual-objects (or alternative presentations), and we would like to select one version for each object. Let p_{ij} be the average PSNR of the j^{th} version of object i and s_{ij} the target bitrate at which it is encoded. Let there be n_i candidate bitrates for the i^{th} object, one of which must be chosen. Therefore, we wish to:

(MCKP): Maximize $\sum_{i=1}^{N}\sum_{j=1}^{n_i} p_{ij}X_{ij}$

Subject to: $\sum_{i=1}^{N}\sum_{j=1}^{n_i} s_{ij}X_{ij} \leq B$ *(C1)*

$\sum_{j=1}^{n_i} X_{ij} = 1$ for $i=1,2,...,N$ *(C2)*

$X_{ij} \in \{0,1\}$ $i=1,2,...,N$ and $j=1,2,...,n_i$ *(C3)*

The solution is a binary set $\{X_{ij} \mid i=1,2,...,N$ and $j=1,2,...,n_i\}$ that specifies which objects are selected. Like the knapsack problem, MCKP is also known to be NP-hard.

2.3.2 Intuition behind the Linear Programming (LP) relaxation

The primary focus in this work will be on solving the LP relaxation of an integer problem (instead of its Lagrangian relaxation). Using strong-duality we will solve the LP-dual problem

instead. Finally, exploiting complementary slackness, we will construct the optimal solution to the primal problem. The overall complexity of the procedure will be shown to be *linear* in the size of the problem. This improves on the celebrated work of Shoham and Gersho by an *O(n)* factor. The algorithmic framework used here to search for the optimal dual variable is originally due to Dyer and Zemel [44][165]. However, we are not aware of prior use of LP relaxation of this problem for signal compression. Before proceeding with the outline of the algorithm, we contrast LP and Lagrange relaxations. In particular, for MCKP, we will show that both these relaxations give exactly the same duality gaps.

2.3.2.1 Connecting Lagrange and LP relaxations

The LP relaxation of an integer program is obtained by relaxing integrality constraints on integer variables (here, $X_{i,j}$). Due to the constraint *C2* in MCKP we can go ahead and remove upper bounds on these variables as well. Thus, we get

$$\text{Maximize} \quad \sum_{i=1}^{N}\sum_{j=1}^{n_i} p_{i,j} X_{i,j}$$

$$\text{Subject to:} \quad \sum_{i=1}^{N}\sum_{j=1}^{n_i} s_{i,j} X_{i,j} \leq B$$

$$\sum_{j=1}^{n_i} X_{i,j} = 1 \quad \text{for} \quad i=1,2,\ldots,N$$

$$X_{i,j} \geq 0 \quad i=1,2,\ldots,N \quad \text{and} \quad j=1,2,\ldots,n_i$$

Although it may not be feasible for the integer program, a solution to the above certainly gives an upper bound on the objective function of the integral MCKP. Unlike an LP relaxation, in Lagrangian relaxation (as applied to integer problems), we relax certain hard constraints into the objective function using Lagrange multipliers. Since some constraints are removed from the problem, a solution to LR also gives an upper bound on the objective function (for max problems). Unlike LP relaxations, however, integrality constraints on the variables are retained in LR. The resulting relaxed *integer* problem is solved *exactly*. The trick one usually exploits is that certain integer problems are very easy to solve. If the resulting IP can be easily solved, the problem boils down to searching over Lagrange variables to find the set of multipliers that

satisfy the relaxed constraints most closely. We may define three *LRs* for MCKP; i.e., those obtained by relaxing constraints *C1*, *C2*, and both *C1* and *C2*. These are:

$LR1:$

$$\min_{d \geq 0} \{ \max \{ \sum_{i=1}^{N} \sum_{j=1}^{n_i} (p_{i,j} - d s_{i,j}) X_{i,j} + d B \} \text{ s.t. } \sum_{j=1}^{n_i} X_{i,j} = 1 \ \forall \ i \text{ and } X_{i,j} \in \{0,1\} \ \}$$

$LR2:$

$$\min_{l_i \text{ unconstrained}} \{ \max \{ \sum_{i=1}^{N} \sum_{j=1}^{n_i} (p_{i,j} - l_i) X_{i,j} + \sum_{i=1}^{N} l_i \} \text{ s.t. } \sum_{i=1}^{N} \sum_{j=1}^{n_i} s_{i,j} X_{i,j} \leq B \text{ and } X_{i,j} \in \{0,1\} \ \}$$

$LR3:$

$$\min_{l_i \text{ unconstrained}, d \geq 0} \{ \max \{ \sum_{i=1}^{N} \sum_{j=1}^{n_i} (p_{i,j} - l_i - d s_{i,j}) X_{i,j} + \sum_{i=1}^{N} l_i + d B \} \text{ s.t. } X_{i,j} \in \{0,1\} \}$$

For a given d, the solution to the *max* part of *LR1* is to set for each i all $X_{i,j} = 0$ except for the variable corresponding to $j_i^* = \arg\max \ (p_{i,j} - d s_{i,j})$, which is set to 1. Notice that if we were to solve the LP relaxation of the *max* part in *LR1,* there would be n constraints in it. By linear programming [18] we know that an optimal solution to a linear program is a basic feasible solution. The number of positive valued variables in a basic feasible solution is at most the number of constraints in the LP. Hence there may be at most n positive $X_{i,j}$'s for the problem under consideration. But, at least one $X_{i,j}$ must be positive for each i in order to satisfy each constraint; and so an easy counting argument tells us that exactly n $X_{i,j}$'s will be 1 and the rest will be zero in the LP relaxed max problem. Since this solution is integral, it means that integrality constraints in *max* part of *LR1* could have been be dropped (to start with) without any loss of optimality. After dropping integrality constraints from *LR1*, we could equivalently consider the LP-dual for the expression inside *max* part. This would lead to a problem identical to the LP-dual of the original MCKP. Hence, both the LP relaxation and *LR1* will have exactly the same duality gap. Consider next *LR2* and *LR3.* For a given set of Lagrange multipliers $(d \geq 0$, and l_i unconstrained$)$, the solution to *max* part in *LR3* will be to set $X_{i,j}=1$ if $p_{i,j} - l_i - d s_{i,j} > 0$ and zero otherwise. Again, since integrality constraints would have been met in the LP relaxed (*max*) problem, the duality gaps of *LR3* and LP relaxation are the same. For a given l_i (unconstrained) in *LR2,* the problem inside the *max* part is a knapsack problem. Hence, integrality cannot be dropped without a potential loss of optimality. It must further be noted that the complexity of an LR comes both from solving *exactly* the relaxed problem and searching for

the best multipliers. In both *LR2* and *LR3*, we need to search in higher dimensional spaces (*N* and *N+1,* respectively) that are even unconstrained in many dimensions. This complexity is not likely to be less than that of *LR1,* which in turn, guarantees a gap no better than LP relaxation. Therefore, it makes sense to explore LP techniques directly to see if they lead to faster algorithms than LR.

2.3.2.2 Outline for solving the LP relaxation

Consider the LP relaxation that we defined before. Using strong-duality theory of linear programming, the strategy will be to formulate and solve an alternative problem called the LP-dual. The dual of a max problem is a min problem, and is constructed by associating a dual variable for each constraint in the primal and a dual constraint for each variable in the primal problem. Based on whether the primal constraint is equality or inequality, the corresponding dual variable is either free or sign constrained. Further, based on whether a primal variable is free or sign constrained, we have equality or inequality constraints in the dual. There is a close connection between the primal problem and its dual. By *weak duality theorem [18]*, we know that the value of a dual is always more than the value of any feasible primal solution. Strong duality further asserts that *at optimality*, the values of primal and dual problems are the same. The formal statement is as follows:

Strong Duality Theorem [18]: If either the primal LP or the dual LP has a finite optimal solution, then so does the other, and the max of primal problem equals the min of the dual problem. If either problem has an unbounded solution, then the other has no feasible solution.

Let the dual variables corresponding to primal constraints *C1* and *C2* be w and v_i, respectively. The dual of LP relaxed MCKP (call it DL-MCKP) is then given by:

(DL-MCKP) Min. $Bw + \sum_{i=1}^{N} v_i$

s.t. : $ws_{i,j} + v_i \geq p_{i,j}$ or $v_i \geq p_{i,j} - ws_{i,j}$; $w \geq 0$; v_i are not sign constrained

Define, for a fixed w, $f_i(w) = \max_{1 \le j \le n_i} \{p_{i,j} - w s_{i,j}\}$. Notice that since it is the point-wise maximum of linear (convex) functions, the function f_i is also convex in w. With this change of notation, we get:

(DL-MCKP) Min. $F(w) = Bw + \sum_{i=1}^{N} f_i(w)$

s.t.: $w \geq 0$; v_i are not sign constrained

Obviously, the most time consuming part in solving the above problem is the search for LP-dual variable w. This will be discussed in the next section where an $O(n = \sum_{i=1}^{N} n_i)$ algorithm due to Dyer and Zemel [44][165] is outlined. This may well be the best possible complexity achievable for this problem considering that there are n choices to start with. A critical part in the search for w^* is the use of "Dominance Lemma" which lets us eliminate some j's in the search for $f_i(w) = \max_{1 \le j \le n_i} \{p_{ij} - w s_{ij}\}$ in each iteration of the algorithm. The Lemma states that:

Dominance Lemma [44]: If we know that for a certain range of w, $p_{i,j_1} - w s_{i,j_1} \geq p_{i,j_2} - w s_{i,j_2}$, then we can eliminate j_2 in the search for $f_i(w)$ in this range of w (see Figure 2.2).

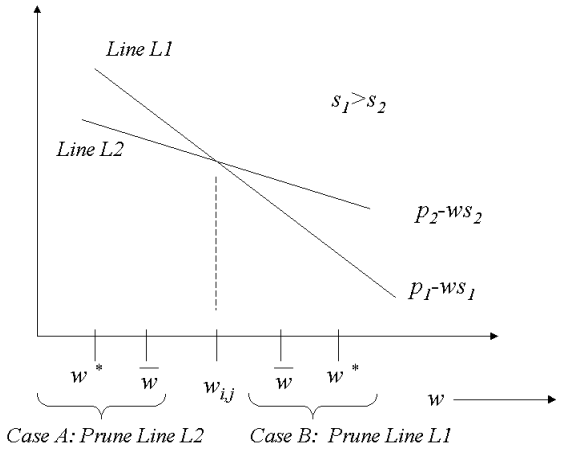


Figure 2.2 Illustration of dominance lemma. Modified from [44].

Assume for now that we have somehow found the optimal $w = w^*$ that solves the dual problem. We will then construct the solution to the primal problem in *O(n)* time using complementary slackness. Complementary slackness conditions relate the primal and dual optimal solutions in a compact way. They state that:

Complementary slackness [18]: Feasible solutions to primal and dual problems are optimal if and only if (i) a variable is zero in one of the problems whenever the corresponding slack variable is strictly positive (i.e., the corresponding inequality constraint is strictly satisfied) in the other problem, and (ii) a slack variable is zero (i.e., the inequality constraint is satisfied as an equality) in one of the problems whenever the corresponding variable is positive in the other problem.

2.3.3 Searching for the optimal LP-dual variable (Dyer [44], Zemel [165])

We now go back to the dual problem and outline a scheme for finding the optimal LP dual variable. The treatment given here follows closely from the one in Dyer [44]. The complete algorithm consists of three parts: initialization, searching for the optimal dual variable, and constructing the optimal primal solution.

2.3.3.1 (STEP 1) Initialization

The min or max of a function in a single variable either lies at one of the boundary points or in the interior (in the latter case first and second order optimality conditions must be satisfied). In the initialization step, we check if the optimal $w \geq 0$ lies at one of the extreme values, $w = 0$ or $w = \infty$.

1. Check if $w^* = \infty$. This will be so if the right gradient *(G_R)* of *F(w)* for large *w* is negative. Intuitively, this means that the bandwidth constraint does not allow accommodating even the least acceptable version of each object. Hence, if $G_R(w \to \infty) = B - \sum_{i=1}^{N} \min_{1 \leq j \leq n_i} \{s_{i,j}\} < 0$, there is no feasible solution to DL-MCKP.

2. Check if $w^*=0$. The optimal $w=0$ if the right gradient of $F(w)\big|_{w=0}$ is positive. Thus, if $G_R(w=0) = B - \sum_{i=1}^{N} \min_{1 \le j \le n_i} \{s_{i,j} \mid f_i(0)=p_{i,j}\} \ge 0$, then $w^*=0$. In this case, we can go directly to Step 3 to find the primal solution. Intuitively, this case means that the bandwidth constraint allows accommodating the most acceptable version of each object.

2.3.3.2 (STEP 2) Search for the optimal *w*

The fundamental theorem of linear programming [18] states that the optimal solution to a linear program, if one exists, lies on a vertex (extreme point) of the polytope defined by the constraints. In the above problem, the objective function is not linear. In fact, it is a piecewise linear and convex function defined by the intersection points of lines (in variable w) of the form $p_{i,j} - ws_{i,j}$. Therefore, using an argument similar to the one used to prove the fundamental theorem of LP, one can argue that the minima of the above piecewise-linear objective function will also lie on an intersection point. There are n_i lines in each set. These lines can intersect in a maximum of $\binom{n_i}{2} = O(n_i^2)$ possible ways. Thus, a brute force approach would have to search over $O(\sum_i n_i^2) = O((\sum_i n_i)^2)$ possibilities.

Instead, Dyer's approach [44] is based on a clever usage of *linear time* median finding algorithms with pruning. There are three steps to search for the optimal *w:* (i) forming pairs of lines of the form $p_{i,j} - ws_{i,j}$ and finding the median of intersection points of these lines, (ii) finding the gradient of the aggregate function $F(w)$ at the median point, (iii) exploiting convexity of $F(w)$ to eliminate some intersection points. In every iteration of these three steps about ¼th of *j's* are eliminated using the dominance lemma. This will eventually lead to a combined *O(n)* complexity of search over all iterations. We initialize the two components (a constant part, *C* and a linear part, *L*) of *F* as *C=0* and *L=B*. The following three steps are then repeated forever:

1. Form Pairs:

For $i=1,...,N$

- If $n_i = 1$, update $C = C + p_{i,j}$ and $L = L - s_{i,j}$. This line need not be considered in any future iteration.
- If $n_i > 1$, form pairs of lines $(p_{i,j} - ws_{i,j1})$ and $(p_{i,j2} - ws_{i,j2})$. Find the intersection point $w_{i,j}$ of these two lines. There can be at most $0.5ën_iû$ such pairs for each i. Overall there are $\sum_{i=1}^{N} 0.5\lfloor n_i \rfloor < 0.5n$ $w_{i,j}$'s.

After iterating over all i find the median of these $w_{i,j}$'s. Call this $\bar{w}$.

2. Find the gradient of F at $\bar{w}$, and using convexity of F, find the side of $\bar{w}$ that has w^*

For every i, the component-wise left and right gradients at $\bar{w}$ are:

$g^i_L(\bar{w}) = -\max_j \{s_{i,j} | \; p_{i,j} - ws_{i,j} = f_i(\bar{w})\}$ and $g^i_R(\bar{w}) = -\min_j \{s_{i,j} | \; p_{i,j} - ws_{i,j} = f_i(\bar{w})\}$

The value of F at $\bar{w}$ is: $F(\bar{w}) = C + \bar{w}L + \sum_{i=1}^{N} f_i(\bar{w})$

And using the above gradients, the slopes of F at $\bar{w}$ are:

$G_L(\bar{w}) = L + \sum_{i=1}^{N} g^i_L(\bar{w})$ and $G_R(\bar{w}) = L + \sum_{i=1}^{N} g^i_R(\bar{w})$

3. Prune out some j's

F is a convex (U) function in w. Hence, based on the sign of $G_L(\bar{w})$ and $G_R(\bar{w})$, one can narrow down the range of w that contains the optimal value w^*. If we are to the left of w^* then $G_L(\bar{w}) \le G_R(\bar{w}) < 0$. Similarly, towards the right of w^*, $0 < G_L(\bar{w}) \le G_R(\bar{w})$. Summarizing these ideas, we get:

- If $G_L(\bar{w}) \le 0 \le G_R(\bar{w})$, then $\bar{w} = w^*$. We can go to Step 3 to find the optimal primal solution.
- If $G_R(\bar{w}) < 0$, then $w^* > \bar{w}$. Therefore, from each pair of lines which intersect at $w_{i,j} \le \bar{w} < w^*$, we may prune out the line with bigger $s_{i,j}$. (See Figure 2.2, case B).
- If $G_L(\bar{w}) > 0$, then $w^* < \bar{w}$. Using the Dominance Lemma, from each pair of lines which intersect at $w_{i,j} \ge \bar{w} > w^*$, we may prune out the line with smaller $s_{i,j}$. (See Figure 2.2, case A).

Notice that in all at least $\left\lceil 0.5 \sum_{i=1}^{N} \lfloor 0.5 n_i \rfloor \right\rceil \ge n/6$ *j's* will get pruned out in each iteration. Thus, the overall complexity is $O(\sum_{i=1}^{\infty} (5/6)^i n) = O(n)$.

2.3.3.3 (STEP 3) Solution to the primal

The relation between the optimal solution to the primal and dual programs is specified by complementary slackness conditions given above. The dual variables corresponding to primal constraints *C1* and *C2* are w and v_i, respectively. Likewise, corresponding to dual constraint $ws_{i,j} + v_i \ge p_{i,j}$ we have primal variables $X_{i,j}$. From complementary slackness, we know that if a dual variable is not zero, there should be no slack in the corresponding primal constraint. Similarly, if there is any slack in a dual constraint, the corresponding primal variable is zero. For each i, the optimal dual variables $u_i^* = \max_{1 \le j \le n_i} \{p_{ij} - w^* s_{ij}\}$. Therefore the recipe to construct optimal primal solution is the following:

- For all i except for the set that determines the optimal dual variable w^*, find the j that determines the max in $u_i^* = \max_{1 \le j \le n_i} \{p_{ij} - w^* s_{ij}\}$. For the corresponding $X_{i,j}$, there is no slack in the dual constraint, i.e., $ws_{i,j} + v_i = p_{i,j}$. Hence this $X_{i,j} = 1$, and $X_{i,j}$'s corresponding to other j's in this set are 0.
- For the set that determines the optimal dual variable w^*, there are two lines that attain the max in $u_i^* = \max_{1 \le j \le n_i} \{p_{ij} - w^* s_{ij}\}$. Thus there can be two fractional variables for this set. To find

these fractional values, we apply complementary slackness to the dual variable *w,* and its primal constraint *C1*. If $w^*=0$, then there is slack in constraint *C1*. We set $X_{i,j} = 1$ for the line having higher $p_{i,j}$. Recollect that this is the scenario where the most preferred version of each object can be included in the presentation. If, on the other hand, $w^*>0$ there are exactly two fractional $X_{i,j}$'s, and we wish to find values they attain. Denote by i^* the set under consideration and by j_1 and j_2 are the lines that attain $u_{i^*}^*$. Thus,

$$\sum_{i=1}^{N}\sum_{j=1}^{n_i} s_{i,j}X_{i,j} = \sum_{i=1,i\neq i^*}^{N}\sum_{j=1}^{n_i} s_{i,j}X_{i,j} + s_{i^*,j_1}X_{i^*,j_1} + s_{i^*,j_2}(1-X_{i^*,j_1}) = B$$

Solving, $X_{i^*,j_1} = \dfrac{B - \sum_{i=1,i\neq i^*}^{N}\sum_{j=1}^{n_i} s_{i,j}X_{i,j} - s_{i^*,j_2}}{s_{i^*,j_1} - s_{i^*,j_2}}$, and

$X_{i^*,j_2} = 1 - X_{i^*,j_1}$

2.3.4 Features and limitations of LP relaxations

Let us reminisce the basic features and limitations of the approach taken. These are:

- The numbers p_{ij} and s_{ij} are arbitrary. We make no assumptions about convexity of R-D curves for each *i.* There is no need to obtain or estimate the entire rate-distortion curve. The solution we obtain gives the best possible selection from data points that have been collected.
- PSNR's and rates are usually considered to form a continuum. But, a typical motivation for an MCKP-like formulation could be when the objects have been pre-encoded at various quality levels and there is an inventory of candidate objects.
- The LP-dual approach can be used if the objective function and the constraints are linear. Otherwise, the objective function and the constraints must be linearized (if possible). Alternatively said, solving the LP relaxation (of an IP) is useful to approximate a non-linear R-D function using piecewise linear segments between points on the curve.
- The LP relaxation (like LR) does not require finding convex hull of the R-D points. This circumvents an obligatory $O(\sum n_i \log(n_i))$ complexity for calculating the hull.
- It must be noted that an LP relaxation (again, like Lagrange relaxations) gives an upper bound on the objective function. Hence, it is *not* a heuristic in the real sense of the word. A heuristic gives a solution that is *feasible*, but *sub-optimal*. Thus, heuristics provide a lower

bound on the optimal objective function value for a max problem. A particularly attractive feature about MCKP is that its LP relaxation can have at most two fractional variables in the optimal solution. The proof of the above statement follows a counting argument like the one used before. Firstly, in LP relaxed MCKP, the size of the basis is *N+1* since there are a total of *N+1* constraints. Therefore, at most *N+1* variables can be non-negative in the basic feasible solution. Secondly, the second constraint (in MCKP) implies that at least one version of an object must be selected from each set. Each fractional variable implies that there exists at least one more fractional variable in the same constraint. Therefore, at most one set can have two fractional values in an optimal basic feasible solution (BFS). This can be exploited to generate heuristics from upper bounds. In order to generate a lower bound (and hence a heuristic), we deterministically set one of the two fractional values to one, and the other to zero. The selection is done so that constraint *C1* is not violated.

- There are some key similarities and differences between the approach here and that in Shoham's paper [144]. First, Shoham's approach was based on LR as applied to integer programs. As shown above, for the problem under consideration, the LR solution has the same duality gap as the one obtained with LP relaxation. Second, although the approach in [144] used continuous quantizer functions, they were defined over discrete variables. Thus, the problem considered here and in Shoham's work is the same if one considers a fine discretization of the quantizer function for each quantizer and linearizes the resulting problem. Needless to say, the approach here does not assume any specific discretization structure or bounds on parameters $s_{i,j}$ or $p_{i,j.}$ Third, Shoham's solution scheme uses singular points to search for the optimal Lagrange multiplier. These singular points correspond exactly to intersection points of lines considered above. Hence, the base algorithm of [144] has to look over $O(\sum_i n_i^2) = O((\sum_i n_i)^2)$ points. Variant 1 in [144] does indeed consider pruning to reduce complexity, as is done above also. But, dominance considerations were not systematically exploited to attain a linear-time bound. Another key difference here is the usage of (linear time) median finding algorithms that avoid any sorting steps.

This concludes the treatment for modeling of scalable objects. In the next section, we discuss how to impose logical inter-dependencies among objects and show how certain non-linearities can be easily linearized.

2.4 Inter-dependencies and non-linearities

Inter-dependencies usually exist between objects that are to be included in a presentation. These dependencies may arise due to synchronization or compression constraints, or be entirely perceptual user preferences. We address two types of logical dependencies here, namely those due to (a) If-then, and (b) Either-Or constraints[1]. Integer programming is well equipped to handle both these types of logical implications. It is also interesting to note that MCKP is essentially a very special form of inter-dependency among objects – one in which exactly one version from a candidate set of objects is selected.

Several inter-object dependencies may be modeled as if-then constructs. For example, imagine a scene with a weatherman, a map, a caption, and some scrolling text. In such a scenario, it may be imperative that if the weatherman is included in the scene, the map is too. As another example, a user may be interested in seeing both a video with associated caption or none. Furthermore, if we include a video stream it may be imperative to multiplex the associated audio. All of the above can be written as if-then relations among some Boolean variables. Several other kinds of dependencies can be modeled as either-or constraints among objects. For example, in his/her personalized presentation, a user may be interested in hearing at least one of two audio tracks or in seeing a caption in at least one of many possible languages. Next, we discuss how to specify the above forms of logical implications by adding constraints to the above optimization framework.

[1] In principle, we can express any Boolean relation using AND and OR operations. An AND is equivalent to two *if-then* statements and an OR to one *either-or* construct. Although we focus on these two constructs exclusively, we observe that many relationships among objects (e.g., those due to spatial or temporal layout) are better not expressed as *logical* constraints. Which formulations to express these relationships are more intuitive, which are easier to solve, which do a better job of modeling etc. are certainly open questions.

2.4.1 If-Then constructs

Assume that the pair under consideration is (O_i, O_j) and it is required that if O_i is included, so is O_j. Notice that the above relation is non-commutative. Thus it is not equivalent to solve a modified problem with a joint (O_i,O_j) object having a combined quality of (p_i+p_j) and bit-rate of (s_i+s_j). However, we can enforce this *logical* requirement by adding the following constraint:

$$X_j - X_i \geq 0$$

By adding this constraint the only valid (X_i, X_j) combinations are $(0,0)$, $(0,1)$, and $(1,1)$ and the combination $(1,0)$ is ruled out. Consider another example in which we require that if O_i is included then O_j may not be included. Adding the following constraint can satisfy this:

$$X_j + X_i \leq 1$$

Note that $O_i \Rightarrow Not(O_j)$ is equivalent to $O_j \Rightarrow Not(O_i)$. The allowed (X_i, X_j) combinations for either of these expressions or the above constraint are $(0,0)$, $(0,1)$, and $(1,0)$ and the combination $(1,1)$ is ruled out. In general, if $\psi(X_1,X_2,\ldots,X_n) > 0$ implies $\Phi(X_1,X_2,\ldots,X_n) \geq 0$, we may add

$$-\Phi(X_1,X_2,\ldots,X_n) \leq My$$

$$\Psi(X_1,X_2,\ldots,X_n) \leq M(1-y)$$

$$y \in \{0,1\}$$

to the problem. Here M is a large enough number such that $\Psi(X_1,X_2,\ldots,X_n) \leq M$ and $-\Phi(X_1,X_2,\ldots,X_n) \leq M$ is true for all possible $X_1,X_2,\ldots,X_n$. It is easy to see the proof. If $\Psi > 0$, then from the second inequality we have $y=0$; which from the first one implies that $\Phi \geq 0$. Next, if $\Psi \leq 0$, we can have both $y=0$ and $y=1$. If we let $y=1$, we make sure that we do not restrict Φ in any way.

2.4.2 Either-Or constructs

Like in (1) above, we may have situations where certain either-or type of logical implications must be satisfied. Consider an example where we have the pair (O_i,O_j) and require that at least one of O_i or O_j be included in the presentation. We may impose this by adding the constraint $X_i + X_j \geq 1$. In general, we may want that at least one of the following two inequalities be true:

$$\Phi(X_1,X_2,\ldots,X_n) \leq 0$$

$\Psi(X_1,X_2,\ldots,X_n) \pounds 0$

For this we could add

$\Phi(X_1,X_2,\ldots,X_n) \pounds M(1-y)$

$\Psi(X_1,X_2,\ldots,X_n) \pounds My$

$y \hat{I} \{0,1\}$

to the problem. Again, M is a large enough number such that $\Psi(X_1,X_2,\ldots,X_n) \pounds M$ and $\Phi(X_1,X_2,\ldots,X_n) \pounds M$ is true for all possible $X_1,X_2,\ldots,X_n$. The proof here is in line with the one we gave for *(2.4.1)*.

The functions Φ and Ψ above are functions in *Boolean* variables $X_1,X_2,\ldots,X_n$. An OR operation among Boolean variables can be represented as a linear constraint. However, an AND operation in these variables (or their negations) will lead to non-linear terms. A typical non-linear term could be of the form: $X_1 \bullet X_2 \bullet \ldots \bullet X_n$. A term like this (either in the objective function or in one of the constraints) can be easily linearized by adding extra variables and constraints. We present two such linearization schemes[2]. To discuss the first one let $W_n = X_1 \bullet X_2 \bullet \ldots \bullet X_n$. Since $X_i\big|_{i=1,..,n}$ are binary, so is W_n. But, we must ensure that W_n=1 only when all $X_i\big|_{i=1,..,n}$=1 and is zero otherwise. For this, we add constraints:

$$W_n \geq \sum_{i=1}^{n} X_i - n + 1$$

$$W_n \leq \frac{\sum_{i=1}^{n} X_i}{n}$$

If all $X_i\big|_{i=1,..,n}$=1, then the first constraint implies that $W_n \geq 1$; while the second says that $W_n \leq 1$. Together they both imply that $W_n = 1$. If any of $X_i\big|_{i=1,..,n} \neq 1$, then the first constraint is not

[2] Both the transformations we discuss linearize *polynomial* non-linearities. They cannot be applied if Φ and Ψ have *non-polynomial* non-linearities. There do exist direct schemes (such as generalized penalty and interior penalty methods) that may be used for solving such non-linear integer programs without linearization. A discussion of these and their computational efficiency is beyond the scope of this work.

restrictive in any way. But, the second one implies that $W_n < 1$, or that $W_n = 0$. Together, the two constraints imply that that W_n=1 only when all $X_i\big|_{i=1,..,n}$=1 and is zero otherwise.

We can linearize polynomial non-linearities of the form above using an alternate method. Let us define a sequence of variables $Z_0, Z_1, ..., Z_n$ as follows:

$Z_n = Z_{n-1} \bullet X_n,\ Z_{n-1} = Z_{n-2} \bullet X_{n-1}, ..., Z_1 = Z_0 \bullet X_1,\ Z_0 = 1$

It is obvious that $Z_n = X_1 \bullet X_2 \bullet ... \bullet X_n$. Also since $Z_i = Z_{i-1} \bullet X_i$, we must add the following *linear* constraints for $i = 1,...,n$:

$Z_i \le Z_{i-1}$

$Z_i \le X_i$

$Z_i + 1 \ge X_i + Z_{i-1}$

$Z_i \in \{0,1\}$

Of course, we also add: $Z_0 = 1$.

The first two constraints enforce that Z_i is not 1 unless both X_i and Z_{i-1} are 1. The third constraint ensures that $Z_i=1$ if $X_i=1$ and $Z_{i-1}=1$ and does not restrict any of the other possibilities. All the above constraints are linear, and the entire non-linearity gets captured in Z_n.

Note that the second transformation needs many more constraints and variables as compared to the first one. This is not entirely wasteful. By deriving the constraints for the first transformation using the inequalities for the second one, we can easily show that if there is a point in the second polyhedron it also lies inside the first one. In other words, the second polyhedron is smaller and hence packs the integral points more tightly. Thus, if we use an iterative scheme such as branch and bound, the second problem is expected to converge much faster.

After we do the aforementioned modifications in *(2.4.1)* or *(2.4.2)* above and remove any non-linearities that may arise, the new problem is still a (linear) IP. Therefore, at least in theory, we may find the optimal solution using the branch and bound method described in Appendix A.1. Unfortunately, branch and bound is very computationally intensive[3]. We give a brief

[3] We believe that it will be challenging to find fast approximations after adding the discussed logical constraints. It is unlikely that polynomial time approximations can be found for very generic functions y and F.

computational experience in Section 2.7. An interesting point to note here is that additional constraints (e.g., those due to *if-then* and *either-or* constructs) actually reduce the search space of the problem. Hence, from an *enumeration perspective*, the complexity of the problem gets reduced.

2.4.3 Non-linearities in the objective function

As another usage of linearization, this section discusses two common non-linearities in objective functions, both of which can be easily linearized. Sum of distortions (or PSNR's) has been the most popular criterion for solving bit allocation problems. Despite having a nice separable structure to it, the criterion in itself is not the most appropriate one for image and video processing applications. This is because min-sum or max-sum measures can lead to an optimal solution that has regions of large localized distortion. One could also get a solution having sharp changes in quality across various parts of a scene. Although both these may make small statistical contributions, the human visual system is very sensitive in picking them up. Having a *fairness* measure is one way to avoid these problems. We discuss below two fairness measures for an MCKP-like problem – *ABS_DEV* and Maxmin. Both of these introduce easily linearizable non-linearities. The first criterion tries to find an allocation for which there is least absolute deviation in quality around the mean PSNR. The second one imposes fairness by trying to maximize the minimal PSNR in a frame. More formally, the two problems are:

$$Abs\_Dev: \text{Min} \sum_{i=1}^{N}\sum_{j=1}^{n_i}\left|(p_{i,j}-\bar{p})X_{i,j}\right|$$
$$\text{s.t. } (C1),\ (C2),\ (C3)$$
$$\text{where } \bar{p} = 1/N \sum_{i=1}^{N}\sum_{j=1}^{n_i} p_{i,j}X_{i,j}$$
$$MaxMin: \ \text{Max} \min_{i,j} \{p_{i,j}X_{i,j}\}$$
$$\text{s.t.: } (C1),\ (C2),\ (C3)$$

Both the above have non-linear objective functions that can be easily linearized. For the former, we introduce extra variables defined as follows:

$e^{+}_{i,j} = \max\{p_{i,j} - \bar{p}, 0\}$ and $e^{-}_{i,j} = \max\{\bar{p} - p_{i,j}, 0\}$

Thus, *Abs _ Dev* becomes,

$$\text{Min} \sum_{i=1}^{N} \sum_{j=1}^{n_i} (e^{+}_{i,j} + e^{-}_{i,j}) X_{i,j}$$

s.t.: (C1), (C2), (C3), and

$(p_{i,j} - \bar{p}) X_{i,j} = e^{+}_{i,j} X_{i,j} - e^{-}_{i,j} X_{i,j}$ and $e^{+}_{i,j}$, $e^{-}_{i,j} \geq 0$ and real $\forall i, j$

Note that we are trying to minimize the sum of two deviation terms $(e^{+}_{i,j} + e^{-}_{i,j}) X_{i,j}$, both of which are non-negative. This fact together with the added constraints implies that exactly one of these two variables is non-zero for a given *i, j*. Hence, we need not add non-linear constraints of the form $e^{+}_{i,j} \times e^{-}_{i,j} = 0 \ \forall i, j$.

The latter problem (MaxMin) is even easier to linearize. Let, $J = \min_{i,j} \{p_{i,j} X_{i,j}\}$. Thus, MaxMin becomes,

Max J

s.t.: $(C1)$, $(C2)$, $(C3)$, and

$p_{i,j} X_{i,j} \geq J \ \forall i, j$ and $J \geq 0$ real.

One could also combine the above two criteria to come up with other fairness measures that are easily linearizable. An obvious one is: Min $\underset{\text{i,j}}{\text{Max}} \left| p_{i,j} X_{i,j} - \bar{p} \right|$. After the above-suggested linearizations one could resort to a branch and bound method to find the optimal solution.

This concludes the treatment for dependent objects and some linearizable non-linearities. In the next section, we suggest a transformation that exploits object aggregation and helps to reduce problem complexity.

2.5 Object aggregation

Several applications benefit from grouping, aggregating or multiplexing objects [76]. Application level multiplexing allows one to guarantee integrated QoS for streams multiplexed in a common channel, and differentiated quality to different groups of users based on a pricing

model. Multiplexing objects helps to reduce the number of transport channels that need to be managed. The overhead in packet headers is also usually lower with multiplexed streams than with headers for each separate stream. Grouping or multiplexing content is again important if content from different sources needs to be delivered to a common destination (or vice-versa, as in a multicast scenario). To address some of the above issues, MPEG-4 provides a flexible multiplexing facility at the application layer. In MPEG-4, audio-visual objects are encoded in separate elementary streams. The content creator multiplexes elementary streams along with scene description information. Each object is associated with an object descriptor (OD) that specifies the elementary streams from which this object gets its data. Elementary streams, in turn, are divided into multiple Access Unit Layer (or, AL) PDUs. Below the AL-PDU layer, two levels of multiplexing, called TransMux and FlexMux are identified. The former is not specified by MPEG-4, and it is assumed that the underlying transport protocols provide it. Unlike TransMux, FlexMux lies much closer to the application and therefore one can use content properties to make better decisions about object selection, dropping and scheduling. FlexMux has two modes to flexibly multiplex objects -- a Simple Mode and a MuxCode Mode. The basic unit of the FlexMux layer is a FlexMux-PDU, and a sequence of FlexMux PDUs is called a FlexMux stream. In the Simple Mode, data from only one object may be included in a FlexMux-PDU. In the MuxCode Mode, however, data from multiple objects can be encapsulated in a single FlexMux-PDU.

From an optimization perspective, in this section, we suggest a transformation that exploits object aggregation and helps in reducing the complexity of optimal object selection. It is easy to see that we can still solve the problem using a separate variable for each of the (say, b_i) objects that have same p_i vs. s_i tradeoff. However, by treating all objects having same R-D tradeoff as a single object (an aggregation procedure), we can decrease the number of variables and reduce problem complexity. Instead of the constraint $X_i \hat{I} \{0,1\}$ (as in the original problem), we shall now use a constraint $X_i \hat{I} \{0,1,\ldots,b_i\}$ for $i=1,2,..n$. In essence, we treat all similar objects as a single object and allow multiple copies of this object. For each aggregate object X_i with an upper bound b_i, we can use a transformation that adds only $\lceil \log_2 b_i \rceil$ variables per aggregation. The new objects (variables) are assumed to have PSNRs of $p_i, 2p_i, 4p_i \ldots$ and bitrates of $s_i, 2s_i, 4s_i \ldots$ etc. A last object is added to ensure that total quality is $b_i p_i$ and total bitrate is $b_i s_i$. Since we

can represent any number between $0,...,b_i$ using $\lceil \log_2 b_i \rceil$ bits, only $\lceil \log_2 b_i \rceil$ (and not b_i) variables are needed. The optimal solution with these new objects will give the binary representation of the number of objects of type X_i that should be selected.

2.6 An online heuristic

Until now, we have described a form of problem where the entire input is available to start with. Given the entire input set, we discussed ways to solve the problem optimally and approximately. However, in several applications, the input is made available in "*increments*" only. For example, in a multiplexing scenario, we could have a set of N streams that have been optimally allocated resources to suit their QoS. Next, if a new stream enters service with its own R-D tradeoff, and we wish to do allocation, can we do something better than re-solving the size $N+1$ problem? Another application could be post-authoring editing of a multimedia presentation. We assume that the original scene has been authored with resources optimally partitioned among objects. Due to editing alterations, these resources may need to be re-assigned with minor changes. Yet another application could be an MPEG-4 based system in which user interaction alters scene rendition. Objects may be made to appear or disappear, their enhancement layers could be activated or deactivated, etc. Here also, a minor modification to the solution set $\{X_{i,j}\}$ could yield good approximations.

As we discussed above, a brute force approach would be to solve the larger problem without any prior knowledge. However, we know the best solution for size N problem to start with. Therefore, it makes more sense to use the "marginal analysis" approach of Fox [56]. For an alternative way to model and incrementally solve problems such as the one discussed here, see [86][87]. The performance of marginal analysis approach depends on starting with a good feasible solution. In the worst case, its complexity is similar to that of dynamic programming, but is usually much better if the initial feasible solution guess is good. Assume that the solution to the size N problem is $\{(p_{i,ji^*}, s_{i,ji^*}) \mid i=1,2,..,N\}$ [4]. To find solution to the problem with another added object $(p_{N+1,j}, s_{N+1,j})$ $j=1,...,n_{N+1}$, re-index (or eliminate some) j's such that the following two hold:

[4] We give an illustration for object addition. It is easy to modify it for object removal.

(a) If $p_{i,j1} < p_{i,j2}$ then $s_{i,j1} < s_{i,j2}$ for all $j_1, j_2=1,2,..,n_i$ and $i=1,..,N+1$

(b) For each $i=1,2,...,N+1$, if $s_{i,j1} < s_{i,j2} < s_{i,j3}$, then $\frac{p_{i,j2} - p_{i,j1}}{s_{i,j2} - s_{i,j1}} \geq \frac{p_{i,j3} - p_{i,j2}}{s_{i,j3} - s_{i,j2}}$ for any $j_1,j_2,j_3 \in \{1,2,..,n_i\}$. Note that marginal utility decreases after re-indexing, and we have effectively identified the convex hull for each set i. This is a computationally intensive procedure requiring a minimum workload of $O(\sum_{i=1}^{N+1} n_i \log(n_i))$. However, one can do this in the idle time between successive object additions or deletions. Note that we need not re-index all objects each time the online algorithm is run. Since convex hulls for earlier objects should already be available, we have to identify the convex hull for each new object only.

After making above-mentioned changes, we apply a marginal allocation algorithm. The algorithm is expected to run fast since it starts with the optimal size N solution. We expect that these objects will not be perturbed too much around their starting allocations. Yet another way to visualize this is as an increase in the number of stages of the trellis by 1. To accommodate the new stage, current allocations for first N stages are decreased. Initialize $p_{N+1,j^*}=s_{N+1,j^*}=0$. The other objects start with their size N optimal solution, i.e., $\{(p_{i,ji^*}, s_{i,ji^*}) \mid i=1,2,..,N\}$. Obviously, this is a feasible solution to the size $N+1$ problem. There are n_{N+1} iterations in the procedure. The algorithm decreases in each iteration the allocation for one of the objects from $i \in \{1,2,..,N\}$. Assume that i^* is the index for this object. The allocation of the i^*th object is reduced (by decreasing its index j) and that of object $N+1$ is increased incrementally (i.e., to next higher j). The index i^* is selected from the set $\{1,2,...,N\}$ such that it gives the smallest decrease in marginal quality per unit increase in bandwidth. Due to a discrete nature of the problem, we must ensure that re-allocation does not cause bound B to be exceeded. Any residual bandwidth after re-allocation is kept for future iterations. If, however, the resource bound B will get exceeded if we do re-allocation using index i^*, we relinquish this index and continue our search for another index (a new i^*) in decreasing order of marginal utility until a choice is found that leaves some non-negative residual bitrate after re-allocation. The method terminates if one of the following two hold: (a) the current increase in marginal quality for object $N+1$ is less than any possible decrease in marginal utility among objects $i \hat{I} \{1,2,...,N\}$, or (b) if no change can be done without violating the bandwidth bound B.

2.7 Experimental Comparison

In this section we discuss a brief experimental study on some of the formulations presented above. First, we consider scalable objects without any extra dependencies. We will compare the complexity and solution qualities obtained by exact schemes (like Branch and Bound) with approximate ones (e.g., by solving the LP-relaxation of the IP). Second, we consider more general dependency structures. We will discuss if a combinatorial scheme like Branch and Bound is a practical one; and if for problems with more general dependency structure, the solutions obtained by an LP relaxation is a reasonable approximation.

The scalable texture coding [149] used for synthetic and natural objects in MPEG-4 is used to construct the R-D data set. The underlying algorithm in this codec is based on a multi-scale zero-tree wavelet entropy coding technique. This gives many scalability layers in terms of both spatial resolutions and SNR (or quality). A total of 37 objects are assumed to compose the scene. These objects include instances from standard test sequences like Akiyo, Stefan, birth, coastguard, dance etc. Each of these was encoded into 13 distinct scalability levels, one of which must be chosen for each object. Thus, an exhaustive search for the optimal solution would need to look at a staggering 13^{37} combinations. This is clearly impractical. Below, we tabulate the performance of some of the algorithms described above and show substantial improvements. We ran simulations on 5 problem instances, both for standard modeling of scalable objects and the problem with extra inter-object dependencies. We used LP and Mixed IP solutions of IBM's optimization solutions software package (OSL). The LP product comprises routines for solving linear programs using Simplex (primal and dual) and interior point methods. The mixed IP product uses branch and bound method to solve linear integer programs. The simulations were conducted on a Pentium© 233 MHz PC with 64MB RAM running the Windows 98© operating system.

2.7.1 Without Dependencies

The problem without dependencies is one that has an MCKP formulation. Let G1 be defined as the percentage difference between the optimal IP solution and the LP heuristic solution obtained

by the rounding scheme described in Section 2.3.4. G2 is the normally used duality gap and is defined as the percentage difference between the optimal IP solution and its LP relaxation. Our results are summarized in Table 2.1 below. In this table, the (P) and (D) for branch and bound refer to primal and dual Simplex methods, respectively, that are used to solve intermediate LP relaxations in branch and bound. The LP relaxation was solved using the OSL product. It was set to solve the LP-dual of the integer problem (using dual simplex method). From the results we note that:

- LP relaxations usually give very small duality gaps. Both the LP solution and the heuristic generated from it are very close to the optimal IP solution. The marginal analysis has higher complexity but give similar gaps similar to the LP-dual approach.

- Solving the LP dual problem leads to a tremendous decrease in computational complexity relative to an exhaustive search scheme like branch and bound. The corresponding loss in optimality is very small.

2.7.2 With Dependencies

We modified the above problem by adding if-then constraints to create a problem with dependencies. The dependency constraints were added to ensure there are no sharp quality gradations among closely related objects. We did this by dividing alternate versions of some of the objects into two sets: a low quality set and a higher quality set. Each of these sets can have several versions (or scales) of the same object. The added *if-then* dependency constraints ensure that (all) related objects be chosen from the same respective set. I.e., the selected version of *all* inter-related objects comes from either their lower quality set or from their higher quality set. To some extent this would ensure that closely related objects have similar quality scales. In all, a total of 32 extra constraints were added to the problem. The results are summarized in Table 2.2 below. From this table we note that:

- Although Branch and Bound is exponentially complex, for the problem sizes under consideration, it always completed in less than 30 seconds. Note further that (typically) the

time taken with extra inter-object dependencies is lower than the corresponding time without dependencies. This could be because by adding some additional constraints, one could shrink down the feasible polyhedron, and make the problem simpler from an enumeration point of view.

- For the problem sizes under consideration, solving the linear relaxation using the Simplex method of OSL is extremely quick[5]. The resulting duality gap G2 is usually very small.

- The number of fractional variables in the optimal LP solution is typically very small. Out of a total of 497 variables, (normally) only 8-11 were fractional in the optimal LP solution. For very general dependency structures, we were not able to devise a systematic way (such as deterministic or randomized rounding) to find sub-optimal feasible IP solutions using the fractional solution obtained. However, we were able to exhaustively search through all possible ways of rounding the fractional variables after the variables that attained a 0 or 1 in the LP solution were frozen at their respective values. In all the cases considered, this could find feasible (but slightly sub-optimal) IP solutions with minimal extra computational effort.

	Algorithm	Time Taken (sec)	Duality Gap (%)
Instance 1	Branch and Bound	26.682 (P) and 30.826 (D)	0
	LP relaxation (LP dual)	1.184	0.5462 (G1) and 0.0527 (G2)
Instance 2	Branch and Bound	30.924 (P) and 17.928 (D)	0
	LP relaxation (LP dual)	1.232	0.3239 (G1) and 0.0288 (G2)
Instance 3	Branch and Bound	8.944 (P) and 7.952 (D)	0
	LP relaxation (LP dual)	1.228	0.1062 (G1) and 0.0843 (G2)
Instance 4	Branch and Bound	6.07 (P) and 5.821 (D)	0
	LP relaxation (LP dual)	1.264	0.0496 (G1) and 0.0550 (G2)
Instance 5	Branch and Bound	1.548 (P) and 2.24 (D)	0
	LP relaxation (LP dual)	1.236	0 (G1) and 0 (G2)

Table 2.1: Performance without dependencies

[5] This software was set to choose between the primal and dual Simplex approaches internally.

	Algorithm	Time Taken (sec.)	Duality Gap (%)	# Fractional Vars.
Instance 1	Branch and Bound	7.162 (P) and 4.19 (D)	0	0
	Simplex method (OSL)	1.125	0.058 (G2)	11
Instance 2	Branch and Bound	10.356 (P) and 25.97 (D)	0	0
	Simplex method (OSL)	1.232	0.139 (G2)	9
Instance 3	Branch and Bound	9.418 (P) and 6.682 (D)	0	0
	Simplex method (OSL)	1.12	0.114 (G2)	11
Instance 4	Branch and Bound	6.82 (P) and 4.49 (D)	0	0
	Simplex method (OSL)	1.108	0.055 (G2)	10
Instance 5	Branch and Bound	1.582 (P) and 2.778 (D)	0	0
	Simplex method (OSL)	1.367	0 (G2)	8

Table 2.2: Performance with dependencies

2.8 Concluding Remarks

In this chapter, we proposed mathematical formulations for modeling object-based scalability and some functionalities that it brings with it. Some of the techniques discussed in this work have been known in the OR literature for some time. But, we believe that this framework applies quite well for modeling in MPEG-4 and other structured multimedia systems. It also helps us improve our understanding of the kinds of problems that are likely to arise in multiplexing, authoring and content filtering systems based on structured content. This may help to better design and semi-automate tools that users will use to create and interact with media. On a more rigorous front, the MCKP approach to model scalability helped us identify a faster algorithm for the much-researched operational bit-allocation problem. This was based on solving the LP relaxation using dual descent.

Recollecting, we discussed the following issues in this chapter:

- We modeled object-based scalability as a *"knapsack problem"* and discussed some known schemes to find optimal and approximately optimal object selections. This formulation is useful for modeling value-added services.
- We generalized the above to model authoring or multiplexing of scalable objects (e.g., objects encoded at various target bit-rates) using the *"multiple choice knapsack problem"*.

This problem was related to many known problems in video coding literature – the most prominent one being the bit-allocation problem.

- Almost all earlier work to solve the bit-allocation problem is based on Lagrangian relaxation. Instead, we discussed how one could alternatively solve the linear relaxation to give similar duality gaps. The LP relaxation was solved using strong duality with dual descent – a linear time process. We showed that there can be at most two fractional variables in the optimal primal solution. Therefore this relaxation can be justified for many practical applications and could also be used to generate good lower bounds (feasible solutions or heuristics). This work reduces problem complexity, guarantees similar performance, is slightly more generic, and provides an alternate LP-duality based proof for earlier work by Shoham and Gersho [144].
- We gave additional constraints that may be added to enforce logical *inter-dependencies* among objects in a presentation. Some simple linearization schemes to remove polynomial non-linearities and make the problem amenable to generic linear-IP schemes were suggested.
- A transformation that exploits object aggregation and helps in reducing problem complexity was discussed. Finally, we suggested the marginal analysis approach of Fox as a method to do re-allocation with *incremental inputs*. It is efficient in re-optimizing the allocation when a system has online user interactivity, appearing or disappearing objects, time driven events etc.

We believe that fairly general scenarios may be envisaged using combinations of the basic blocks that we outline above. Further, we hope that some of the optimization techniques used here will be used extensively to solve other problems in signal compression.

This chapter dealt with modeling issues related to MPEG-4 systems. These may aid in transcoding (structured) content once it has been authored. Most of these algorithms are likely to reside at the content server or an intermediate proxy. The next chapter deals with flexibilities that exist at the client side. We will present models and algorithms for adaptive playout control and discuss an alternative (implicit) way of modeling resource constraints.

CHAPTER 3

A framework for optimal scheduling of structured and streaming media

Contents

3.1 Introduction

Future multimedia services are envisioned to run in distributed environments, involving collaborative content creation and editing, intelligent transport mechanisms, and subsequent presentation (composition) of structured multimedia information. The migration from a stream-based compression and manipulation scenario to a structured object-based one has opened new challenges for research in resource-constrained retrieval, temporal synchronization, real-time playback and creation of multi-modal composite documents. Structured multimedia documents comprise both continuous and non-isochronous media. Although scheduling and retrieval of both of these forms of content are challenging tasks, their service requirements are somewhat different and need to be specified and satisfied in different ways. Moreover, the transmission of documents introduces delays due to buffering, packet losses, error recovery schemes, segmentation and re-assembly etc. These delays are fairly unpredictable and introduce skew or jitter in the presentation. Skews and inter-object jitter occur in almost every form of content delivery and consumption; however, most temporal specification and modeling schemes tend to be somewhat rigid in their requirements. An effort therefore is also needed in relating the closely coupled multiplex and playout scheduling issues with a relaxed temporal specification.

Both client and server side decisions have an effect on media scheduling. The server side problem may be summarized as:
Objective: To find a selection and ordering of objects to be multiplexed in the bitstream.
Constraints: Interface buffer bounds, channel capacity, interdependencies among objects, and intended deadlines at which objects need to be displayed.
Freedoms: Due to different media modalities and pre-specified deadlines, the perception of missed deadlines for objects differ.

Unlike the server, scheduling issues at a client (see Figure 3.1(b)) relate to (a) media rendering, processor sharing for decoding multiple objects and its playout control, and (b) finding optimal retrieval schedules. The constraints and freedoms stay the same as with the server side. Note further that if we ignore channel variations, any multiplexing operations done at the server are replicated at the client end. Therefore, it is intuitive that a common modeling and decision framework works at both ends.

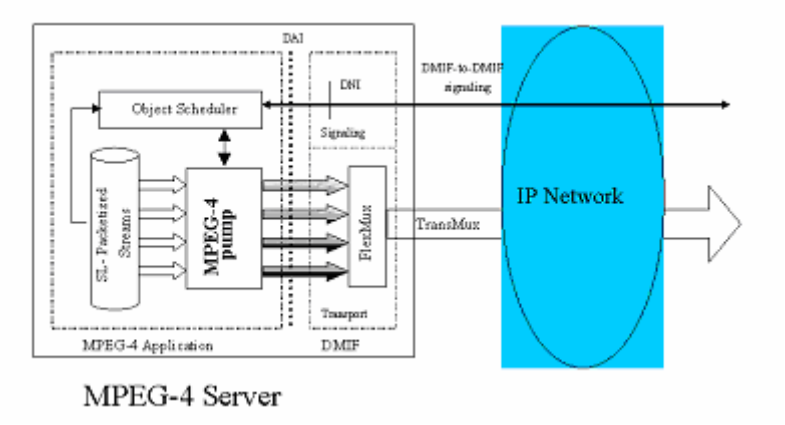


Figure 3.1 (a) System Model: Server architecture

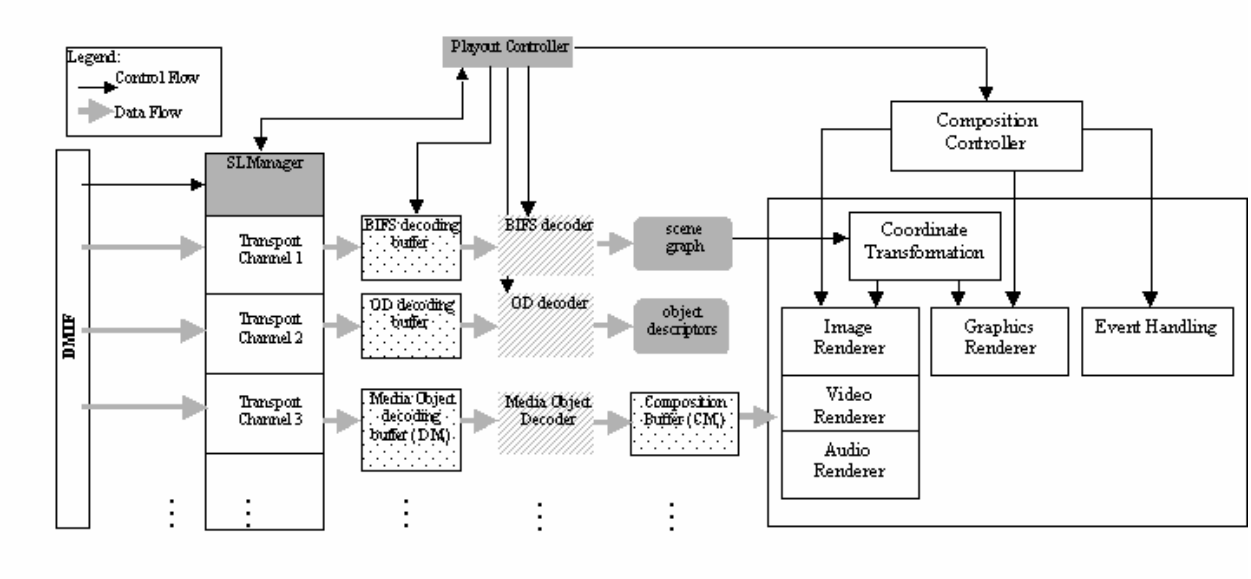


Figure 3.1 (b) Client architecture (figures used from [85] with author's permission).

3.1.1 Approach

We discuss algorithms to schedule the playout, authoring (multiplexing) or retrieval of audio-visual objects under relaxed temporal synchronization restrictions. While all these happen at different stages in content creation and delivery, we try to relate the underlying service requirements and provide a common framework for modeling and optimizing them. An early-tardy (E-T) penalization framework is used to analyze these issues. The E-T problem has been

studied extensively in the operations research literature (see, for example, Baker and Scudder [8] or Cheng and Gupta [35] for excellent reviews). However, we are not aware of any applications of this to multimedia communication and networking. Besides studying the ET problem in a particular application domain, we augment earlier optimization schemes as well. We discuss how fast sub-optimal schemes can be used along with a subsequent neighborhood search mechanism to yield heuristics that perform well in practice.

We recognize two sub-parts to our problem. Although this classification distinguishes scheduling for composition (composition model) and scheduling for streaming (stream-based model), both the schemes are closely related. The composition model involves scheduling the playout (or transmission) of objects having a common display time, e.g., while doing display in a web browser. Using the algorithmic framework based on [8], we present a brute force optimal scheme and several heuristics to find the optimal playout schedule. To capture temporal elasticity in the content, we assume that objects require different tardiness and earliness constraints. Typically, this elastic penalty depends on the visual content of objects and their modality. This observation can be further substantiated based on earlier experimental studies (see for e.g., Steinmetz in [73]) which have evaluated the tolerance of users to skews among audio/video streams. These works suggest three tolerance regions for jitter in playback of audio-video streams: (a) The in-synch region ranges from 80 ms and lower (b) The out of synch region ranges from 160 ms and higher (c) It is also observed that video ahead of audio is more tolerable than audio ahead of video. Similar results, obviously with different tolerance bands, have also been identified for remote teleconferencing applications. Our work extends and formally models the results in these studies and tries to relate these tolerance levels with the underlying scheduling mechanisms. The second subpart, i.e., the stream based model, discusses scheduling of isochronous media. It leads to a non-work-conserving scheduling discipline for the transmission and/or play-out of video or audio. A method to optimally insert idle time in the bitstream, or equivalently, a way to add optimal slippage in displaying frames is presented. The scheme is optimal in the sense that it jointly (a) minimizes the perception of motion discontinuities or gaps seen in displaying late frames, and (b) implicitly minimizes buffering requirements for storing early frames. Thus, the algorithm reduces potential burstiness in traffic at a minimal cost.

3.1.2 Related Work

We are not aware of any prior work on object scheduling, playout control or synchronization that has analyzed the problem using an earliness-lateness penalization framework. But, a significant amount of work has been done for specifying inter-media synchronization and temporal requirements [73][98]. The most well known methods (Blakowski et.al. in [73]) for this are interval-based specification, axes-based methods (e.g., by using global timers, virtual axes), control flow based specification (using reference points or Petri-nets), and event and script based methods. The issues addressed in this chapter have also been part of standardization efforts like the Multimedia and Hypermedia information coding Experts Group (MHEG) and the Hypermedia/Time Based Synchronization Language (HYTIME). Several independently developed structured authoring systems like HyOctane, Nsync, CMIFed, FireFly, and "*temporal-glue*" [120], too, have visited similar issues.

Perhaps the closest to our approach is that of Buchanan and Zellweger [26]. In their study of the FireFly system, the authors associated durations with the presentation of multimedia objects. A duration was given by a tuple of the form <*minDuration, optDuration, maxDuration*>. Costs were associated if media objects were stretched or shrunk away from *optDuration*. Shrinking and stretching can be achieved by altering the rendition speed, skipping or repeating frames, showing only key frames, etc. However, their system did not incorporate network level delays and constraints due to client or network resources. Kim and Song [93] extended Buchanan's concept of *temporal elasticity*. An elastic document has an additional freedom while being authored and browsed, and this can be used to satisfy synchronization requirements or to meet inter-media constraints. Kim and Song, however, did not give a thorough mathematical analysis of the problem and suggest using the Simplex Method after linearizing the problem. Their model falls out as a special case of the stream-based model with the appropriate interpretation of parameters. Consequently, using the analysis we discuss later, one can get much faster heuristics to solve their problem. Some recent work [26][73] has looked at a number of hooks for temporal specification schemes to incorporate on-the-fly changes due to interactivity and network variations. Again, there still seems to be a lack of systematic framework with a rigorous

justification. Yet another body of work (K. Naik in [73]) discusses handling synchronization in the event of faults and incomplete timing information. This is in line with our philosophy of a relaxed temporal specification. More on the specification side of temporal requirements, the authors of the CHIMP project [28] show how several forms of inter-object temporal requirements can be specified using prioritized difference constraints. The authors propose algorithms to incrementally solve the difference-constrained problem and to optimally relax constraints in case the problem becomes infeasible. The primary focus in that work was on collaborative multimedia systems in which several cooperating authors jointly create a document. In our earlier work in [15], we presented algorithms to do resource constrained object selection and transcoding when doing authoring. That approach used explicit constraints and was easily extensible to incorporate several enhancements such as logical inter-dependencies among objects, simple non-linearities, online implementations, and simplifications obtained by object aggregation. In this work, instead of focusing on the use of explicit constraints, we give an implicit way of modeling resource limitations along with temporal requirements. Display deadlines are modeled explicitly, while buffer size variations are included implicitly in the objective function.

3.1.3 Outline

This chapter is structured as follows. In Section 3.2, we introduce and formulate the problem by partitioning it into a *composition* and a *stream-based model*. We discuss how to calculate/estimate earliness and lateness cost parameters for applying these models. Section 3.3 presents a way to find an optimal non-work-conserving schedule to solve the stream-based case. Section 3.4 details properties that must be satisfied by an optimal solution to the composition model. Based on these, a brute force algorithm and several heuristics for finding the optimal solution are outlined in Sections 3.4.1 and 3.4.2, respectively. In Section 3.5, we present an algorithm that uses results from both composition and streaming models together. We also discuss extensions and some related applications of the ET penalization framework. Finally, we conclude in Section 3.6 with suggestions for future work.

For publications related to this chapter, see [11].

3.2 Modeling

MPEG-4 Systems [76][77][124][146] is used as the architecture that we use to build and explain our models. The ideas, however, apply to any structured multimedia format. MPEG-4 encompasses most of the functions that we need to present algorithms for. Figure 3.1(a) shows MPEG-4 System components at the server side. Audio-visual objects are encoded in separate elementary streams. While compressing, the encoder assumes knowledge of the terminals' buffering and composition memory capabilities. The content creator multiplexes elementary streams along with the scene description information. All multiplexing and scheduling operations considered in this chapter are at the FlexMux level. FlexMux has two modes to flexibly multiplex objects -- a Simple Mode and a MuxCode Mode. Both of these are geared towards low-bitrate and low-delay applications. The basic unit of the FlexMux layer is a FlexMux-PDU, and a sequence of FlexMux PDUs is called a FlexMux stream. As we see in Figures 3.2 and 3.3, in the Simple Mode, only data from one object may be included in a FlexMux-PDU. In the MuxCode Mode, however, data from multiple objects can be encapsulated in a single FlexMux-PDU. In the context of multiplex scheduling, we are interested in finding the optimal ordering of AL-PDUs inside a FlexMux PDU in the MuxCode Mode and the ordering of FlexMux-PDUs themselves in both the Simple and MuxCode mode.

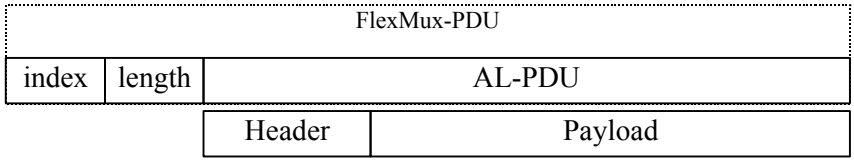


Figure 3.2: Structure of FlexMux packet in simple mode (from [76])

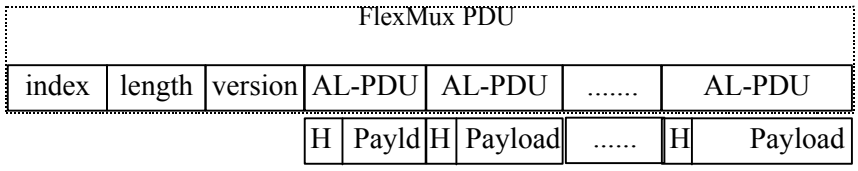


Figure 3.3: Structure of FlexMux packet in MuxCode mode (from [76])

Before proceeding with the two models (i.e., scheduling for composition and streaming), in the following section, we discuss how to calculate the penalty incurred by buffering some objects. We also derive the cost of delaying the display of some objects beyond their target display times, and see how it gets affected by temporal correlation in audio-visual content. We will show that both of these penalization terms vary linearly with the earliness or lateness. Thus our objective function later on becomes a weighted sum of linear earliness and lateness penalization terms.

3.2.1 Earliness-lateness costs

The primary application of the early-tardy composition model considered in this chapter is in the design of an adaptive presentation controller. The client is assumed to have much lower processing and buffering capability than is needed to decode and playback the entire content. We further assume that the application requires certain soft deadline constraints to be met. A solution to the composition model tells us the ordering in which the processor shares its resources among the decoders. The so-called early objects are decoded and displayed at the desired display time. Those in the late set are not decoded and a previous reconstructed instant is repeated and used for prediction.

When scheduling a sequence of objects with soft deadline constraints, two forms of distortion occur. First, an object that is displayed later than its target time causes jitter on display. We show below that the extent of this jerkiness increases monotonically with the difference between target and achieved display times. Moreover, for a given time skew, the effect as perceived by the end user depends on the temporal correlation of the visual content. The spatio-temporal covariance function for the object is used to capture these effects. Second, an additional (usually ignored) component of distortion (or penalization cost) occurs for objects that are decoded prior to their due times. These objects need to be buffered until the target display time arrives and therefore incur an earliness penalty. This buffering cost typically increases with the size of the objects.

3.2.1.1 Lateness penalty

We consider the lateness cost first. The treatment given here derives the lateness penalization term for an image sequence. However, the method is general enough and can be applied to other forms of time dependent (isochronous) media as well. The spatio-temporal covariance function [81][111] between two points (x_1, y_1, z_1) and (x_2, y_2, z_2) in an image sequence is usually assumed to be separable in spatial and temporal dimensions and can be represented as:

$Cov.((x_1, y_1, z_1), (x_2, y_2, z_2)) = Cov._{spatial}((x_1, y_1), (x_2, y_2)) \times Cov._{temporal}(z_1, z_2)$

Here $Cov._{spatial}((x_1, y_1), (x_2, y_2))$ is a two-dimensional Gaussian function, and $Cov._{temporal}(z_1, z_2)$ is decaying exponential. These are given by:

$$Cov._{spatial}(x_1, y_1; x_2, y_2) = s_0^2 \exp[-a_1^2 (x_1 - x_2)^2 - a_2^2 (y_1 - y_2)^2 - 2R a_1 a_2 |(x_1 - x_2)(y_1 - y_2)|]^{0.5}$$

And, $Cov._{temporal}(z_1, z_2) = r_0 \exp[m | z_1 - z_2 |]$

The parameters of this model (i.e., m, a_1, a_2, R) are content specific, and may have to be estimated by doing pre-analysis on the video sequence. If we consider only the temporal part of this expression we get that the correlation, r, between two points separated by f frames is given by, $r(f) = r_0 \exp(-mf)$. The effect of delaying frames can be considered equivalent to the distortion introduced by dropping a sequence of frames. Assume, without loss of generality that $a_1 = a_2 = a_0$ and $R=0$. Therefore, the distortion introduced by skipping $k\ (>0)$ frames is given by:

$$D(k) = \sum_{j=1}^{k} \sum_{x_1=1}^{M} \sum_{y_1=1}^{N} \sum_{x_2^j=1}^{M} \sum_{y_2^j=1}^{N} [1 - r(j) Cov._{Spatial}(x_1, y_1; x_2^j, y_2^j)]$$

Assume further that the spatial distribution stays the same temporally. Thus,

$$\begin{aligned} D(k) &= M^2 N^2 k - \sum_{j=1}^{k} \sum_{x_1=1}^{M} \sum_{y_1=1}^{N} \sum_{x_2=1}^{M} \sum_{y_2=1}^{N} r_0 \exp(-mj) s_0^2 \exp\{-a_0^2 [(x_1 - x_2)^2 + (y_1 - y_2)^2]\}^{0.5} \\ &\approx M^2 N^2 k - \int_{j=0}^{k} r_0 \exp(-mj) dj \sum_{x_1=1}^{M} \sum_{y_1=1}^{N} \sum_{x_2=1}^{M} \sum_{y_2=1}^{N} s_0^2 r^{[(x1-x2)^2 + (y1-y2)^2]^{0.5}} \\ &= M^2 N^2 k - \frac{r_0}{m} (1 - \exp(-mk)) s_0^2 J(M, N, a_0) \end{aligned}$$

where we assume that $r = \exp(-a_0)$, and $J(M, N, a_0) = \sum_{x_1=1}^{M} \sum_{y_1=1}^{N} \sum_{x_2=1}^{M} \sum_{y_2=1}^{N} r^{[(x1-x2)^2 + (y1-y2)^2]^{0.5}}$

As can be seen in Figure 3.4, D varies as a linear function of k asymptotically. To a first approximation, this establishes linearity of the lateness penalization term. The following three cases can be considered separately:

1. m small (low motion, high correlation)

   Let $m = e$. Thus,

$$D(k) = M^2N^2k - \frac{r_0}{e}(ek - e^2k^2 + \ldots)s_0^2 J(M, N, a_0)$$

$$= [M^2N^2 - r_0 s_0^2 J(M, N, a_0)]k \quad \text{as } e \to 0$$

2. m large (high motion, low correlation)

   Taking limit as $m \to \infty$ using L'Hospital's rule, we get $\lim_{m\to\infty} D(k) = M^2N^2k$

3. k large

   The behavior for large k is dominated by the linear term. Hence we get,

$$D(k)|_{k\to\infty} = M^2N^2k - \frac{r_0 s_0^2 J(M, N, a_0)}{m}$$

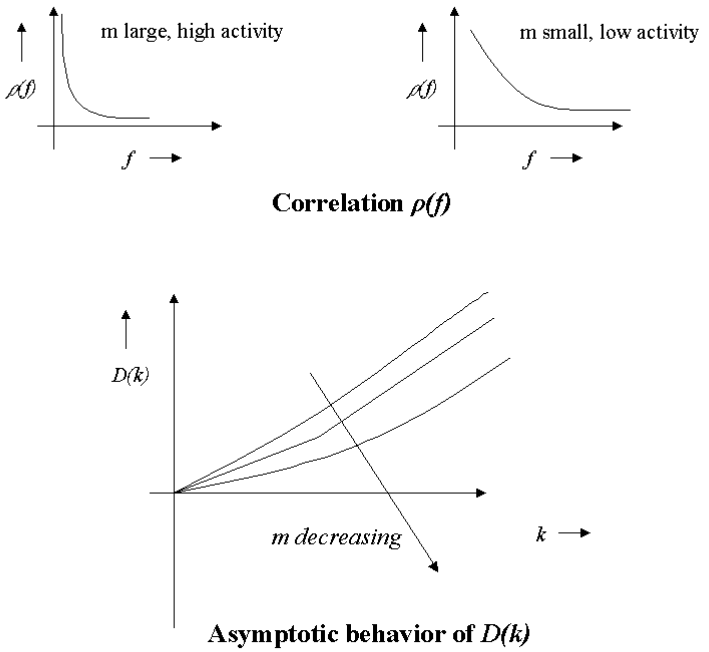


Figure 3.4: Lateness Penalty

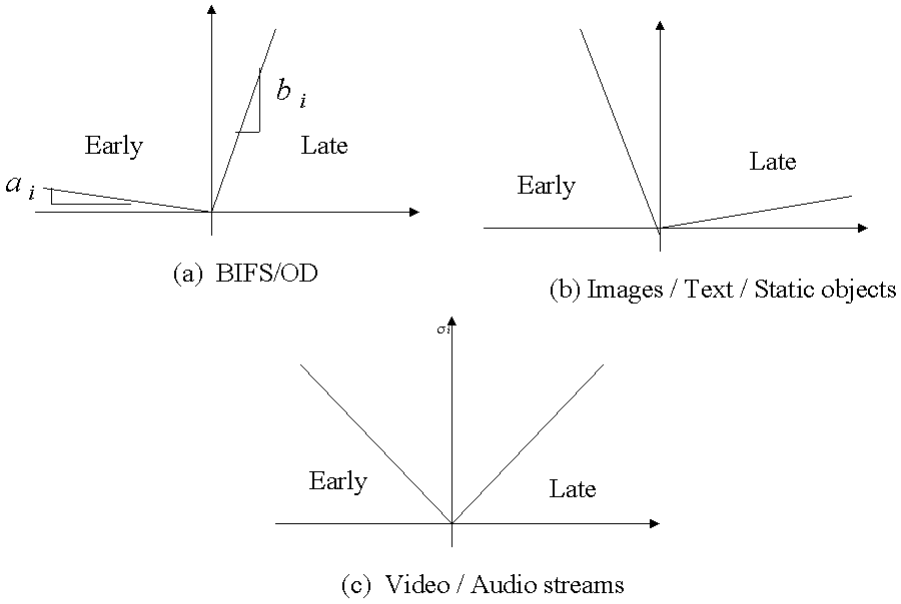


Figure 3.5: Earliness and lateness penalties

From 1 and 2, we note that the slope of *D(k)* for high motion exceeds that for low motion. Hence, for a given *k,* the value of *D(k)* is more for high motion. From (3) we see that as *m* decreases (i.e., for low motion), $D(k \to \infty)$ decreases. Moreover, as should be expected, the reverse is true for high motion. In the $k \to \infty$ limit the difference between the two curves depends on s_0 and r_0.

In summary, the lateness penalty is the loss of correlation incurred by not decoding a sequence of object instants. It gets parameterized using the slope of *D(k)*, which we will call b in the analysis later. For late or un-decoded object planes, the display (let us say at time *n*) is delayed by repeating the previous video object plane (VOP) for *k* time instants. Thereafter, the *(n+k+1)* th object plane is decoded using the earlier reconstructed VOP *(n^{th})* for prediction, and the display is refreshed.

3.2.1.2 Earliness Penalty

Earliness penalty is modeled as follows. Contrary to not decoding (or delaying) the display of a video object (VO), the following happens if an object instance is decoded (see Figure 3.1(b)):

- Compressed VOP_i is extracted from the decoding buffer DM_i thereby freeing up some client memory.
- VOP_i is decoded and this consumes CPU processing power normally proportional to the object size.
- The decoded VOP_i is put in composition memory CM_i. Again the memory consumption is proportional to the object size, s_i. The decoded version is held in composition memory until the display time arrives.

The above three operations are combined to yield an earliness penalty factor. First note that objects should not be decoded too far in advance since the decoded version must be retained in composition buffer until the display time arrives. This suggests a penalization term that increases with the difference *(target display time – decoding time)*. Second, each decoded object increases decoder buffer occupancy by an amount equal to the decoded size (since, typically decoded size >> compressed size). Decoding an object consumes CPU cycles normally proportional to object size. Since both buffering and processing power terms increase with s_i, we assume that the earliness cost is proportional to the object size. We absorb the two factors (i.e., the processing power and buffer consumption) using an earliness penalization factor (a_i) that varies as a monotonically increasing function ($f(s_i)$) of the object size[6]. We use an online statistical estimator to find a_i as described in the following.

3.2.1.3 Estimating Parameters

A set of linear predictors is used to estimate a_i and b_i. a_i is a function of the time needed to decode O_i ($(dt)_i$) and the decoded object size for O_i ($(s_d)_i$). Both $(dt)_i$ and $(s_d)_i$ are typically monotonically increasing functions of the encoded size of O_i ($(s_e)_i$). Denote by index t the current time instant. We wish to predict the CPU cycles needed to decode time instant $t+1$ of O_i

[6] Finding the exact functional form of f is not the focus of this work.

[$(dt)_i \mid_{t+1}$]. Likewise, we also would like to predict the decoded object size at $t+1$ [$(s_d)_i \mid_{t+1}$]. The above two predictions are combined into the earliness penalty factor at time $t+1$ [$(a_i) \mid_{t+1}$]. Similarly, b_i depends on temporal correlation in audiovisual content. We approximate $(b_i) \mid_{t+1}$ using a predictor based on the mean absolute difference (MAD) between adjacent time instances of an object.

An adaptive auto-regressive estimator is used to predict future data points given the past history. The estimator gives future estimates using a weighted average of past values of data points and prediction errors. The weighting parameter is automatically (adaptively) adjusted based on the tracking behavior of the algorithm. Denote by Y_t the variable being autoregressed (for example, $MAD_i \mid_t$) . Let $\hat{Y}_{t+1}$ be the predicted value of Y_{t+1}. We use a prediction procedure given by:

$\hat{Y}_{t+1} = x_t Y_t + (1-x_t)\hat{Y}_t$, where

$x_{t+1} = \left|\frac{E_t^1}{E_t^2}\right|$, $E_t^1 = y\,(Y_t - \hat{Y}_t) + (1-y\,)E_{t-1}^1$, $E_t^2 = y\,(|Y_t - \hat{Y}_t|) + (1-y\,)E_{t-1}^2$, and $0 \le y\, \le 1$

Note that:

1. Five variables ($\hat{Y}_t$ and Y_t for the first equation, and $Y_{t-1} - \hat{Y}_{t-1}$, E_{t-2}^1, and E_{t-2}^2 for calculating $E_{t-1}^{1,2}$) need to be stored at any given time. The entire history gets embedded in these variables.
2. The parameter x_t is updated automatically based on the performance of the predictor. Hence, almost no administrative intervention is required.
3. The parameter x_t adapts based on two exponentially smoothed estimators of prediction error. The first, E_t^1, predicts the residual error; while the second, E_t^2, predicts the absolute residual error. Note that $x_{t+1} \to 1$ if all previous values $E_{t-1}^1, E_{t-2}^1, \ldots, E_1^1$ are of the same sign. Thus, the predicted value is governed primarily by most recent sample and not history before that. Similarly, self-adjusting (positive and negative) $E_{t-1}^1, E_{t-2}^1, \ldots, E_1^1$ terms drive $x_{t+1} \to 0$. Thus a very rudimentary flat predictor $\hat{Y}_{t+1} = \hat{Y}_t$ would work well.

More complicated predictors for estimating a_i and b_i could be used as well[7]. The interested reader may refer to [100] for algorithms to find several other MSE-optimal predictors based on covariance and auto-correlation methods. For this work, we restrict our attention to adaptive exponential smoothing methods to keep the space-time complexity low.

Summarizing, this section discussed how to calculate and estimate earliness-lateness penalization factors. The subsequent sections will present composition and streaming-based models and ways to solve and build on them.

3.2.2 Scheduling for composition

Assume that there are *n* objects, $\{O_1,\ldots,O_n\}$, which have to be composed together to form a display instant of the scene[8]. We wish to design a playout scheduler that selects the set of objects to decode, finds an optimal order in which to decode them, and finds a target display time for this "*quanta*" of objects. We intend to find these such that the aggregate penalty for decoding (and rendering) or dropping (and delaying) the display of objects gets minimized. The two objectives (penalization terms) can be combined using an objective function of the following form:

$$\text{Composition Problem: } f(sequence,d)=\sum_{i=1}^{n} a_i \max(d-C_i,0)+\sum_{i=1}^{n} b_i \max(C_i-d,0)$$

Where d = common display time of the decoded objects.

a_i = scaling factor that indicates cost for earliness of object O_i

b_i = scaling factor that indicates cost for lateness of object O_i. Justification for the linear term was given in Section 3.2.1.1.

---

[7] For example, one could use a finite depth linear filter. Assuming that we use a linear combination of past *p* observations, we can write $\hat{Y}_t=\sum_{i=1}^{p} c_i Y_{t-i}$. Unlike exponential smoothing heuristics, a significant computational effort may be required here in building a good model (i.e., in estimating, storing, and updating the optimal parameter set $\{c_i\}$ under an MSE-like metric).

[8] The *level of granularity* to be used depends on the application domain. Throughout this paper, we will assume that the data unit (object) under consideration corresponds to a displayable entity (e.g., a set of audio samples or a video object plane). We do not focus on lower-layer scheduling schemes like earliest-deadline-first (EDF), rate-based scheduling and pipelining architectures.

C_i = time at which decoded version of object O_i is put in composition buffer CM_i.

s_i = size of object O_i (in time units).

There are several points worth noticing about the above expression:

- The objective function *f(sequence, d)* indicates the penalty incurred due to a variance in rendering around *d*. The distribution to the right of *d* is affected by the loss of correlation. Similarly, the cost of derailing towards the left is a function of buffering and processing power required for decoding O_i.
- We assume that data arriving prior to display time is decoded and held in a composition buffer. The buffer renders its contents to the monitor at time *d.*
- The objective function does not take explicit account of composition buffer size at the client. Instead, it implicitly tries to minimize buffering at the terminal by penalizing early decoding and buffering attempts using an earliness cost. This would make sense, e.g., if during compression or transmission the encoder does not have complete knowledge of terminals' buffering and composition memory capabilities. Alternatively put, the above model is useful for specifying requirements for playout scheduling and synchronization if the end terminal has very *limited buffering* and low CPU power and the application needs that only certain *soft deadline* constraints be satisfied.
- We assume that a_i and b_i are normalized between 0 and 1, and that earliness and lateness penalties may be combined additively.
- The objective function depends on two factors. First, we need to find an ordering in which to decode objects. Second, we also need to find a target display time *(d)* of these objects. These two objectives are inter-coupled in that the choice of one may affect the other. Moreover, since we are considering composition, *d* does not occur periodically (e.g., every 30 ms). The optimal choice of *d* gets decided after solving Composition Problem. Hence, the display could be aperiodic.

3.2.3 Scheduling for streaming

In the previous section, we considered the decoding and playout of objects that need to be rendered together. The optimal rendition time and the decoding order was an output of the

algorithm. However, in most isochronous applications, the display time is a pre-specified quantity and occurs, for example, every 30 msec. In this section, we analyze the optimal transmission or multiplexing of a sequence of frames (or *"quanta"* of objects) having *given* distinct display times. Within each such *quanta* of objects, the algorithms for optimizing playout scheduling using the composition model are applicable.

Like the previous section, an early-tardy penalization framework is used here as well. We wish to find an optimal way to multiplex a sequence of frames under soft deadline constraints. In other words, we intend to find an object-scheduling scheme that minimizes an objective function of the following form:

$$\text{Streaming Problem: } f(sequence) = \sum_{i=1}^{n} \alpha_i \max(d_i - C_i, 0) + \sum_{i=1}^{n} \beta_i \max(C_i - d_i, 0)$$

Here d_i = target display time of object O_i.

α_i = a scaling factor that indicates earliness penalty of object O_i.

β_i = a scaling factor that indicates lateness penalty of object O_i.

C_i = time at which the last bit for object O_i is received.

Here,

- The sets of variables $\{d_i\}$ are *given* and *distinct quantities*.
- The summation is over objects displayed synchronously one after the other. In composition problem, it was over (candidate) objects to be displayed at *d.*
- Candan et.al. [28] classified media objects into several categories. We use a part of their categorization to highlight temporal flexibility for different object modalities. MPEG-4 objects may comprise of:
  (a) <u>*Static objects*</u> including text, non-scalable images, etc. These objects have loose temporal requirements since content does not change frequently. As illustrated in Figure 3.5(b) static objects have low lateness penalty, but a higher earliness penalization factor that increases with the size of object. The choice of β for static objects is application dependent (we use an arbitrary value of 0.1).
  (b) <u>*Temporal objects*</u> including synthetic or natural audio, speech, and video. These objects have stringent ordering, synchronization and temporal restrictions. As illustrated in Figure 3.5(c), this results in intermediate earliness and lateness penalties that have to be

calculated based on Section 3.2.1. Normally, audio is more sensitive to skew as compared to video streams. Since humans are more sensitive to audio-hiccups the penalization parameters (a and b) for audio are intentionally scaled up.

(c) *Object descriptor (OD) and scene description (BIFS)* elementary streams. PDUs from these streams are needed to render objects, and therefore are assumed to have very high lateness penalty (b =0.9) and very low earliness penalty (a =0.1).

The following sections discuss how to optimally solve the scheduling problem for composition and stream- based models.

3.3 Solution to the stream-based model

Note that the early-tardy objective function is not a regular one and hence the optimal solution to the E-T streaming model can be non-work conserving. In other words, the channel may be held intentionally idle (or the display intentionally frozen) even when bandwidth is available. An example best illustrates this unexpected behavior. Assume a hypothetical stream with only two frames which have to be displayed one after the other with a gap of T=30 ms. Assume also that both these frames are very small, so that the channel can carry much more data than is required during the time interval T. Obviously, it does not make sense to transmit the two frames back-to-back if we want to minimize the objective function of the Streaming Problem. Instead, this objective function will be minimum if the two frames are so scheduled that they arrive just-in-time for display. This leads to *empty space* during which the channel is intentionally not fully utilized. If, however, there were no earliness penalization term, one could have transmitted the two frames contiguously.

3.3.1 Algorithm

A sequencing issue normally does not arise when streaming a single isochronous stream. Compression constraints and dependencies govern the ordering of object planes since reference frames are scheduled to arrive prior to predicted ones. Hence the sequencing of VOP's for a single VO gets decided at compression time. Given an ordering of objects, we can still optimally

insert idle time using an approach based on *marginal analysis*. The algorithm [43] can be summarized as follows:

1. We start with the last object instance in the sequence, and move towards the first one.
2. One object gets added in each step. Initially, this added object is placed at time t=0.
3. The added object instance is shifted as far to the right as possible, as long as the marginal cost of moving to the right is negative. The marginal cost is defined as the *unit* cost of moving a *group* of objects to the right. It equals the difference between lateness and earliness costs of late and early objects, respectively, in the *group*. Shifting is done in unit steps of time, and after each step the marginal cost is re-calculated. Notice that the marginal cost changes if a group of objects *hits* on a previously scheduled group of objects. If this is the case, the combined group is shifted as a block in subsequent steps.
4. The procedure terminates after all objects (down to the first one) have been added to the schedule and shifted as far to the right as possible.

3.3.2 Observations

1. The early-tardy objective function is not a regular one, in the sense that having data come late could be beneficial in certain cases. This is because it incorporates (a usually ignored) earliness penalization term. An end device may have very limited buffers (or a very high earliness penalty). Therefore, it does make sense not to transmit as fast as possible in order to avoid buffer overflows. Alternatively, in a playout scenario, it may be better at times not to decode objects as soon as possible in order to avoid unnecessarily hogging up composition memory. The idle time obtained by the ET optimal solution could be a good place to transmit control and signaling information, to accommodate best effort traffic, for applying escape mechanisms, or as an auxiliary capacity for interactive (on-the-fly) changes in resource requirement.
2. The algorithm above can be viewed as a method to smooth out burstiness in traffic. It is likely that by doing optimal scheduling taking into account both lateness and earliness penalties, that potentially less buffer space will be required to remove jitter.
3. In an earlier work, Little and Ghafoor [98] had suggested an algorithm for retrieval and delivery of multimedia data. The primary focus was on retrieval of objects residing on a

single server. The temporal requirements among objects were specified using Object Composition Petri-Nets (OCPN's). Given a temporal specification the authors come up with a method to do optimal retrieval scheduling. Since OCPN's assume tight deadlines, the work in [98] falls off as a special case of the above algorithm if:

- Lateness penalty is infinite,
- Earliness penalty is constant, and
- Network delays (including propagation, transmission, and variable components) are incorporated in object sizes.

3.4 Solution to the composition model

This section explores schemes to find an optimal solution to the Composition Problem. The treatment follows from the algorithmic framework of [8][9][119]. We augment earlier studies by showing how sub-optimal algorithms can be used with a subsequent neighborhood search mechanism to yield good heuristics. To start with, observe that search space for the Composition Problem is very large. If there are n objects, we can have a maximum of $n!$ possible sequence orderings of decoding them. For each such sequence, the target display time could lie virtually anywhere; leading to an infinite number of possibilities that need to be enumerated and compared. Panwalker [119] and later Baker and Scudder [8][9] however, showed that the following properties hold for an optimal solution to this problem. These help us constrain the search space greatly and are given by:

Proposition (a) There is no idle time in the optimal schedule.
Proposition (b) The target display time of objects coincides with C_i of some object.
Proposition (c) The early objects (i.e., those which are decoded prior to the target display time) are ordered in non-increasing order of s_i / a_i [or in weighted longest processing time first (WLPT first) order].
Proposition (d) The late objects are ordered in non-decreasing order of s_i / b_i [or in weighted shortest processing time first (WSPT first) order].

Proposition (e) The object whose completion (end of decoding) time coincides with the target display time (say, O_r) satisfies $\sum_{i=1}^{r} a_i \geq \sum_{i=r+1}^{n} b_i$, where objects are ordered in increasing order of index i.

The interested reader may find proofs in [119]. Each of these proofs involves an interchange argument in which a proposed schedule, not satisfying one of the above, is replaced by one with a smaller objective function. As an illustration, we reproduce the proof of (c) in Appendix A.3. What interests us here is how each of these propositions helps in reducing the search space. Using (a), we are constrained to *n!* choices as we had originally conjunctured. From (b), once an ordering is fixed, we need to look at the completion times of objects only as candidate positions for setting target display time. This leads to an additional *O(n)* computation for each of the *n!* choices. Propositions (c) and (d) together give the most substantial reduction in computation. They say that once the two *sets* of early and late objects are fixed, we only need to carry out a sorting of objects in each of them. Therefore, we should look at all possible ways to form two sets from n objects. This may be done in $\sum_{i=0}^{n} \binom{n}{i} = 2^n$ ways, which is an improvement over *n!* choices. Finally, we see that proposition (e) supplements (b). It says that once an ordering is given, we may get the desired display time by finding the minimum r that satisfies the inequality in (e)[9].

3.4.1 A brute force optimal algorithm

Combining propositions (c) and (d) from [8][9][119] it is easy to come up with a brute force approach of $O(2^n)$ complexity to find the optimal schedule for composition-model. The algorithm (ALG 1) has the following steps:

0. Divide the n objects into two sets. Call one of them an early set and the other the late set. The display time gets decided by how the objects are partitioned into two sets.

[9] Unlike the composition model, in Streaming Problem, the optimal schedule can have idle time. Furthermore, early and late objects do not have any specific sequencing constraints like WSPT or WLPT first. This makes Streaming Problem much more intractable if a sequencing issue needs to be resolved. Luckily, for the application under consideration, this is not the case.

1. In the early set, reorder objects in non-increasing order of s_i / a_i. Resolve ties arbitrarily.
2. In the late set, reorder objects in non-decreasing order of s_i / b_i. Resolve ties arbitrarily.
3. Calculate the objective function. If it is lower than the best ordering yet, replace the best ordering by the current one.
4. Move to the next way of dividing *n* objects into two sets. Repeat steps 1-5. Stop if all the possibilities are already looked at.
5. After looking at all possible sets, the one with the lowest objective function is the desired solution.

3.4.2 Heuristics

Although ALG 1 solves the Composition Problem optimally, it is computationally intensive. It may be prohibitive to use it for any reasonable *n* (e.g., for *n*=20, we have about a million options to look at). From this section onwards, we propose faster heuristics that may be used to solve the above problem approximately. Four heuristics are suggested. The first modifies the dynamic programming approach of [65] and uses it in conjunction with a local search technique called generalized apparent pair-wise interchange (API). The second considers some dispatching rules (WSPT, WLPT and one due to Ow and Morton [116]) and studies them when followed by neighborhood search. The third technique uses Tabu search to look in the neighborhood, while the final one is based on genetic algorithms to do local search. Our primary contribution is in modifying earlier studies on related (though slightly different) problems [8][9][35][65][116][119] to yield heuristics that use a fast sub-optimal search followed by local search.

Common to all these schemes is the idea of the *neighborhood* of a sequence. First, we use a heuristic to get a sub-optimal but feasible solution. An approximate dynamic programming method, WSPT first, WLPT first or a combined dispatching rule is used for this. Although these methods give a solution that is sub-optimal, our intuition suggests that the optimal schedule lies close to it. The idea of *closeness* is described using *neighborhoods.* For a given starting schedule, a set of neighboring schedules is generated using a fixed transformation of the starting

sequence. The transformation we use is called the generalized apparent pair-wise interchange (API) method. In this method, two objects are picked from the original schedule, and their order is swapped. This is done for all possible pairs to obtain an $n(n-1)/2$ sized neighborhood. The original schedule is then perturbed to one of the schedules in its neighborhood and this new schedule becomes the current one, and the procedure is repeated iteratively.

Summarizing, there are four critical steps in these heuristics. First is the selection of a sub-optimal method that generates the original (seed) schedule. Second is how a neighborhood is created. Although we focus on generalized API, simpler or more sophisticated ways are also possible. Third is how the original solution is perturbed to a new one. This may be a simple random choice in the neighborhood or it could be the best choice from this set (which may even be worse than the original schedule). One could choose to move to any or first-hit better choice or select from a subset of choices in the neighborhood, as is done in Tabu search. Finally, the fourth decision is how we decide to stop searching in the neighborhood. We may do this after a fixed number of iterations. Another way could be to stop if there is no better choice in the neighborhood or if the improvement is below a threshold. The heuristics we present below differ in the aforementioned four aspects. In Sections 3.4.2.1 and 3.4.2.2 we give two different ways of choosing the seed solution. Sections 3.4.2.3 and 3.4.2.4 explore the other two aspects, i.e., those of neighborhood creation and searching in it to choose the next solution.

3.4.2.1 An approximate dynamic programming approach

This Section describes an approximate dynamic programming algorithm. The treatment given here modifies the dynamic programming approach of Hall and Posner [65]. First note that propositions (c) and (d) above give us two ways of ordering the schedule of objects. Using them, we can get two lists of objects, say L_1 (WLPT first) and L_2 (WSPT first). Let us look at the optimal schedule starting from the target display time and move outward one object at a time. Objects decoded before the target time come in increasing index order from list L_1. Those after it are in increasing order of index from list L_2. What makes things complicated is that there need not be a single ordering of objects (i.e., a combined list L obtained from lists L_1 and L_2) such that objects can be seen coming from it in increasing or decreasing index order as we move

outward from the target display time. This happens because $a_i \neq b_i$, whence the same object could lie at *contradictory* positions in the two lists. If we can somehow combine the earliness and lateness costs together into one parameter, we may get a single list. This will be the approach we take to get a first hand approximate solution to the Composition Problem. The combined cost for object *i* could be any arbitrary function of a_i, b_i that increases with both parameters; i.e., any $l_i = f_i(a_i, b_i)$ having $\partial f_i / \partial a_i > 0,\ \partial f_i / \partial b_i > 0\ \forall i$ is a valid candidate. Some simple choices could be $l_i = a_i + b_i$, $a_i^n + b_i^m$, $a_i^n \times b_i^m$ etc. For simplicity, we consider only the first one, and for the rest of this work assume $l_i = a_i + b_i$. Using this combined cost factor, l_i, we obtain a list L in which objects are in increasing order of s_i / l_i. Note that with this choice the objective function also gets simplified to $\sum_{i=1}^{n} l_i \,|\, C_i - d \,|$.

After getting a single ordering, it is easy to write a dynamic program [65] to solve Composition Problem. Due to the above-mentioned aggregation of lateness and earliness costs, the solution we obtain will not be optimal to the original problem. Represent by $h_k(e)$ the objective function value for scheduling objects *n, n-1,..., k+1, k* such that the total decoding time (size) of early objects (including the one that gets decoded right on time) is *e*. This leads to the following dynamic program recursion:

$$h_k(e) = \min\{ l_k e + h_{k+1}(e + s_k),\quad l_{\mathrm{k}}(\sum_{i=1}^{k} s_i - e) + h_{k+1}(e)\} \quad \text{for k} = 1,..,\mathrm{n};\ \mathrm{e} = 0,1,...,\sum_{\mathrm{j}=1}^{\mathrm{n}} s_j, \text{ and}$$

$$h_{n+1}(e) = 0 \qquad \text{for } e = 0,1,...,\sum_{j=1}^{n} s_j$$

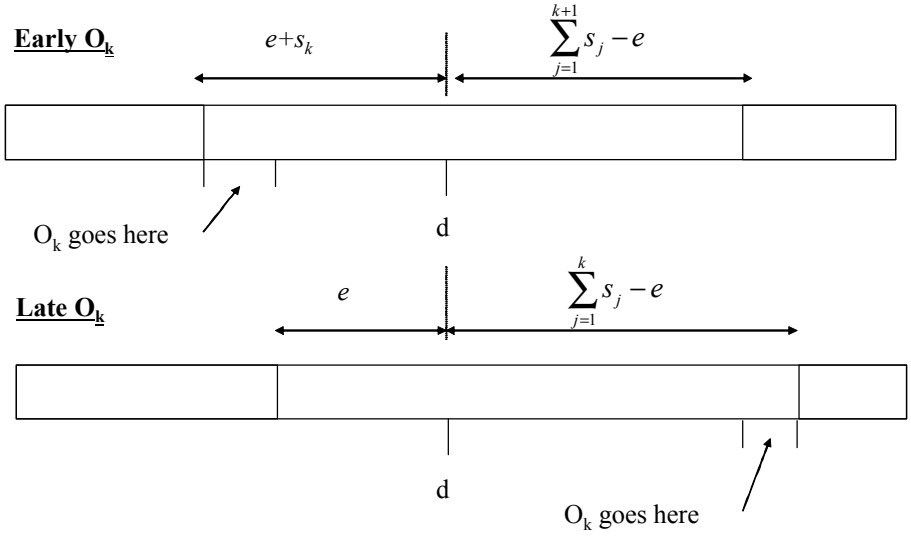


Figure 3.6: Dynamic programming when used on a relaxed problem can be used to get seed solution (situation before O_k is scheduled, i.e., with $h_{k+1}(.)$ already computed and available).

Each incoming object is scheduled either early (to be decoded and rendered) or late (to be delayed or dropped). This accounts for the first and second options, respectively, in the dynamic programming recursion. These two choices (i.e., those of adding object O_k in the early or tardy sets) are illustrated pictorially in Figure 3.6. The algorithm proceeds in decreasing order of k (i.e., from $h_{n+1}(\cdot)$ down to $h_1(\cdot)$), and the final solution is given by $h_1(0)$. The space-time complexity of above program is $O(n\sum_{i=1}^{n} s_i)$. In other words, dynamic programming leads to a pseudo-polynomial algorithm.

After finding $h_1(0)$ and backtracking the optimal solution, we use it as the input "seed" schedule for a subsequent neighborhood search procedure. The neighborhood creation, schedule perturbation etc. are carried out as follows:

1. The solution of the above approximate dynamic program is taken as the initial "seed" schedule.
2. We create a neighborhood around the seed schedule using the generalized API procedure.

3. For each ordering in the neighborhood, a target display time is found by Proposition (e), and using that, we calculate the objective function value. The schedule that gives the smallest objective function value in the neighborhood is labeled as the *critical schedule*.
4. We assume that early and late *sets of objects* are the same as those in critical schedule. Therefore, to satisfy propositions (c) and (d), we reorder objects in early set in WLPT first order, and objects in late set in WSPT first order.
5. The schedule at the end of Step 4 is taken as the new "seed" schedule.
6. Steps 2-5 are repeated using the new seed until its neighborhood has no better schedule.

3.4.2.2 Dispatching schemes

In this section we suggest how dispatching rules can be used to get the seed schedule. A dispatching scheme looks at the current set of unscheduled objects and schedules them one after the other using a predefined rule. Unlike DP, dispatching rules can be both dynamic and static. In other words, the decision rule can stay the same or change over time. Moreover, since a dispatching rule is incremental by its very nature (unlike DP), it is a better choice for online implementations. We consider three dispatching rules. D_1 and D_2 are the trivial WSPT and WLPT rules respectively. D_3 is a slight modification of a dispatching rule first suggested by Ow and Morton [116].

Dispatching rule D_1: All objects are assumed to be in the tardy set, and so the dispatching is done according to WSPT first rule. Note that this dispatching rule is a static one, in that it does not change as more and more objects get scheduled.

Dispatching rule D_2: All objects are assumed to be in the early set, and thus the optimal schedule is in WLPT first order. Rule D_2 is also static.

Dispatching rule D_3: Dispatching rules D_1 and D_2 operate at two extremes. While D_1 gives the optimal strategy for tardy objects, rule D_2 is optimal only if all objects are in the early set in the true optimal. What we ideally need, in a sense, is something that operates in between. The rule should also be able to adaptively switch between rules D_1 and D_2 depending on the *present*

scheduling scenario. Ow and Morton [116] built on these ideas to come up with a mixed dispatching rule and had studied its application for randomly generated input. To start with, all objects are assumed to be unscheduled. Thereafter, objects are scheduled one after the other depending on a prioritizing rule. This rule is used to find the priority of scheduling each unscheduled object next. The object with highest priority is scheduled to be decoded (or gets added to the stream depending on the application), and the process continues until all the objects have been scheduled. As each object gets scheduled, it is removed from the unscheduled set and added to a scheduled one and the current time is incremented by the size of object just scheduled. The priority function of object *i* (P_i) is modified[10] from [116] and is given by:

$$P_i(K_i) = \begin{cases} b_i/s_i & \text{if } K_i \le 0 \\ b_i/s_i \; - \; \dfrac{K_i}{k\bar{s}}(b_i/s_i + a_i/s_i) & \text{if } 0 \le K_i \le k\bar{s} \\ -a_i/s_i & \text{otherwise} \end{cases}$$

Here $K_i = d - t - s_i$ is the slack for object *i* if the current time is *t*. Slack indicates whether object *i* would be early ($K_i>0$) or late ($K_i<0$) if it were scheduled now. $\bar{s}$ is the average size of an unscheduled object. Figure 3.7 shows $P_i(K_i)$ as a function of slack (K_i). We see that if $K_i \le 0$, objects are likely to be tardy, and are therefore should be scheduled in WSPT first order. Similarly, if $K_i \ge k\bar{s}$, object *i* is likely to be early, and gets scheduled in WLPT first order. In the intermediate range, the priority of scheduling object *i* next is assumed to vary linearly between the two limits. *k* is a look-ahead parameter that tells how many objects on the average will get affected by scheduling an object next. A larger value of *k* means that we look far into the future to avoid clashes, hoping to get a globally optimal solution. A smaller value gives a locally optimal schedule, but can lead to multiple objects all competing closely to be scheduled at a future time. Since all objects are due at the same time in our problem, we set *k* as a reasonably large number to avoid clashes.

[10] Ow and Morton used a very similar rule for a case where the target display time is a given input and could vary for different objects. Our problem is distinct in that the display time has to be decided by the rule, and it is common for all objects. Further, we don't use dispatching rules in isolation. There is a follow-up neighborhood search stage that generates the final solution. Of course, this method can be easily extended to schedule packets due at different times, and is therefore applicable for stream-based applications as well. For that we would substitute $K_i=d_i-t-s_i$, and the rest of the algorithm would stay the same.

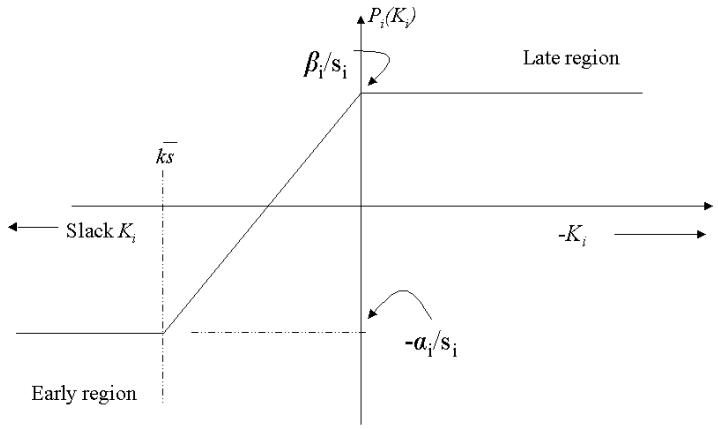


Figure 3.7: Combination of WSPT first and WLPT first dispatching rules (modified from [116]).

The above problem differs from the one analyzed by Ow and Morton in that here we don't know a *d* to start with. To get around this, we initially set *d* to $n\bar{s}/2$ and iteratively try to converge to its optimal value. Given a *d*, a sequence is found using Rule D_3, and the target display time is again calculated by Proposition (e). The procedure is repeated iteratively a couple of times using an estimate of *d* from the previous iteration. After obtaining solutions using dispatching rules (D_1, D_2, or D_3), we do a neighborhood search around it. The method used is same as with dynamic programming (steps 2-6). DP and dispatching rules are ways to obtain seed solutions. The next two sections explore the other part of optimization, namely local search. In particular, we study two methods based on Tabu Search and Genetic Algorithms.

3.4.2.3 Tabu search

Like most local search methods, Tabu ("taboo") search is also based on moving from one feasible solution to another one while looking in a neighborhood. In this method, however, we make a deliberate attempt to avoid being trapped in a local minimum while searching for the

global minima. This is done as follows (for a more complete treatment, the reader is referred to [62][123]):

1. Tabu search allows us to shift to the best possible move from the neighborhood *even if* it does not lead to an improved solution.
2. To avoid jumping back from a new state (schedule) to a recently visited one, a list of tabu ("taboo") schedules is maintained. After moving to a new schedule, we add it to the tabu list. The earliest added schedule is removed from the list to keep the number of tabu schedules constant.
3. If all moves in a neighborhood are tabu, one is picket at random and the process continues.
4. Since the procedure could continue forever, we enforce a stop after a fixed number of iterations. The best solution yet is taken as the heuristic solution.

Solutions of WSPT first, WLPT first, dynamic programming and dispatching rule D_3 are used to find the initial seed. The above outlined search procedure is carried out starting with each of these. Finally, the best of all four schemes is chosen as the final solution.

3.4.2.4 Genetic algorithm

Genetic programming provides a generic set of techniques for doing local search. These methods are based on Darwin's theory of "*survival of the fittest*" in the process of evolution. Although a different terminology is used (see Table 3.1 for a list), the basic ideas are very similar to tabu search. A formal statement of the method we use is as follows (for a thorough overview, the reader is referred to [19][96]):

1. As with Tabu search, the solutions of WSPT first, WLPT first, dynamic programming and dispatching rule D_3 are used to obtain the initial population. A constant population size of 4 is assumed. Initialize k=1.
2. Consider the k^{th} generation. Find the most and least fit individual in this generation. Form a neighborhood of the most-fit individual (in our case, using generalized API). Add the most fit individual from this neighborhood to the population. Remove the least

fit individual from it so as to keep the population size constant. Keep track of the most-fit individual until the k^{th} generation. Increment k.

3. Stop after the number of generations is more than a threshold, else go back to step 2.
4. Pick the most-fit individual yet as the solution.

Scheduling terminology	Genetics terminology
Current set of schedules	Current population
Sequence or schedule	Individual in population
Value of objective function	Fitness of individual
Iteration	Generation
Possible new solution	A newborn child
Previous solution	Parent
Finding a new solution	Applying a mutation

Table 3.1: Scheduling terminology vs. genetics terminology.

3.5 Variants

Below, we discuss two extensions of the above-presented early-tardy penalization framework.

3.5.1 Using both the models together

The treatment until now analyzed algorithms to schedule the playout of audio-visual objects using an early-tardy penalization framework. The problem was discussed by dividing it into a *composition* and a *stream-based* model. In this section, we give a combined algorithm that uses results from both the models together to schedule the *transmission* (or multiplexing) of a mixture of composite and streaming media. The algorithm is self explanatory, and its outline is as follows:

1. Consider N objects, $O_1,..,O_N$, including both streaming and composite media. Object O_i has n_i desired display instants during the interval of the presentation. This gives us a set of display instants $\{d_{ij} \mid i=1,..,N \text{ and } j=1,..,n_i\}$. n_i could be 1 for static or composite objects.
2. For all pairs (i_1, j_1) and (i_2, j_2), if $|d_{i_1, j_1} - d_{i_2, j_2}| < e$ (e application dependent), bundle the two time display instants. This is an aggregation step in which media that could be treated as a composite is identified. In each quanta of indistinguishable display times, optimize the order of decoding (transmission) of objects using one of the algorithms for composition model. Since the common display time (d) is an output of composition algorithms, shift objects until d matches with the target display time, without changing the ordering.
3. Taking into account the aggregation done in step 2, redefine the set of display instants as $\{ d^*_{ij} \mid i=1,..,N \text{ and } j=1,.., n^*_i \}$. All d^*_{ij} are henceforth sufficiently distinct, and $n^*_i \leq n_i$ (as a degenerate case n^*_i could be 1 for static objects e.g., images.)
4. Consider the matrix of display times:

$$D^* = \begin{bmatrix} d^*_{1,1} & < & ... & < & d^*_{1,n^*_1} \\ d^*_{2,1} & < & ... & < & d^*_{2,n^*_2} \\ . & . & ... & . & . \\ . & . & ... & . & . \\ d^*_{N,1} & < & ... & < & d^*_{N,n^*_N} \end{bmatrix}$$

   The ordering constraints required by the application (due to inter-frame compression) are indicated using "<" signs. Observe that due to inter-dependencies, at any given time, only one from a maximum of N objects could be scheduled next. We use the following steps to do the object scheduling:

   4.1 Define the initial *current* set as the set of objects with display times $\{ d^*_{i,1} ;$ for all $i\}$.

   4.2 Find the best object to be scheduled next from the *current* set using dispatching rule D_3, and schedule it.

   4.3 Remove the object just scheduled (say, $O_{i^*j^*}$) from the current set and add $O_{i^*,(j^*+1)}$ to this set.

   4.4 If there are any unscheduled objects, go back to step 4.2, else stop.
5. Optimally insert idle time in the schedule using the algorithm discussed for the streaming problem.

3.5.2 Interdependent objects

Objects that compose a presentation are generally not independent. Inter-dependencies among objects usually arise due to synchronization or compression constraints. The playout controller must schedule taking due account of these inter-dependencies. Inter-dependencies due to synchronization are easily incorporated in the playout controller by aggregating multiple objects and scheduling their decoding (or delay) concurrently. Consider next embedded coding of objects. Denote by $O_{i,j}$ the j^{th} scaled version of object O_i. We assume that layer j cannot be decoded without prior decoding of layers $1,..,j-1$. Layers among objects are assumed to be independent. Unlike the independent layers case, here we solve a *sequence* of optimization problems. Each optimization step involves finding a locally optimal greedy solution. The procedure starts with the first layer of all objects, i.e., the object set $\{O_{i,1}\ "\ i\}$. Each optimization step is similar to the Composition Problem, with the difference that every object has at most one candidate version (with parameters a_{ij} and b_{ij}) entering in a given step. After solving this sub-problem, the layer (say, O_{i*j*}) that is scheduled to be decoded first is removed from the candidate set, and the next layer of this object (i.e., $O_{i*,(j*+1)}$) is entered for the next problem. The procedure is repeated iteratively, until all layers of each object have been scheduled. Subsequently, d is calculated using proposition (e) in Section 3.4. Objects in the early set are decoded and played out in the order obtained and those in the late set are delayed.

3.6 Summary of the chapter and concluding remarks

In this chapter, we discussed algorithms to schedule the playout or retrieval of audio-visual under relaxed temporal constraints. In particular, we used a framework based on the early-tardy due-date assignment problem to investigate these issues. Throughout this chapter, we assumed that objects require different tardiness and earliness guarantees based on their visual content and media-modality. We discussed the justification of using an ET framework and presented ways to estimate parameters of this model. Two sub-parts were recognized for the problem. The first involves scheduling the playout and decoding of objects having a common display time. A brute

force optimal scheme and several heuristics were suggested to solve the composition-model. These heuristics use a preliminary fast sub-optimal scheme to get seed solutions. Subsequently, a local neighborhood search procedure is used to improve the seed solution by giving it a minor perturbation. The second part of the problem, called the stream-based model, leads to a non-work-conserving scheduling discipline to control the transmission and/or play-out of a sequence of video frames. A method to optimally insert idle time in the bitstream, or equivalently, a way to add optimal slippage in displaying frames was suggested. This can be viewed as a way to reduce potential burstiness in traffic at a minimal cost.

Finally, we presented schemes to exploit results from both composition and streaming models together. We discussed how the two sub-parts could be used simultaneously to schedule the transmission of a mixture of non-isochronous and streaming media. Some extensions of the ET penalization framework were also suggested. Summarizing, this work studied a formal framework for handling synchronization and scheduling related research for continuous and non-isochronous media and combinations of these. For publications related to this chapter, see [11].

This chapter dealt with playout scheduling issues in MPEG-4 like systems. We presented models and algorithms for adaptive playout control and discussed an implicit way of modeling resource constraints. Earlier in this chapter we showed how spatio-temporal variance in visual content affects our scheduling decisions. The next chapter explores these ideas further. We present content based approaches for visual segmentation and associated resource allocation for transmission over wireless channels. We use extent of intra-coding, scene changes and motion based features for segmenting content at a frame level. An experimental simulation platform will be used to test the objective (SNR) effectiveness of the proposed FEC and ARQ algorithms.

CHAPTER 4

Content-based schemes for video transmission over wireless channels

---

Contents

4.1 Introduction

Recently, there has been a great demand for audio/visual services to be provided over wireless links. However, due to bandwidth constraints, high error rates and time varying nature of these channels, the received video quality is usually inadequate. New algorithms are needed to enable reliable video communication in wireless environments. This chapter describes new schemes for video transmission over wireless channels. It can be broadly divided into two parts. Firstly, content based approaches for video segmentation and associated resource allocation are proposed. Secondly, joint source/channel coding techniques (in particular, content/data dependent FEC/ARQ schemes) are used to do adaptive resource allocation. FEC based schemes are used to provide class dependent error robustness and a modified ARQ technique is used to provide constrained delay and loss. The approach is compatible with existing video coding standards such as H.261 and H.263 at rates suitable for wireless communication.

Traditional methods of segmenting video into sub-streams involve frame type, headers, and data type (e.g., motion vectors vs. transform coefficients). Improvement has also been shown by separating video to sub-streams and using adaptive resource allocation mechanism [134] These techniques are developed based on the notion that different types of video sub-streams have different levels of importance and should be handled differently. However, these approaches are restricted and do not take into account the ``content'' of the video. In this work, we propose content-based segmentation methods to augment the traditional approaches. Here, video content refers to the inherent visual features present in the video. Examples are structure of the scenes, objects contained in the scene, attributes of the objects (e.g., motion, size, number etc.). Our goal is to demonstrate the value of the content-based video transport framework by providing proof-of-concept results of selected types of video content. We present resource allocation algorithms based on two simple types of video content: scene changes and extent of motion or activity[11]. Both of these features can be extracted from compressed video streams using automatic algorithms [102]. Our approach uses content processing schemes to classify the video content types and then allocate network resources (i.e., FEC and ARQ) adaptively. An experimental

simulation platform is used to test the objective (PSNR) and subjective effectiveness of proposed algorithms. A logical level figure illustrating the system architecture is given below (see Figure 4.1).

The first half of the chapter involves the adaptive schemes based on FEC control. These algorithms are applicable to both one-way and two-way real-time applications. In the second half of the chapter, we present algorithms using selective repeat ARQ for one-way real time video applications (like video on demand). The ARQ-based algorithms may incur retransmission delay that might be too long for two-way interactive services (such as video conferencing). To accommodate the variable delay caused by the selective use of ARQ schemes, our approach also includes an "elastic" buffer before the decoder/display module to "absorb" the delay jitter. The buffer absorbs the rate variation caused by selective ARQs and provides a compatible interface to the decoder. We will present efficient algorithms for buffer control in this case.

The remainder of this chapter is organized as follows. Section 4.2 reviews existing approaches and related work in this area. Section 4.3 explains our proposed approach. Section 4.4 presents the schemes adopted for FEC. We also elaborate on the simulation method and results for forward error correction in this section. ARQ based recovery methods are explained in Section 4.5. Section 4.6 briefly summarizes the contributions of the work and suggests some future directions.

This chapter contains material that is joint work with Prof. S-F Chang. For publications related to this chapter, see [12].

4.2 Related work

Our work focuses on content-based segmentation and associated resource allocation using modified ARQ and FEC schemes. Although there has been a lot of work in scalable or layered coding, and ARQ/FEC usage for videophone applications; the idea of content and object based

[11] For example, new frames after a scene change are subjectively important in order to establish the new visual context during the playback session. Image frames with high motion are less sensitive to errors due to the masking effects in the human vision model.

approach to video coding is very recent and is largely limited to standardization bodies or related works [75][125][156]. Below, we give a brief overview of scalability, ARQ/FEC schemes and concealment options with particular emphasis on wireless video applications.

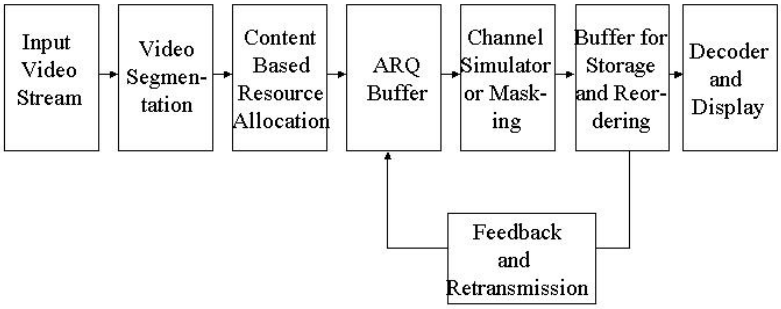


Figure 4.1 Logical level diagram of the system

4.2.1 Data partitioning and layered coding

Scalable [1][57][60][61][130] and joint source-channel coding techniques [104] are two main methods used in real time transport on Internet and ATM networks. Unlike wired media, data loss is a much bigger problem for transmitting video over wireless networks. Bit error rates of around 10^{-3} to 10^{-2} may arise for long periods of time (approximately hundreds of milliseconds) especially in mobile environments. Data loss can usually be prevented using sufficient error correction codes, however, this results in increased bandwidth requirements since channel errors usually occur in bursts.

Error protection schemes are normally used in conjunction with conventional data partitioning schemes [23][99][121] leading to graceful degradation over channels of varying quality. In their study, Arvind, Civanlar, and Reibman [2][6] discuss the packet loss resilience of MPEG-2 scalability profiles. They concluded that for cell loss ratios of about 10^{-3} [12] spatial scalability

[12] which translates into a BER of $O(10^{-5})$

performs the best, followed by SNR scalability and data partitioning in that order. Wavelet, sub-band [21][151], and pyramid coding techniques which lend themselves to scalable architectures have also been studied for mobile environments. Aside from these and other standards based studies, there have also been somewhat unconventional ones as well. These include projects such as the conditional replenishment based Asynchronous video [134] and Priority encoded transmission (or PET) [95] related work at ICSI, Berkeley.

4.2.2 Combined ARQ/FEC schemes

As mentioned earlier, both FEC and ARQ schemes have to be used judiciously for error correction. Khansari et.al. [91] use a combination of scalable (dual rate) source coding with hybrid (Type I) ARQ and FEC for transmission of QCIF resolution H.261 video over wireless channels. Their work concludes that usage of a mixture of FEC and ARQ based structure needs less overhead as compared to using FEC alone for similar visual quality. Likewise, work at the University of Southampton [67][153] also suggests using hybrid ARQ and unequal-FEC codes to improve robustness in transport of DCT-based compressed QCIF videophone sequences. A different flavor of ARQ has been studied in recent work by Han and Messerschmitt [66]. They present a "leaky ARQ" scheme that successively improves the quality of delay-non-critical portions of graphics by sending more refined versions. It trades off quality (reliability) for delay for portions (e.g., menus) that should appear without significant delay. This work was, however, limited to "window based text/graphics" and did not study the extra considerations due to video[13].

4.2.3 Concealment options and robust codecs

Unlike FEC and ARQ schemes, concealment based approaches use spatio-temporal correlation in the bitstream to repatch errors in the video. Typically, these methods use temporal replacement and motion-compensated temporal replacement or interpolation [58][59][94][167]. Current MPEG-4 standardization efforts are actively investigating concealment options such as duplicate information, two-way decode with reversible VLC, and data partitioning

[13] For example, real time deadlines, buffer requirements, error propagation, and inter frame coding etc

[75][76][77][125] showing very promising results. The idea there is to add the ability to quickly resynchronize and localize errors in compressed video streams. Our work shares great synergy with the content-based theme of MPEG-4 and MPEG-7.

Besides aiding in the design of a smart decoder, concealment schemes may also be applied interactively between the source and the receiver. For instance, depending on the channel state or decoder's inability to re-synchronize properly, the encoder can increase the frequency of I frames/macroblocks and decrease slice sizes to help localize errors. Based on feedback about the burstiness of the channel, the encoder could also may also increase the minimum distance between the predicted and predicting frames to prevent temporal error propagation.

4.3 An Approach Based On Video Sub-stream Segmentation

Traditional data partitioning and segmentation methods have exploited the hierarchy in the coding structure. Both the H.26x and MPEG-x suite of standards divide the coded bitstream into many syntax layers. In MPEG-2 (ISO/IEC 13818-2), for example, the entire sequence is divided into group of pictures (GOPs). Each GOP has a specified number of pictures/frames, and starts with an intra-coded I frame. I frames have no motion compensation performed on them. Also present are the P frames which are predictively coded from previous I and P frames. Between the anchor frames (I/P frames), are bi-directionally predicted B frames. Each picture is further composed of slices, which in turn consist of macroblocks. 16x16 macroblocks consist of 4 8x8 blocks. Motion compensation is applied at a macroblock level, wherein a best matching block is found and one or two motion vectors are sent in the bitstream.

Because of this hierarchy in coding, wireless errors affect video quality depending on where they hit. Errors occurring in headers (e.g., sequence start-codes, GOP headers, picture headers, slice headers) have the most detrimental effect. Next, the effect of errors depends on the image frame type -- the sensitivity being I>P>B. The perceived error sensitivity for picture data is motion vectors > low frequency DCT coefficients > high frequency DCT coefficients. Depending on the above, we can define a data segmentation profile in decreasing order of priority in terms of error robustness. The above is basically the philosophy behind the MPEG-2 data-partitioning method.

For wireless applications, the H.26x suite of standards are more popular due to their focus on low bitrate. Although, there are additional options (e.g., PB frames, extended motion vector range etc.) in H.263, the underlying structure of codecs is very similar.

Our approach differs primarily in that it uses video content as a means of further data segmentation. Our goal is to develop a content-based video transport framework. We use a psycho-visual (i.e., content based) framework for video segmentation. A typical video sequence can be divided into a set of independent scenes. Further, a scene is composed of frames of different activities (motion). We use scene changes and motion to further segment the video stream. For resource allocation, we argue that different video content requires different form of resources to be allocated to it. For example, new frames after a scene change are subjectively important in order to establish the new visual context during video playback. Image frames with high motion are less sensitive to errors due to masking effects in the human vision model.

The primary QoS constraints that we consider are error robustness and bounded end-to-end delay. We use variable FEC to provide multi-tile error resilience. We study both the individual and aggregate effects of using video headers, frame-type, motion, scene changes, transform coefficients, and motion vectors as a basis for choosing the error correction overhead. We also use a restricted link level retransmission scheme to provide bounded end-to-end delay. We propose innovative schemes for selectively applying ARQ-based retransmission and controlling the interface buffer between the receiver and the video decoder.

4.4 FEC Based Schemes

4.4.1 Frame Segmentation Based on Scene Changes or the Extent of Intra-Coding

The H.263 codec [64] is used for encoding the video. Hence, INTRA/INTER mode decisions for prediction are made at a macroblock level [64]. The scheme to make these decisions is the same as specified in Test Model Number 5 (TMN-5) [79]. Given a pre-encoded video sequence, frames are prioritized based on the fraction of macroblocks that are intra-coded in them.

Following this, three partitions are generated as:

$$a = \frac{\#Intra\_Coded\_Macroblocks}{\#Inter\_Coded\_Macroblocks + \#Intra\_Coded\_Macroblocks}$$

Partition 1: $a \in [0.4,1]$ Highest priority
Partition 2: $a \in [0.1,0.4]$ Medium priority
Partition 3: $a \in [0,0.1]$ Low priority

The choice of partitioning thresholds (i.e., 0.1 and 0.4) is adhoc here. We show variations in a for a typical H.263 video sequence in Figure 4.2. The priority assignment we consider here is in terms of error resilience and is provided by decreasing the level of FEC protection from Partition P1 through Partition P3. There are several justifications for using this ordering of priorities:

- Frames that are mostly INTRA coded provide refresh instants and help in preventing error propagation. Hence we would like to have least number of errors in them.
- For H.263 based encoding, the first partition typically corresponds to scene changes in the sequence. Perceptually, scene change frames are important for a complete understanding of the new video scenes.
- These frames can also be used to provide scene change based scalability in addition to other schemes of temporal scalability[14].

On the other side, high intra frames use more bits than lower intra frames. Thus, using higher levels of FEC results in significantly higher increase in bit-rate. Using the experiments results we present later in Section 4.4.4, we found that on the average this improves the visual quality under a global (channel + source) bitrate constraint. In other words, increasing the error protection for high intra frames pays off in the long term by preventing error propagation temporally.

[14] e.g., frame filtering options in MPEG which drop B or B+P frame combinations.

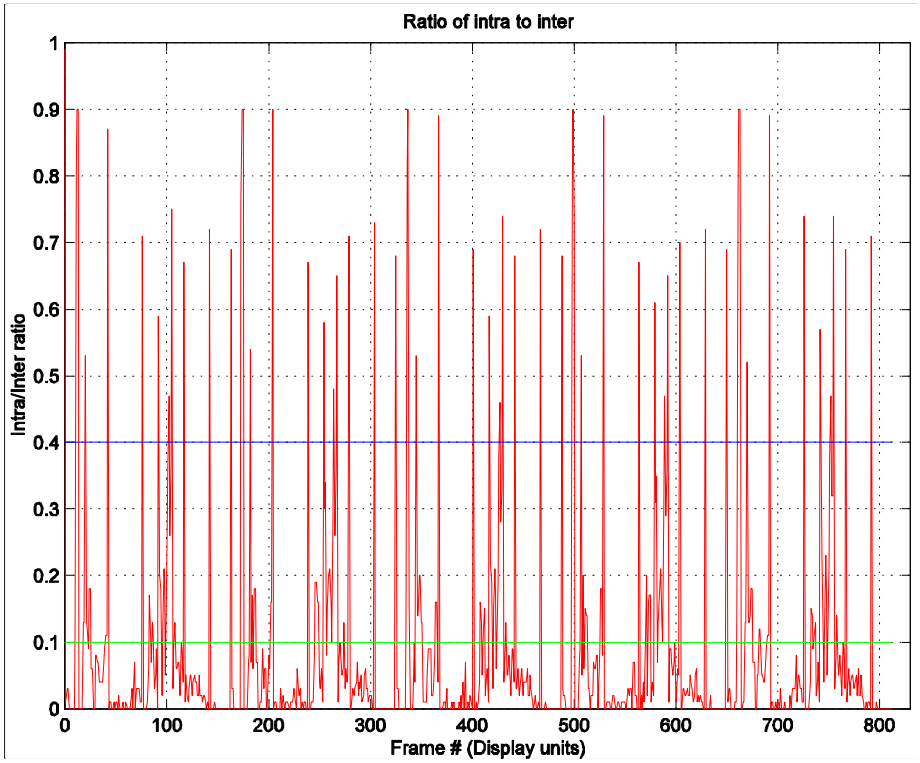


Figure 4.2 Percentage of intra-coded macroblocks in a typical H.263 video sequence.

4.4.2 Motion Based Segmentation

In H.263, macroblocks get coded in intra mode if the best prediction error is beyond a threshold. If not, the macroblocks are inter-coded. Partitions P2 and P3 are largely inter-coded. Hence we use an additional way of partitioning them. The segmentation is done into two levels -- low and high motion, and motion is estimated at a frame level. For each frame, a frame level average motion vector is calculated, and compared with the global average. This is used to classify frames into high and low motion ones. Motion based segmentation can also be applied at a finer granularity e.g., at the macroblock or VOP level[15].

It is known that quantization errors are much less perceptible in areas of high texture due to masking in the human visual system [39]. We experimented to see if similar masking effects could be used when protecting from channel errors (i.e., after run-length and Huffman encoding

[15] VOP or video object plane is a temporal instance of an arbitrarily shaped region in a frame. It is defined in MPEG-4.

of the quantized signal). If we make decisions taking only these perception studies into account, high motion frames should be able to tolerate more channel errors (perceptually) as compared to low motion frames. Hence, we should assign lower FEC overhead for high motion frames as compared to low motion ones. It must be noted[16] that perception based decisions are somewhat contradictory to compression-only based decisions. Through experiments discussed later, we found that the 'compression effects' dominate 'perception-based' decisions for higher intra coded partitions. For partitions that were largely inter coded (i.e., Partitions P2 and P3), we experimented further to see if perception based decisions would work better.

4.4.3 FEC Based Resource Allocation

Figure 4.3 shows the segmentation hierarchy for error-robustness we obtained after experiments. Each state (leaf node) corresponds to a given set of allocated resources. Nodes higher up in the tree have more priority than those below. Similarly, nodes to the left carry more priority than those to the right. We use stronger FEC protection for higher priority layers. Attributes having highest incremental effect on the video quality are placed higher up in the tree and to the left side. Headers form the highest priority layer to ensure that the corrupt bitstream can be decoded properly. This is followed by scene change frames (Partition P1 above). Frames that are not in partition P1 are further divided into partitions P2 and P3. Motion information is used to again split these partitions. Finally, the actual data (motion vectors and DCT coefficients) are at the innermost layer, and normal MPEG or H.26x data partitioning like schemes are applicable within the frame[17]. Specific allocation schemes of different degrees of FEC and their experimental results will be presented in Section 4.4.4.

[16] The author wishes to thank Dr. Amy Reibman for bringing this to his attention.
[17] In this study, all segmentation is done at a frame level. The lowest layer is constructed assuming the experimental results obtained by some standardization bodies.

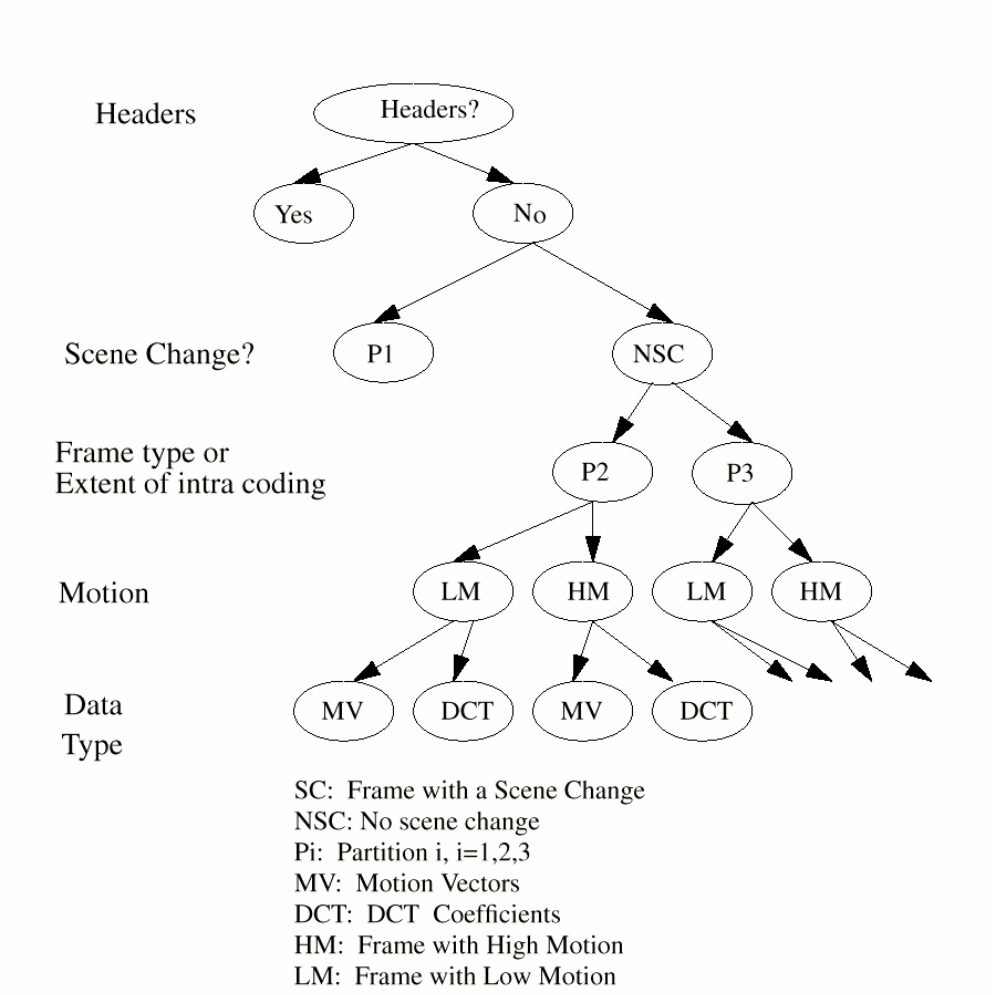


Figure 4.3 Template assignment for error robustness

4.4.4 FEC Results

We use the wireless channel simulator described in [68]. The wireless channel is modeled using the Rayleigh fading multipath model. Personal Access Communications Services (PACS) system is used for the air-interface [112]. The Rayleigh fading channel is simulated using the Jakes Model [83]. Other wireless parameters used in the FEC-simulations are provided in Table 4.1 [18] The simulator is used to generate simulation data with errors under different error controls. The

[18] The authors thank Dr. Li-Fung Chang (Bellcore) for this information.

final data with errors is compared with the original data to generate error masks. Error masks are used to mask the video data to simulate the transmission of video over the wireless channel. The wireless channel simulator [68] is used to generate bit error pattern files on an off-line basis. Error pattern files can be generated for different channel conditions, error correction capabilities, and interleaving degrees. The coded video sequence is then sequentially parsed, and depending on its current content (motion, extent of intra coding etc.), an appropriate error pattern file is used to mask it.

Maximum Doppler frequency, f_d = 1 Hz.
Perfect carrier recovery
Optimal symbol timing
Transmitted signal power = 19dB
QPSK modulation (2 bits/symbol).
Coherent demodulation
Matched Nyquist filter with a roll-off factor of 0.5
Diversity = 1
Round trip delay = 6msec.
Multiple access = TDMA
Air interface: 400 frames/sec each with eight 32 kbps slots
Code length = 40 symbols
Symbol length = 64 bits
FEC: RS(40, 40-2*t*)
Interleaving degree = 40

Table 4.1: Wireless parameters used in the simulation [68].

The simulations are done using the H.263 codec. Further details on the video parameters used in the simulation are given in Table 4.2. The quality of final decoded video is measured using its PSNR, defined as:

$$PSNR = 10\log_{10}\left(\frac{1.5\times176\times144\times255^2}{\sum(Y_e - Y_d)^2 + \sum(U_e - U_d)^2 + \sum(V_e - V_d)^2}\right)$$

where the subscript *e* indicates the decoded version of a frame with errors and *d* stands for a normal decoded frame (not the frame that was encoded). *Y* is the luminance, and *U* and *V* are the chrominance components respectively. Further, because the proposed algorithm is psycho-visual in nature, subjective evaluations are also made on the decoded stream.

Sequence: Unrestricted CNN news sequence (including commercials).
Codec = H.263 [64]
Rate control: Simple offline rate control with fixed frame rate
Rate control ON for the entire sequence.
Resolution: QCIF (176x144 pixels).
Chroma format = 4:1:1
Motion estimation search window = 10 pels.
Encoded bit rate ≈ 32Kbps (CBR)
Frame rate = 7.5 fps.
All enhancement options ON
Number of frames ≈ 106 sec (800 frames)
Other encoding specific parameters are as in TMN5

Table 4.2: Video parameters

In the following, we study the incremental effect of using headers, scene-changes, intra-coded macroblock percentage (a), and motion as a basis for choosing the error correction capability of the code. The PSNR graphs plotted below show PSNR between a normal decoded sequence in the absence of any channel errors and that with channel errors. The large spikes (bound by 150 dB.) indicate infinite PSNR or no errors in decoded video. To maintain fairness in comparisons, the average FEC overhead in each case was maintained approximately same. The total overhead due to channel coding is about 25% for all cases. We compare the average frame level PSNR for the various cases under consideration. Figure 4.4 shows results for a typical frame. Figure 4.4(a) is the normal decoded frame, while Figure 4.4(b) shows the same frame after it is transmitted over the wireless channel. No data segmentation or concealment is used in Figure 4.4(b). Figure

4.4(c) shows the result with header plus a based segmentation. We see that there is a significant improvement in quality as compared to 4.4(b). Figure 4.4(d) uses motion information in addition to headers and a based segmentation. The subjective quality improvement is quite obvious, especially in the impaired image areas.

FEC parameters for Figure 4.5 are shown in Table 4.3. In this table *t* is the error correction capability of the code. Our objective in this study is to compare the increase in FEC overhead per unit increase in PSNR under different scenarios. Thus, you will notice that the effective overhead for one of the curves in each of the graphs (a), (b) and (c) is higher than the other one. Figure 4.5(a) compares PSNR of the decoded stream with no data segmentation and that with FEC done based on headers (e.g., headers for the picture layer). From Figure 4.5(a) we see that a significant gain can be obtained by giving high protection to headers. Giving higher error protection to headers ensures that the decoder finds clean start codes and parts of video do not become un-decodable until next sync. Figure 4.5(b) shows the incremental effect of giving higher protection to Partition P1 frames. These frames are largely intra-coded, and are assigned next higher priority to prevent error propagation. Black spikes in Figure 4.5(b) indicate the location of high-intra frames. It can be seen that the dashed line (header + intra) consistently outperforms or equals the solid one (header only) for a very little increase in the effective error correction overhead. Figure 4.5(c) compares the effect of using Partitions P2 and P3 for further data segmentation. Based on the a value, we use different *t* values for Partition P2 and P3 frames. The large black spikes indicate frames in partition P1 and the smaller spikes are frames in partition P2. The dashed line is seen to outperform the solid one for all the frames. It is worth noticing that with a very small extra FEC overhead for frames in partition P2, much perceptual quality improvement can be obtained. However, the increase is not as much as obtained with headers or Partition P1. In other words, although the effective error correction overhead increases in both (b) and (c) (as compared to (a)(dashed)), the increase in this overhead per unit increase in PSNR is much less with (b) as compared to (c). Similarly, this incremental overhead is least with (a). Thus, we justify the arrangement of top three layers of the tree shown in Figure 4.3.

Note that until now the layering structure was dominated primarily by ‘compression’ (extent of

intra coding) based decisions. It is also well known that quantization errors are much less perceptible in areas of high texture due to masking in the human visual system [39]. Next, we experiment to see if similar masking effects could be used when protecting from channel errors (i.e., after run-length and Huffman encoding of the quantized signal). For this we use a separate motion-vector based feature to further segment partitions P2 and P3. If we make decisions taking only these perception studies into account, high motion frames should be able to tolerate more channel errors (perceptually) as compared to low motion frames. We did not further segment partition P1 since 'compression-based' decision dominates the final quality there. Further, these frames don't have many motion-vectors to start with. We indicate frames with high motion by black spikes in Figure 4.5(d). We considered two cases: in the first, high motion frames are given less FEC overhead as compared to frames with low activity (dashed line); for the other case (solid line), it is the opposite. We noticed that the quality is degraded less in the former case as compared to the latter; and therefore masking based results are substantiated. However, the improvement that we obtained is fairly marginal compared to improvement from top three layers. Further, note that this perception based result is somewhat contradictory to compression-only based decisions. To start with, one would expect that there is high correlation between the intra-coded macroblock percentage and the size of motion vectors. However, due to non-linearities in codecs and source statistics there is no guarantee that it should be so. We believe that lack of significant correlation between the two features (extent of intra coding for layers two and three and motion vector size for the fourth layer) is what is causing this discrepancy.

Overall, it is seen that headers give the largest improvement in SNR for minimal increase in error correction overhead. Scene change (Partition P1) and the extent of intra coding (Partitions P2 and P3) information can be used to give higher error protection to selected frames with decreasing marginal returns in improved visual quality per unit overhead. Motion vector size was then used to further segment Partitions P2 and P3. We found that this feature suggests decisions contradictory to those suggested by extent of INTRA coding. However, the improvement obtained by taking motion vector size into account is much less compared to marginal improvements obtained with top three layers. That is why layer three feature (a) lies above layer four feature (mv size).

Looking at the incremental improvement obtained by each of the above criteria, we verify the design of the layering structure shown in the tree in Figure 4.3. Within each criterion (e.g., partitions P2 and P3 combined), the chosen left to right arrangement is seen to give best results. At a given depth in the tree, leaf nodes to the left are more important in terms error robustness. Hence, if the underlying network allows only a limited number of service classes, or to decrease the FEC encoding/decoding complexity, it may be feasible to cut down on classes bottom up in the tree. Bundling leaf nodes into groups from left to right and providing them same QoS guarantees is yet another possibility. Through experiments discussed later, we found that the 'compression effects' dominate 'perception-based' decisions for higher intra coded partitions.

Figure	**Scenario**	**Parameters**	$t_{effective}$
(a) (solid)	No segmentation.	t=5	t=5.00
(a) (dashed)	Header based segmentation.	t(header)=9, t=5 else.	t=5.04
(b) (solid)	Header based segmentation.	t(header)=9, t=5 else.	t=5.04
(b) (dashed)	Header + Scene changes a e [0.4,1].	t(header)=9, t(scn)=6, t=5 else.	t=5.21
(c) (solid)	Same as (b) (dashed).	t(header)=9, t(scn)=6, t=5 else.	t=5.21
(c) (dashed)	(b) (dashed) + P2 or a_{medium} e [0.1,0.4].	t(header)=9, t(scn)=6, $t(a_{medium})$=6, t=5 else.	t=5.28
(d) (dashed)	High motion – lower FEC.	t(header)=9, t(high motion)=4, t(low motion)=5.	t=5.06
(d) (solid)	Low motion – higher FEC.	t(header)=9, t(high motion)=6, t(low motion)=4.	t=5.03

Table 4.3 Simulation conditions for FEC

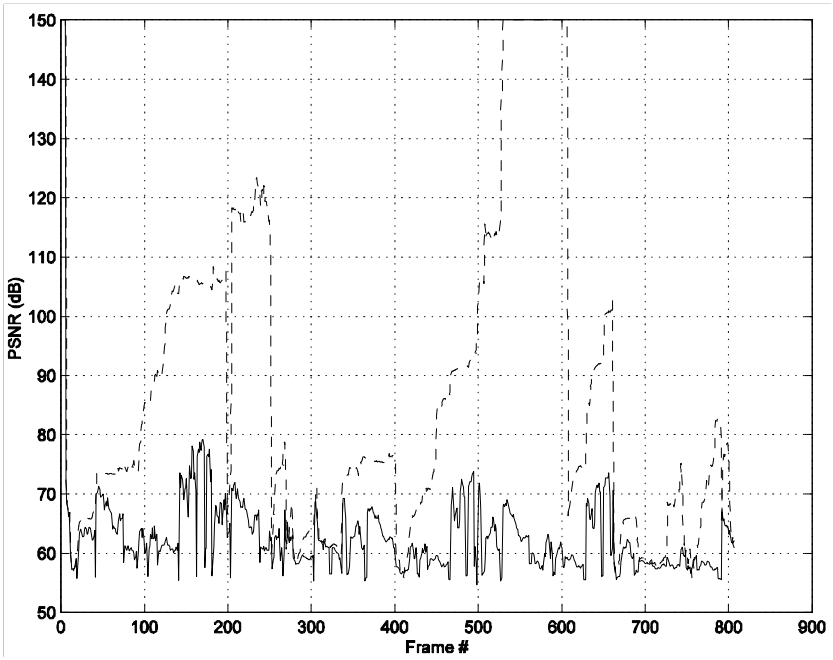
(a)

(b)

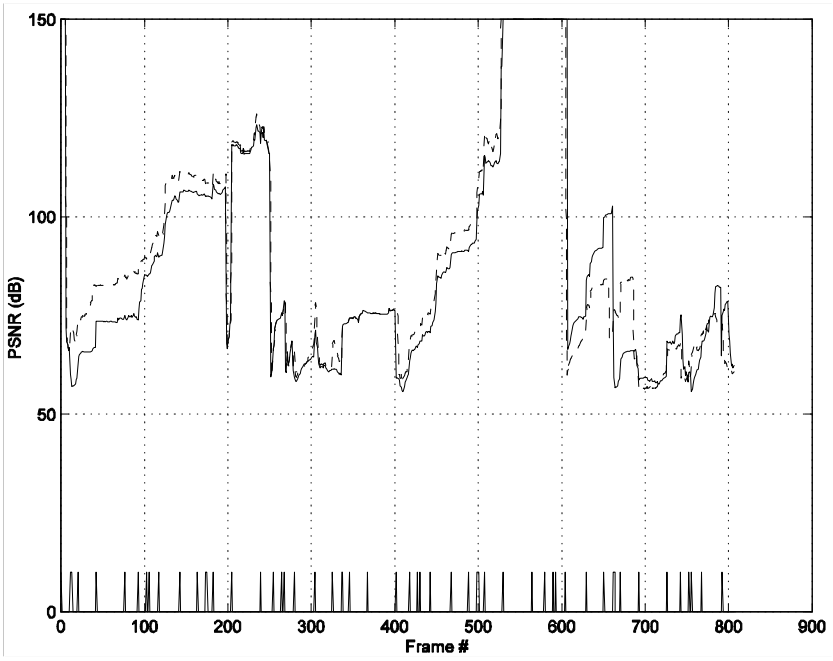
(c)

(d)

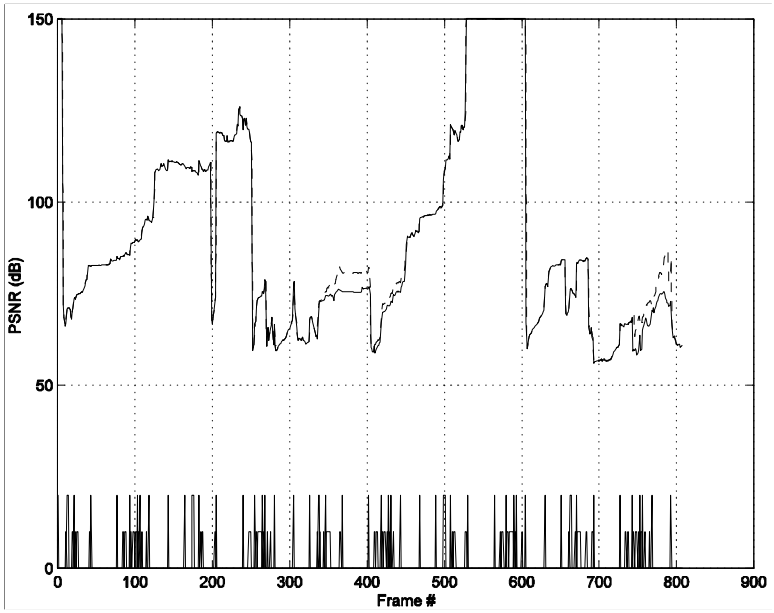
Figure 4.4. Results from different FEC allocation schemes. Channel BER without any FEC=5.1x10^{-3} on the average (a) Original sequence (b) no data segmentation (c) headers + P1 + P2 +P3 and (d) headers + P1 + P2 + P3 + motion.


PSNR (dB)
Frame #
150
140
130
120
110
100
90
80
70
60
50
0
100
200
300
400
500
600
700
800
900

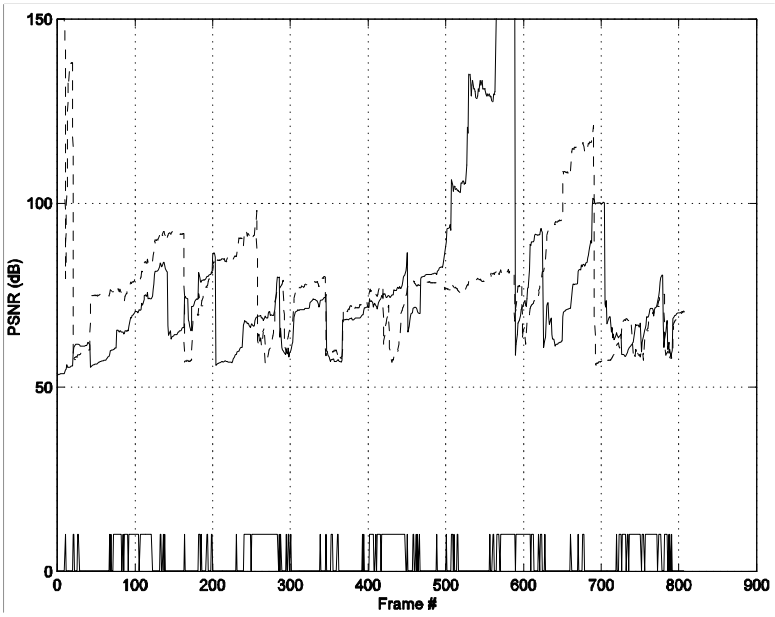
(a)


PSNR (dB)
Frame #
150
100
50
0
0
100
200
300
400
500
600
700
800
900

(b)



(c)



(d)

Figure 4.5: Incremental effects of various attributes. Channel BER without any FEC=5.1 x10^{-3} on the average. The PSNR values shown are between a decoded frame with errors and that without any errors (and not between a decoded frame with errors and the original frames with are encoded). Note that the scale (a) is different from that in (b) onwards.

4.5 Retransmission Based Schemes

4.5.1 Model

In this section, we present the content-based algorithms in relation to ARQ. The link between the first part (on FEC) of this chapter and the second part (on ARQ) is the common theme of using adaptive resource allocation according to video content. We first discuss the ARQ model, present the content-based schemes based on scene change and motion, describe an algorithm for solving the delay jitter problem caused by the proposed ARQ scheme, and finally present simulation results.

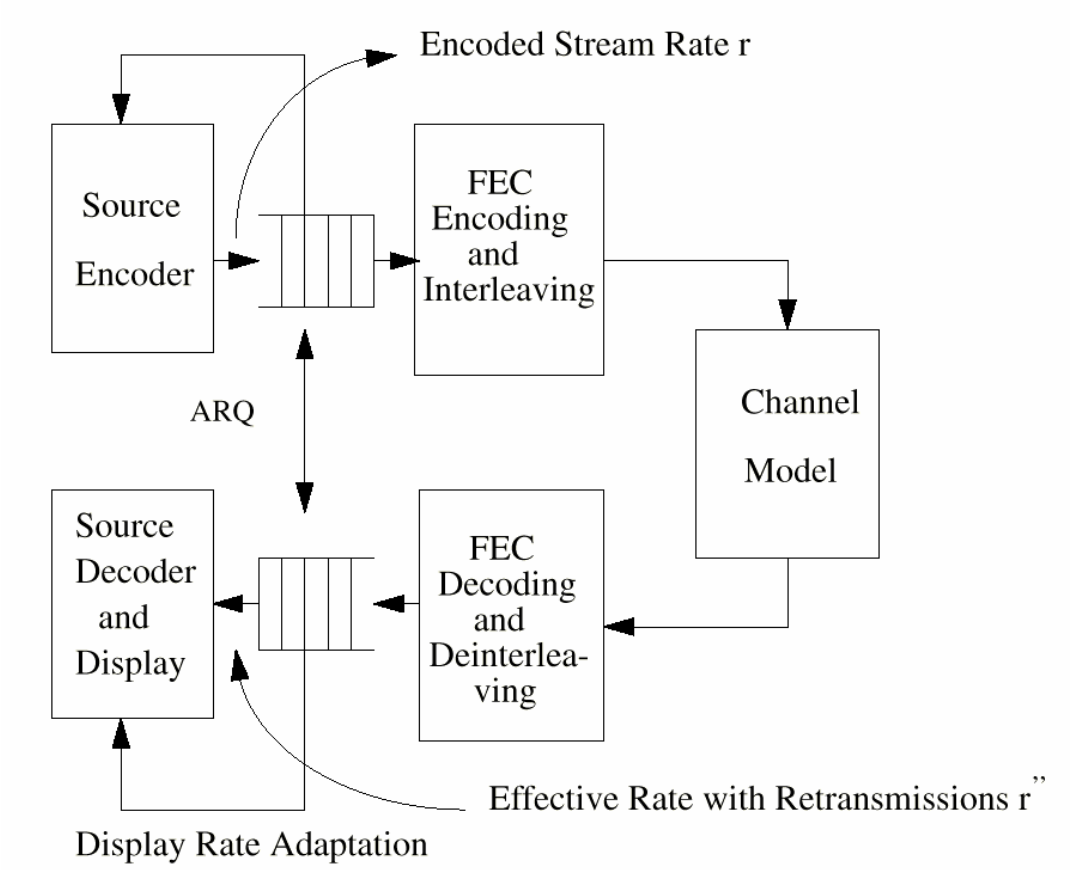


Figure 4.6 System diagram of the proposed ARQ-based schemes. The buffers shown are ARQ buffers.

A simplified system model is shown in Figure 4.6. We consider transmission across a single last

hop wireless channel. We model channel delay as a deterministic random variable. Incoming compressed stream is interleaved and FEC encoded before transmission through the simulated channel. The ARQ buffer at the sender stores frames that have been sent but not yet acknowledged. Its size is a function of the ARQ protocol being used and the window size[19]. Similarly, an ARQ buffer for storing (and, if need be for reordering) of frames is required at the receiver. The feed-forward loop is for display rate adaptation and its functionality will be discussed in later sections.

Sequence: Unrestricted CNN news sequence (including commercials).
Codec = H.263 [64]
Rate control: Online rate control with variable frame rate
Rate control ON for the entire sequence.
Resolution: QCIF (176x144 pixels).
Chroma format = 4:1:1
Motion estimation search window = 10 pels.
Encoded bit rate $\approx$ 24 Kbps (CBR)
Achieved frame rate = 6.4 fps (mean).
All enhancement options ON
Number of frames $\approx$ 106 sec (800 frames)
Other encoding specific parameters are as in TMN5

Table 4.4: Video parameters for ARQ simulations

Video encoding parameters used in ARQ simulations are indicated in Table 4.4. Unlike FEC simulations, the video is aggressively rate controlled using H.263 online-rate control (See TMN5 for details). This H.263 option produces a variable frame rate bit stream with skipped frames so that target bitrate constraints are met. We used this more aggressive rate control option (unlike that in FEC simulations) since our primary objective when studying ARQ is to contain decoder buffering requirements while meeting display constraints.

[19] The maximum number of en-route frames at any instant.

The optimal packet or frame size that should be used is in general a function of the data rate, channel rate, the ARQ protocol being used, the allowed latency etc. However, here we assume that a video frame is the packet entity under consideration. This gives us significant advantage in terms of speed, complexity, synchronization, packet handling and header integrity. Furthermore, for the above mentioned encoding parameters, the average video frame size is about 620 bytes which is a reasonably small size to be handled (ATM based transport uses a packet size of 188 bytes, which is of similar order of magnitude).

For 2-way interactive video, ARQ may not be suitable due to latency concerns. But, for 1-way video applications, the proposed scheme with retransmission as a last wireless hop recovery scheme can be used. Using ARQ for one way video playback is a complex task as has been shown in literature. We will elaborate more on issues regarding the usage of ARQ schemes in Section 4.5.3. Thereafter, experimental results justifying the approach are provided in Section 4.5.4. For publications related to these results, see [12].

4.5.2 Segmentation

We segment frames based on two types of basic video content - the extent of intra-coding (a, equivalent to scene changes), and the extent of motion. Since using ARQ involves latency buildups leading to shifts in the display axis, the idea is to identify a subset of frames which are more important. Only this subset uses ARQ based recovery. High-intra frames prevent error propagation. Being scene changes, they are also important for full subjective understanding of the video. Hence they are allowed one retransmission for recovery in case they are in error. Motion is the second cue used in segmentation. High motion frames are allowed one retransmission for recovery since the content changes significantly from the previous display instant and most error concealment schemes at the decoder may not work. As illustrated in Table 4.5, ARQ based recovery is used for frames of class C1 only and not for those belonging to class C2. Note further that a low-motion high a frame, if one exists, is retransmitted according to this Table.

Class	Criteria: Intra and motion
C1	High Intra (a e [0.4,1.0]) or High Motion
C2	Otherwise.

Table 4.5: Classes for ARQ

4.5.3 Analysis for adaptive decoder buffer control

4.5.3.1 Issues

Due to selective repeat ARQ, the CBR bit stream will suffer from rate variation at the receiver. Using the same notation as in [136], the buffer evolution of the encoder buffer is given by:

$$B_i^e = B_{i-1}^e + E_i - R_i$$

with

$$0 \le B_i^e \le B_{\max}^e$$

Here B_i^e is the encoder buffer occupancy at the end of frame period i, R_i is the channel rate for the same period, and E_i is the encoded rate for that period. The second constraint is required to prevent encoder buffer overflow or underflow. To prevent under/overflow of the encoder buffer, E_i and R_i should satisfy some constraints[20]. Rate control involves the proper selection of E_i (or equivalently the quantization parameter and frame-rate) and R_i so that these requirements are met.

Video transmission over networks normally assumes a fixed end to end delay between the encoder and decoder. The dynamics of the decoder buffer, is therefore, a time-shifted version of the above buffer dynamics, and is given by:

$$B_i^d = B_{i-1}^d - E_i + R_{i+L}$$

with

$$0 \le B_i^d \le B_{\max}^d$$

[20] Refer to [136] for details

Here we assume an end-to-end delay of L frame periods and $B_0^d = \sum_{j=1}^{L} R_j$ is the startup decoder buffering. For CBR video streams, it can be shown [136] that we can prevent underflow or overflow of the decoder buffer by preventing overflow or underflow at the encoder buffer. If the stream is VBR, then E_i and R_i must satisfy certain constraints to avoid buffer overflow and underflow in the encoder and the decoder.

The scenario we are considering is much more complex due to possible retransmissions. Hence, even if the video is aggressively rate controlled and the target constant bit rate is met, the display constraints may not be met. Or equivalently, the decoder buffer may underflow due to loss of time during retransmissions. Although it is possible to model retransmissions delays as simple network delay jitters and modify L to $L+\Delta$[21], the scheme has its drawbacks. Firstly, errors normally occur in bursts in wireless channels, and so the order of Δ we wish to de-jitter is very large. This may require large initial startups or initial buffering to prevent underflow. Secondly, the channel characteristic is time varying and is not known a-priori, and in general cannot be pre-negotiated (although channel dependent renegotiation is a possibility).

4.5.3.2 Analysis and Algorithm

The retransmission scheme we use is a modification of the Go-Back-W scheme with selective retransmission of erred packets. W, the window size, indicates the maximum number of outstanding packets sent out by the sender that have not yet been acknowledged by the receiver. The receiver sends bundled ACK/NAKs at the end of the window and only frames in error are retransmitted.

We focus on a single retransmission period, i.e., W frames are sent and B of these are assumed to be in error and are retransmitted. Assume that t_{prop} is the end to end propagation delay, r_{in} (indicated by R_i above) is the rate of input to the decoder buffer, r_{out} (indicated by E_i above) is the output rate from this buffer, B_{start} is the decoder buffer occupancy prior to the retransmission

[21] This suggestion was made in [136] for channels having variable delay (i.e., with a jitter term added to a mean value).

period, B_{end} is its occupancy at the end of this period, and T_{max} is the maximum frame transmission time. Then, assuming that $2t_{prop}$ goes idle during ACK/NAK[22], we get

$$B_{end} = B_{start} + WT_{\max} r_{in} - (WT_{\max} + 2t_{prop} + BT_{\max}) r_{out}$$

We are concerned with the extent of imminent buffer underflow due to retransmissions. Therefore, to a first approximation, we neglect variations in r_{in} and frame transmission times E_i. The analysis is easily extendible to incorporate these variations. In fact, these give us further ways to prevent underflow by changing the source coding (changing E_i) or renegotiating with the channel (changing r_{in}; see [69] for details).

Assuming that the buffer occupancy was oscillating around its nominal value prior to the retransmission period, the extent of buffer underflow due to retransmissions is:

$$U = (WT_{\max} + 2t_{prop} + BT_{\max}) r_{out} - WT_{\max} r_{in},$$

with

$$0 \le B \le W$$

Again, to a first approximation, and for time scales in operation, we assume $r_{out} = r_{in}$.

Hence,

$$U = (2t_{prop} + BT_{\max}) r_{out}$$

To accommodate this imminent underflow, we propose to slow down the rate of display[23]. If we choke the output rate r_{out} by an amount Δr, then the buffer will return to its nominal value after a time:

$$T = \frac{(2t_{prop} + BT_{\max}) r_{out}}{\Delta r}$$

For the scenario under consideration, $t_{prop} << BT_{\max}$, and $\Delta r / r_{out}$ can also be considered as fractional clock slow-down at the receiver. To simplify the display modification, we assume that slow down in display is possible only using frame skips (i.e., by repeating frames in the display). Thus, the average number of display instants to be spread out or the total necessary shift in the display axis is,

[22] For simplicity, we do not consider continuous ARQ protocols. Further, we assume that the first erred frame of a maximum of W buffered up is passed to the decoder only after its retransmission is received. Later packets have to be buffered until then to avoid reordering problems.

$N_s = T / D$

where D is the average frame interval (e.g., 34 msec. for 30 Hz.). This shift is in addition to any frame-skips in the encoded stream itself (which is variable frame rate in this case). The above assumes that the network adaptation layer provides the necessary timestamp support for synchronization. Other de-jittering mechanisms to absorb slight variations in network delay [42][107] (e.g., end-system traffic shaping or smoothing) can be applied in addition to the scheme mentioned here.

In our study on ARQ, the first place we used visual content was in deciding which frames to allow retransmissions for. The other place we use current frame content is in deciding how the decoder buffer spreads the anticipated buffer underflow. That is, the manner in which the N_s instants are spread out in time differs depending on content. The content that we consider here is motion. If the current frame to be displayed has high motion then the display axis is shifted slowly to avoid jerks. But, if it has low motion, the entire shift is given instantaneously. Notice that the receiver knows N_s after receiving W frames, and does not have to wait for the retransmitted B to start shifting the display axis.

As an illustration, we give below sample code for a single window size i.e., $W = 1$ clock rate=30 Hz. Important observations on the code are also mentioned.

Pseudo code.

Step 0. $B_{init}^{d} = first\_frame\_size + \overline{m} + (s / 2)$ where $\overline{m}$ is the mean frame size and s is the standard deviation in frame size. Let i be the frame at the head of the decoder buffer,

Case (a). if $(display_i - arrival_i) < Threshold$ or $(low\_motion)_i$ then

$display_i = (display_i)_{orig} + (N_s \times 34)$

---

[23] This requires much less latency as compared to re-negotiating R_i (since MAC layer delays are quite high for the current system) or changing E_i. However, video quality may suffer from jerks. Our objective hereafter is to minimize the perception of these jerks.

Case (b). if $(display_i - arrival_i) > Threshold$ and $(high\_motion)_i$, then

$(display_{i+k-1} = (display_{i+k-1})_{orig} + \max(\lceil N_s / 2^k \rceil, 1) \times 34$ for $k = 1, 2, 3, ..., K_{\max}$

where K_{max} is the minimum number satisfying $\sum_{k=1}^{K_{\max}} \max(\lceil N_s / 2^k \rceil, 1) > N_s$

Observations on Step 0.

This total startup buffering is useful if we shift the display axis slowly as in Case(b). In other words, $\overline{m} + s/2$ is a cushioning term for slow shift in Case (b).

Observations on Case(a).

The shift in the display axis is instantaneous. This is because we can tolerate jerks with low motion. The first condition is used as an emergency measure and *Threshold* is set at $\overline{m} + s/2$.

Observations on Case(b).

It takes logarithmic time to provide the necessary shift to the display axis, or $K_{\max} \le \log_2 N_s$. In this case, we do not wish to accommodate the entire shift on one frame since the frame has high motion and we wish to avoid jerks. The amount of shift decreases exponentially and is most at time instant *i* (equal to half of the total required). This is adopted since errors normally occur in bursts. By doing this, the effect of tails on following frames falls off fast, and we do not get big carry over terms from previous frames in case of a burst of errors. Formally, the effect of these terms at a given frame is bounded from above by slippage required by the maximum sized frame.

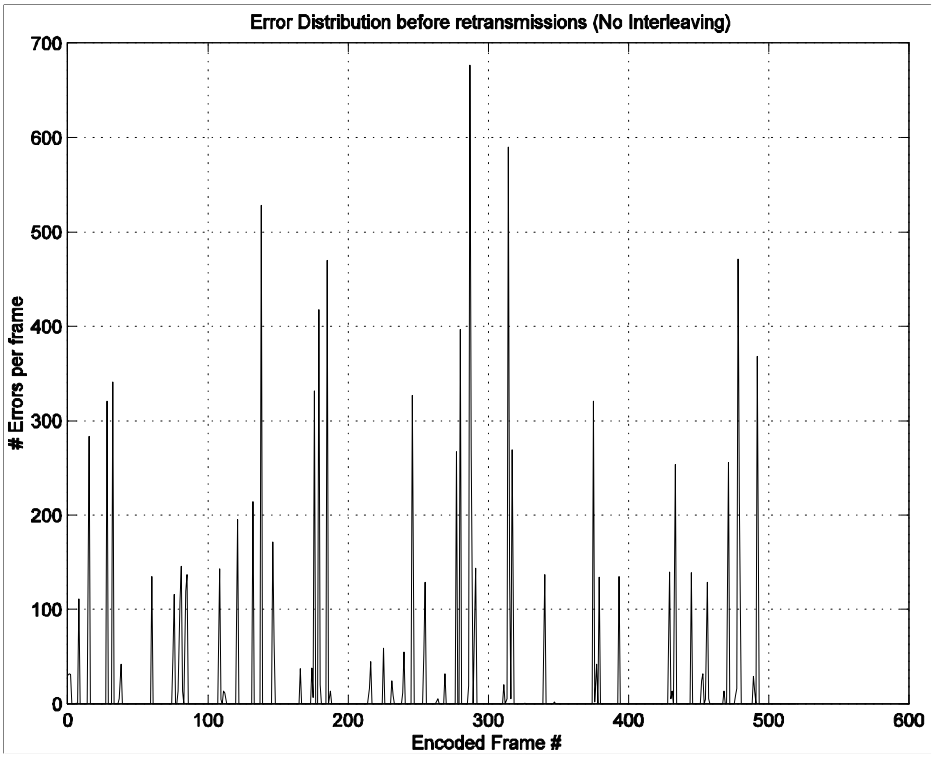
For example, if *W*=1, and we consider a frame of size $\overline{m} + s/2$ = 829 bytes, its transmission time is about 276 msec. Hence, N_s = 6+276/34 = 9 skips. Depending on the content of the currently being displayed frame, this shift would be given instantaneously (Case a) or over $K_{\max} \le \log_2 N_s$ = 4 to be displayed frames (Case b). We will now discuss results obtained for this algorithm in the following section.

4.5.4 ARQ Results

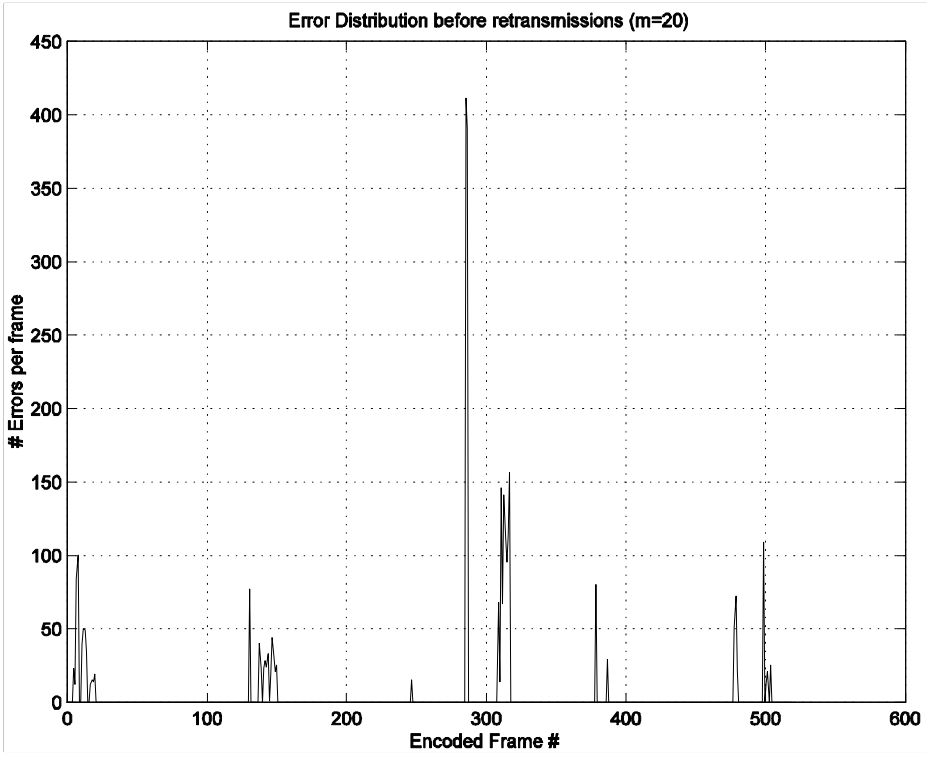
The ARQ scheme we used is also known as Type-I ARQ. In Type-I ARQ, FEC is applied to all frames and ARQ is used for erred frames after FEC decoding. The analysis was carried out assuming header+a based FEC. We used $t(a_{high}) = 6,\ t(a_{medium}) = 6,\ t(a_{low}) = 5,\ t(a_{header}) = 9$. Different degrees of interleaving are used to vary the effective burst lengths. Figure 4.7 shows the error distribution after FEC correction, but without any retransmissions. It can be seen that even if we make binary decisions at a frame level, errors normally occur in bursts.

Figure 4.8 shows the number of frame errors after retransmissions. It can be seen that a significant improvement can be obtained by using ARQ (compare with Figure 4.7). Implicit in Figure 4.8 is the assumption that late packets are useful. This we achieve by reducing the effective frame display rate at the receiver nominally using the algorithm mentioned above.

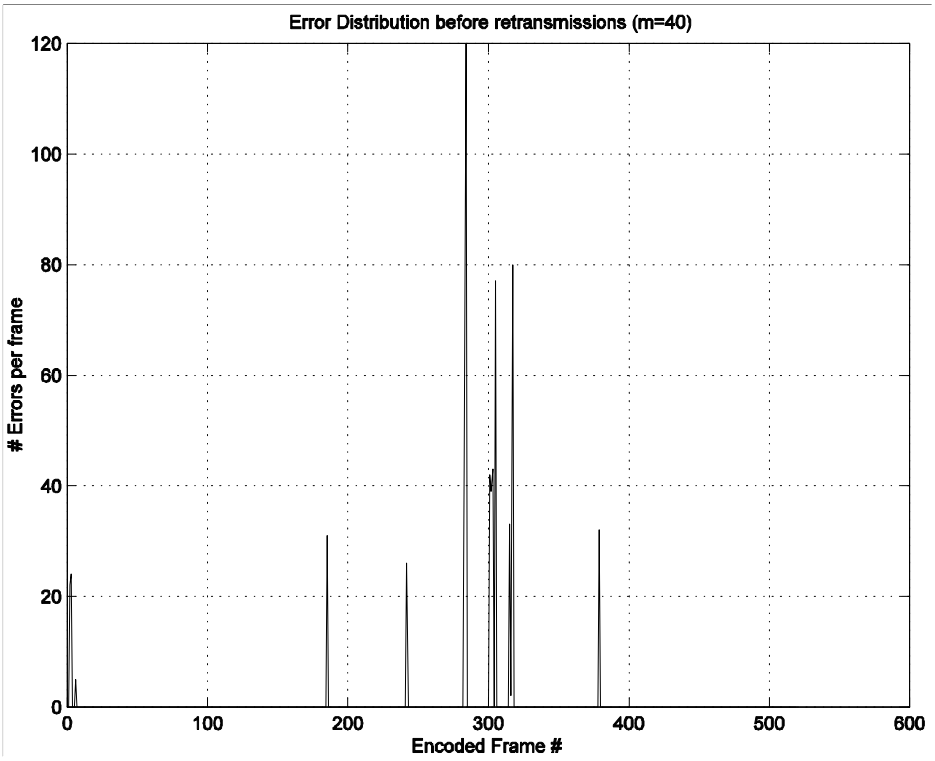
Figures 4.9(a) and (c) show the arrival and display deadlines without any shift to the display axis. An alternate (equivalent) way to validate the above algorithm would be to plot the receiver buffer occupancy as a function of frame number. In Figure 4.9, the dashed line shows the target display deadline and the solid line is the arrival time. It can be seen that the display deadlines cannot be met if we use ARQ without any shift to the display axis.


Error Distribution before retransmissions (No Interleaving)
Errors per frame
Encoded Frame #
0
100
200
300
400
500
600
700

(a)


Error Distribution before retransmissions (m=20)
Errors per frame
Encoded Frame #
0
50
100
150
200
250
300
350
400
450

(b)

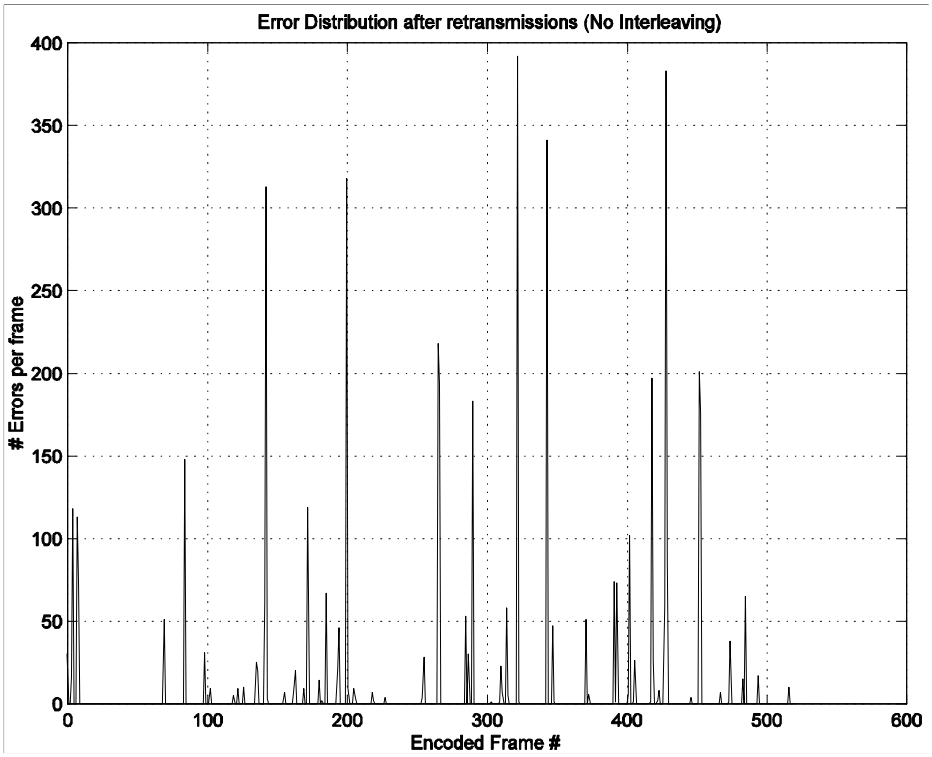


(c)

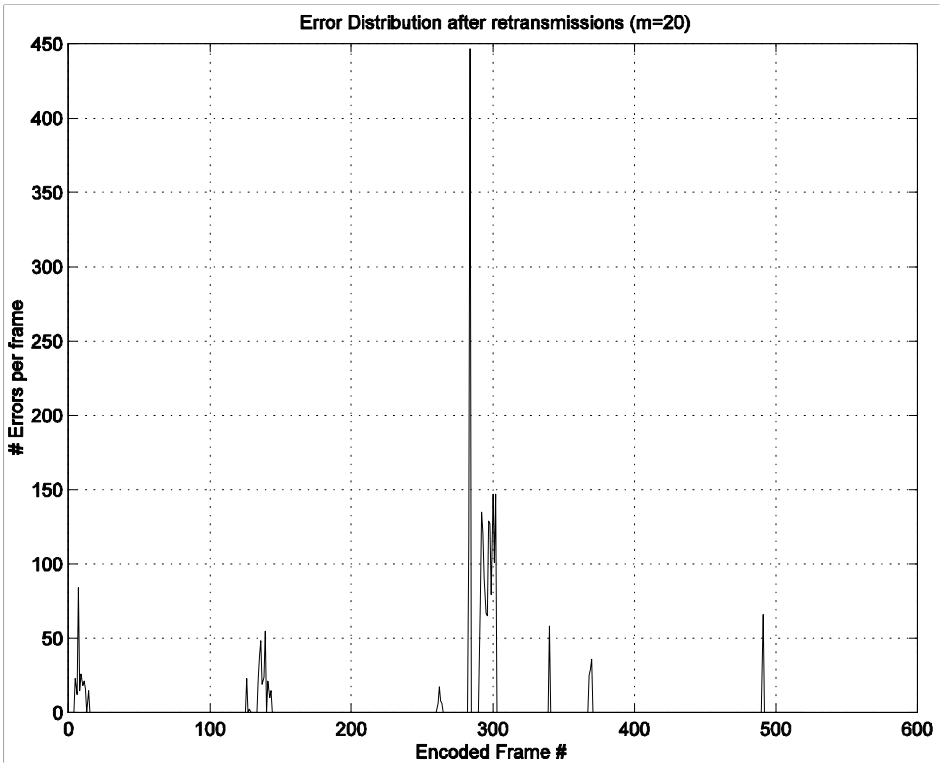
Figure 4.7 Error distribution before retransmissions. Channel BER (without any FEC) = 5.23×10^{-3}. Errors shown are in the encoded video stream. (a) No interleaving (b) interleaving degree = 20, and (c) interleaving degree = 40.

We also show the error distribution in Figure 4.9(c) to illustrate how retransmissions cause display deadlines not to be satisfied. Also worth noticing is that the time it takes for arrival times to start falling behind target display time depends strongly on the extent of interleaving and error protection used.

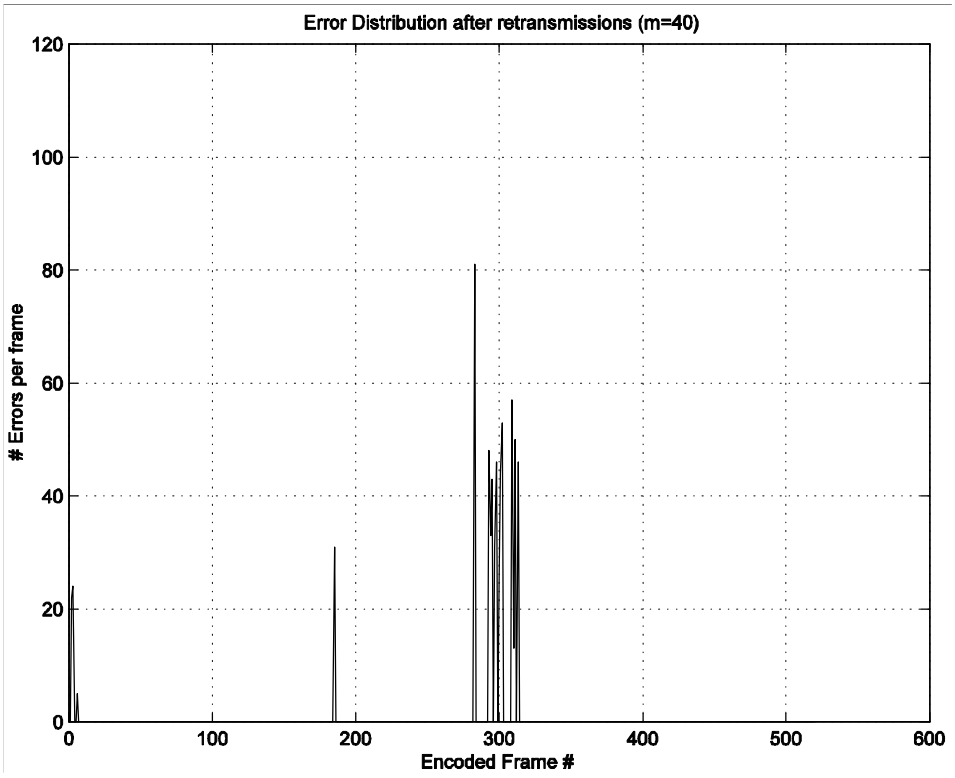
Figures 4.9(b) and (d) show the arrival and display deadlines with shifts to the display axis according to the algorithm above. We note that the arrival curve is maintained below the display curve in the long run. By giving an elastic content-dependent shift to the display axis we are able to meet display deadlines while maintaining good visual quality.


Error Distribution after retransmissions (No Interleaving)
Errors per frame
Encoded Frame #
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600

(a)

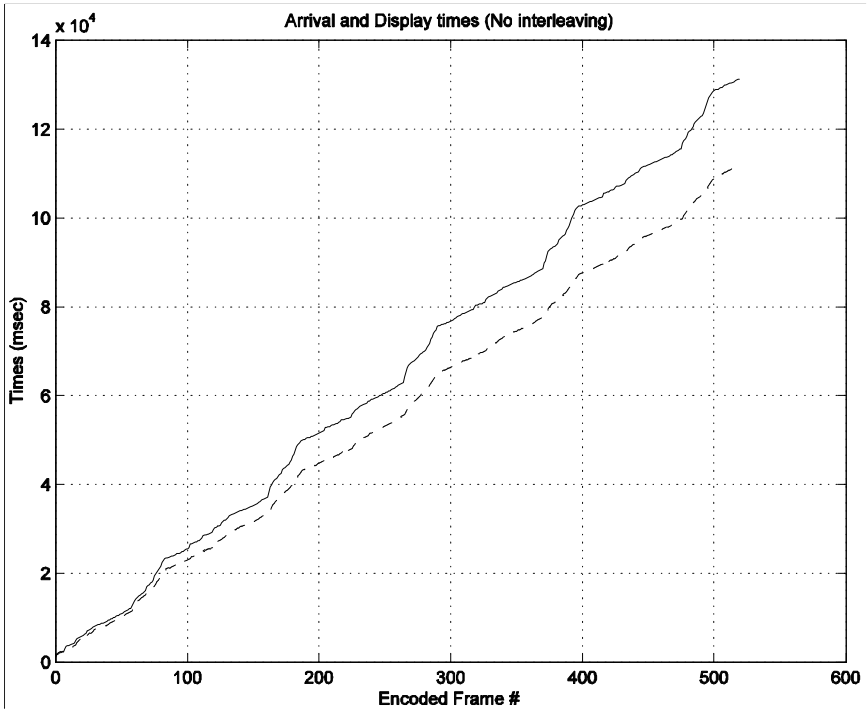

Error Distribution after retransmissions (m=20)
Errors per frame
Encoded Frame #
0
50
100
150
200
250
300
350
400
450
0
100
200
300
400
500
600

(b)

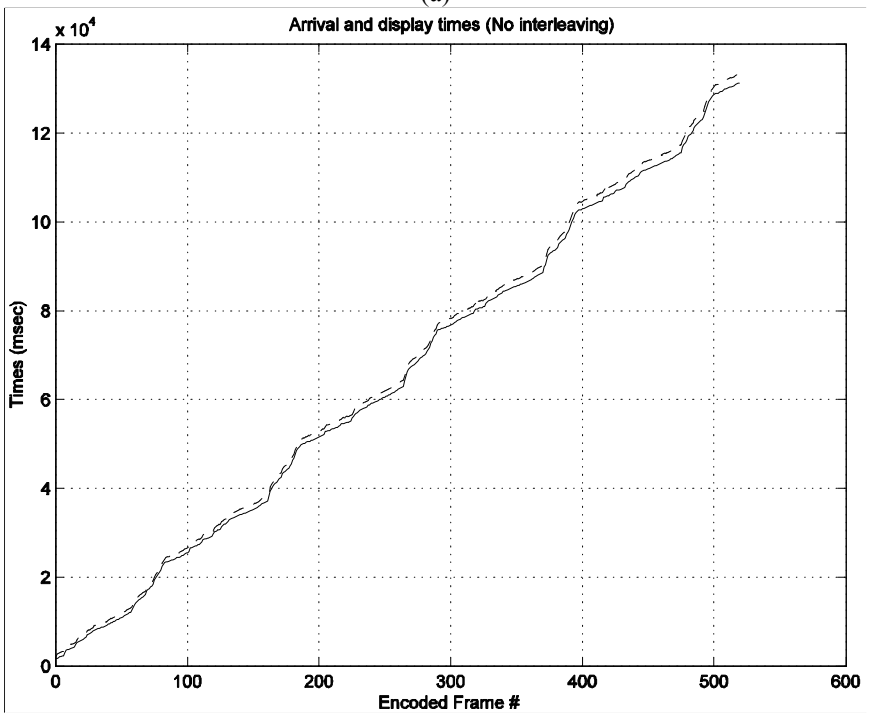


(c)

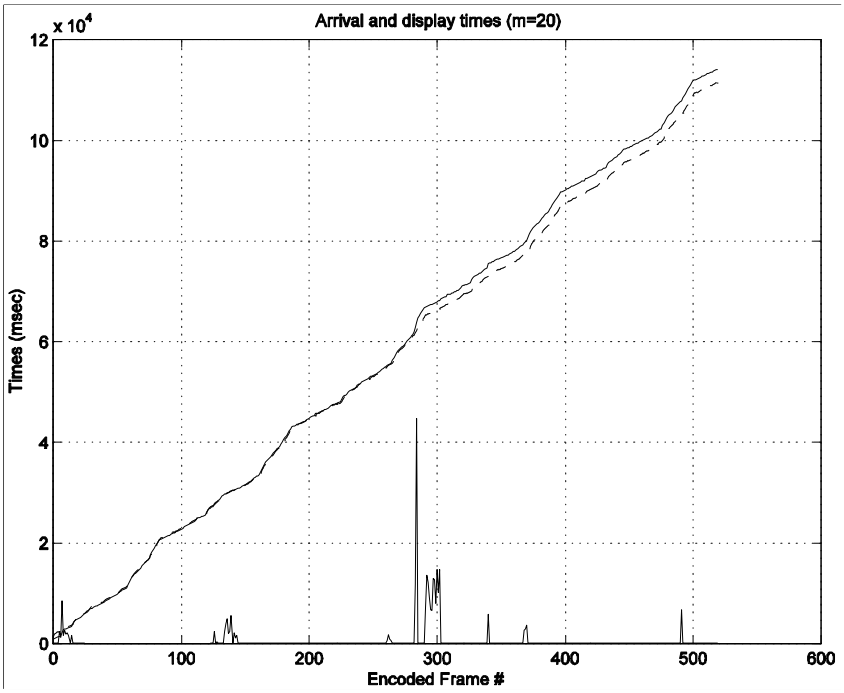
Figure 4.8 The number of frames in error after retransmissions. Channel BER (without any FEC) = 5.1×10^{-3}. Errors shown are in the encoded video stream. (a) No interleaving (b) interleaving degree = 20, and (c) interleaving degree = 40.


Arrival and Display times (No interleaving)
x 10^4
Times (msec)
Encoded Frame #
0
2
4
6
8
10
12
14
0
100
200
300
400
500
600

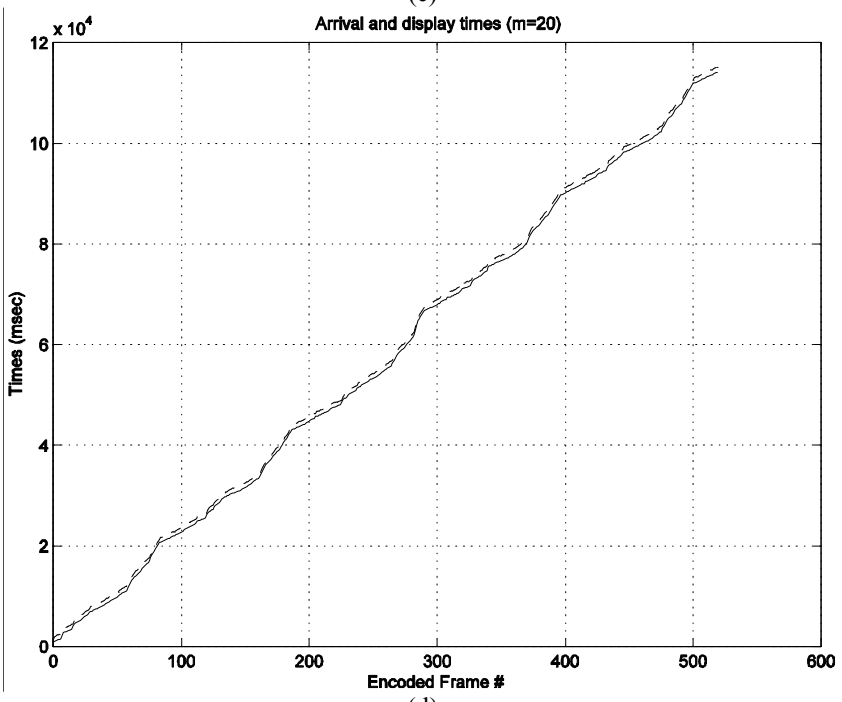
(a)


Arrival and display times (No interleaving)
x 10^4
Times (msec)
Encoded Frame #
0
2
4
6
8
10
12
14
0
100
200
300
400
500
600

(b)



(c)



(d)

Figure 4.9 Arrival and display deadlines. Window size = 1. Dashed-display, Solid-arrival times. (a,c) without rate adaptation and (b,d) with rate adaptation; (a,b) no interleaving; (c,d) interleaving degree=20.

4.6 Summary of the chapter

In this chapter, we presented a new approach for adaptive transmission of video over wireless channels. We proposed a scheme for segmenting video into classes of various perceptual importance. We presented a detailed mechanism for mapping between segmentation and resource allocation. The proposed approach can be used as an enhancement to the currently available scalability profiles, both at a frame and future video objects (as defined in MPEG-4). In particular, the extent of intra coding, scene changes, and motion based procedures are used to provide finer level of control for data segmentation. We present a comprehensive analysis on the incremental effect of these attributes using a practical mobile and wireless channel simulator. Based on our FEC results, we see that header and high intra (or scene change) based segmentation gives the largest increase in PSNR. The extent of intra coding (Partitions P2 and P3) and motion information can be used to further increase the perceptual video quality. In the second half of the chapter, we present algorithms using selective repeat ARQ for one-way video applications. These algorithms used visual content to identify a subset of frames which could be retransmitted. To accommodate the variable delay caused by the selective use of ARQ schemes, our approach suggests using an "elastic" buffer before the decoder/display module to "absorb" the delay jitter. The buffer absorbs the rate variation caused by selective ARQs and provides a compatible interface to the decoder. Our method for buffer control (slippage) also uses visual cues to maintain good visual continuity white preventing decoder buffer underflow.

The proposed framework using video content provides great synergy to recent video coding standards, e.g., MPEG-4, that use object-based video representations. Based on our results we argue that introduction of video content into video segmentation allows us to define more semantic classes of importance. We believe that by using such techniques we can get better visual quality under given resource constraints. For publications related to this chapter, see [12].

In Chapter 2 of this thesis we discussed how to represent logical inter-dependencies among objects that compose a presentation. Although this can be done easily, the resulting integer problem may become extremely difficult to solve. As a way around it, we proposed to use LP relaxations, and froze the variables that attained integral values in the optimal LP solution at their

respective values. This was followed by an exhaustive search for best solution over fractionally assigned variables. The limited experiments that we conducted gave very encouraging results for the dependency structure considered. However, as an approach, this is certainly not a generic way, and may never even lead to a feasible integral solution. In the next chapter, we revisit this issue by restricting our attention to some special dependency structures. We will present ways to model predictive inter-dependencies in motion-compensated coders. We also discuss ways to solve this problem and some of its natural extensions.

CHAPTER 5

Alternative formulations for bit allocation with dependent quantization

---

Contents

5.1 Introduction

A common problem that arises in image and video compression is the so-called bit allocation problem [71][131][140][141][144][158]. In most Discrete Cosine Transform (DCT) based

compression standards, an image or video frame is divided into sets of 8×8 pixels that are called blocks. Pixels in each block are transformed using DCT that does an approximate principal component analysis on the data. The DCT coefficients are then quantized, run-length encoded, and Huffman coded to generate the final bitstream. The main distortion that gets introduced in this process and hence compresses data is due to quantization. Normally, coding standards provide quantization matrices that may be scaled up or down to choose quality for each macroblock or a slice of macroblocks. These scales must be chosen from a few (around 32 to 64) discrete levels. The objective is to choose the scale values for each macroblock so as to minimize global distortion in a frame under an overall bit-rate constraint. In this chapter we address this bit-allocation problem for video coding using mathematical programming.

5.1.1 Related Work

The bit-allocation problem has been studied earlier in the context of image and still texture compression. In its discrete form (i.e., when the scale factor of the quantization matrices should be chosen from a specified discrete set), the bit allocation problem can be formulated as a multiple choice knapsack problem (MCKP) [148]. This is a well-studied problem in the operations research literature. In this variation of the knapsack problem, there are some disjoint sets of objects; objects in each of these sets have values and sizes associated with them The objective is to choose exactly one object from each set without violating the knapsack size so as to maximize the sum of the values of the chosen items. This is an NP-hard problem, but can be solved relatively efficiently, both in theory and practice. From a practical point of view, several good heuristics have been discovered for MCKP. One class of heuristics is based on solving the underlying linear programming (LP) relaxation [15][18][44][110][165], and *rounding* the (possibly) fractional solution obtained to an integral one. Such rounding heuristics construct solutions with values that are provably within a factor of 2 of the best possible; perhaps more importantly, such heuristics perform extremely well in practice [15]. Several studies have also looked at Lagrange relaxations for solving bit allocation problems (see for example, [52][53][54][55][144]). These usually guarantee gaps no better than the LP relaxation and (at least in the above cited works) are computationally more intensive. From a theoretical point of view, MCKP is known to admit a polynomial-time approximation scheme [30]: thus, for any

fixed e, a solution with value that is within a factor of (1+e) of the best possible can be computed in time that is polynomial in the input size. This algorithm is not usable in practice but, for instance, could be used to quantify the complexity-quality trade-off in a tunable way [13]. Moreover, since the guarantees are true irrespective of problem instance, in compression terminology, this leads to a lossy-universal quantization (compression) scheme.

While the MCKP formulation can be used in applications involving image and still texture compression, it is not directly applicable to video coding. This is because video coding further compresses data by exploiting strong inter-frame correlation using motion compensation. Typically, in block or optical flow based compression, pixel values in future macroblocks are predicted using decoded pixel values in an earlier frame. The prediction error for the current macroblock is then encoded using DCT, quantization, run-length, and Huffman coding. In addition, a motion vector that identifies the location of the predicting block is sent in the bitstream. Because of its predictive nature, the quantization choices we make for a particular frame M (predicted macroblock) will now depend on how we quantized frame M-1 (predicting macroblock). This leads to an exponential growth in the number of states of the *trellis* used to solve the problem. In general, for a given quantization choice for the predicted macroblock, the rate distortion behavior of the predicted macroblock worsens as predicting macroblock is quantized more coarsely. However, counter claims about this monotonicity assumption have also been reported in literature [34]. This behavior is usually attributed to non-linearities in encoders and source statistics and the manner in which codecs make INTRA-INTER coding decisions. To some extent this can also be caused by the non-convex R-D behavior the predicting macroblock. Because of all the above reasons, the problem with inter-frame dependencies turns out to be a much more difficult problem to solve.

5.1.2 Results

This chapter discusses a new approach to solve the bit-allocation problem with dependencies. Our primary contributions are the following:

- We discuss several integer-programming (IP) formulations for the bit-allocation problem with dependencies. These formulations are more general and easier to solve than those discussed

in earlier works [45][131]. Unlike [131], we don't use monotonicity assumption, either in the formulation stage or to simplify solving the problem. Further, unlike this work, we don't use dynamic programming [20] and instead focus on LP and Lagrange relaxations with randomized rounding. These are more efficient and require less memory than dynamic programming.

- We show how to relax a so-called *causality* assumption [45] . The causality assumption allows one to pre-quantize the predicting frame prior to finding the best choices for the predicted macroblocks. We study the incremental improvement obtained by incorporating for dependencies explicitly in the model. For simplicity, we restrict our attention to first order backward and forward dependencies. The drift due to the recursive nature of prediction beyond a single frame (both forward and backward) is not explicitly accounted for.
- Integer problems may end up being very difficult to solve. As an approximate way to solve our IP formulation, we discuss the usage of linear and Lagrange relaxations along with randomized-rounding. We present experimental results on real data to validate the performance of these rounding methods and observe that the rounding algorithms perform extremely close to optimality. Results also show an improvement of about 0.5 dB over the causally optimal solution. From a practical standpoint this validates that the causality assumption is a very justified one in reducing problem complexity for a marginal loss of optimality.

The balance of this chapter is organized as follows. We formulate and analyze the problem in Section 5.2. Here, we also give several algorithms to approximately solve the problem. Section 5.3 gives an experimental study on the performance of the suggested algorithms. We conclude in Section 5.4 with suggestions for future work.

For publications related to this chapter, see [16][17].

5.2 Problem formulation and analysis

We describe two formulations for the dependent quantization problem – a disaggregated one **(F1)** and an aggregated one **(F2)**. Considering the debatable nature of the aforementioned monotonicity condition, none of the formulations we discuss require assumptions on convexity of rate-distortion curves of predicting macroblocks or that of the predicted one conditioned on

specific quantization choices for the predicting one. Throughout the analysis we assume a first order backward dependency (as with P frames).

5.2.1 Disaggregated formulation (F1)

We wish to jointly minimize the distortion over two frames under a global rate constraint. By restricting to 2 frame problems, we bound the number of states. A finite width *beam,* while compromising global optimality, prevents the number of states from growing exponentially as the degree of dependency increases. After solving a sequence of joint 2-frame problems, the heuristic given in Section 5.2.2 is used to reconcile the results and determine final the quantization choices. This helps us study the incremental improvement obtained by accounting for dependency effects explicitly in the model.

Using the notation and variables given in Table 5.1 below and in Figure 5.1, the joint frame *M-1* and frame *M* problem is given by:

(Formulation F1)

$$\min\{ \sum_{k,l} d_k(l) X_k(l) + \sum_{k,l,i,j} d_{i,k}(j,l) Z_{i,k}(j,l) \}$$

Subject to:

$$(C1) \quad \sum_{l} X_k(l) = 1 \quad \text{for } k = 1,..,n$$

$$(C2) \quad \sum_{k,l,j} Z_{i,k}(j,l) = 1 \quad \text{for } i = 1,..,n$$

$$(C3) \quad Z_{i,k}(j,l) \le X_k(l) \quad \forall i,j,k,l$$

$$(C4) \quad \sum_{k,l} r_k(l) X_k(l) + \sum_{i,j,k,l} r_{i,k}(j,l) Z_{i,k}(j,l) \le R$$

$$(C5) \quad X_k(l) \text{ and } Z_{i,k}(j,l) \in \{0,1\} \ \forall\ i,j,k,l$$

Variable	Interpretation
$X_k(l)$ $k=1,...,n$ and $l=1,...n_k$	1 if macroblock k in frame *M-1* is quantized at level l. 0 otherwise.
$Z_{i,k}(j,l)$ $k=1,...,n$; $l=1,...n_k$ $i=1,...,n$ and $j=1,...n_i$	1 if macroblock i in frame M is quantized at level j assuming that it is predicted using macroblock k in frame *M-1* encoded at level l. $Z_{i,k}(j,l)$ is 0 otherwise.
R	Global bit-rate bound over both frames
$d_k(l)$ $k=1,...,n$ and $l=1,...n_k$	Distortion for choice *(k, l)*
$r_k(l)$ $k=1,...,n$ and $l=1,...n_k$	Bits used for choice *(k, l)*
$d_{i,k}(j,l)$ and $r_{i,k}(j,l)$ $k=1,...,n$; $l=1,...n_k$ $i=1,...,n$ and $j=1,...n_i$	Distortion and bits used if macroblock i in frame M is quantized at level j assuming it is predicted using macroblock k in frame *M-1* encoded a level l.

Table 5.1 Variables used.

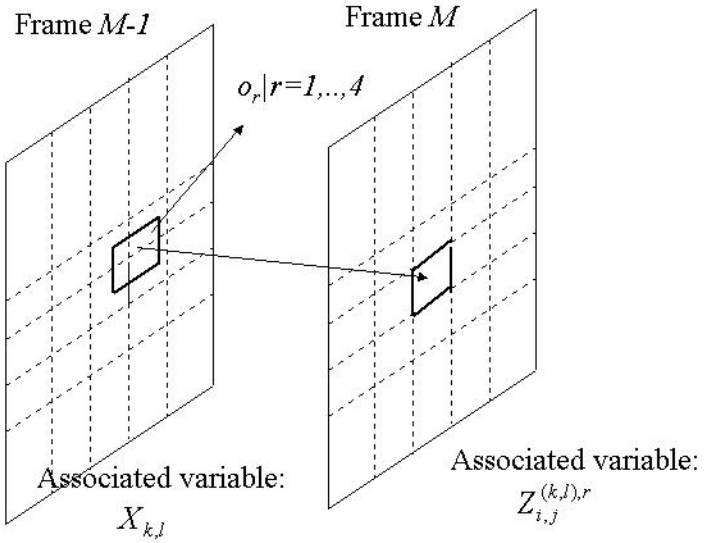


Figure 5.1: Dependent quantization framework. Ignore the subscript *r* and the term o_r | *r=1,..,4* for boundary aligned predictor.

Note that:

1) For notational convenience we suppress repetitive summation signs. The summations run on all the indicated variables, and bounds on them are as given in the table above. Constraint C1 ensures that each macroblock in frame *M-1* is encoded at one level. Constraint set C2 imposes that each macroblock in frame *M* is encoded at one level and is predicted from exactly one macroblock in frame *M-1*. Constraint C3 ensures that $Z_{i,k}(j,l)$ can be 1 only if we chose to encode macroblock *k* in frame *M-1* at level *l*. This captures the dependency in quantization choices of successive frames. Constraint C4 is the global bit-rate constraint over the two frames. Note that by ignoring terms having $Z_{i,k}(j,l)$ from **F1** we get the standalone problem for frame *M-1*. Obviously, this is a pure MCKP.

2) Dependency effects in MC-DCT coders come in two forms. First, the search for best motion vectors depends on quantization choices in a previous frame. This is because most encoders use reconstructed previous frames for motion estimation. This implicitly enforces a causality constraint by assuming that all previous frames have been quantized. Second, given a motion vector, one can still search for the best quantization choice for jointly encoding the predicting macroblock and the prediction error. This step, however, would implicitly assume that the motion vector does not change in the process (perhaps by doing motion estimation on raw frames). In reality, both the above forms of dependency are tightly inter-coupled. Our formulation captures the second dependency. We assume that motion estimation is done on raw frames, and try to determine the effect of future and historical dependencies in quantization choices.

3) Formulation **F1** makes the implicit assumption that predicting pixels lie strictly on macroblock boundaries in frame *M-1*. In general this is not true since the predicting pixels could overlap with at most four aligned macroblocks in the previous frame. A first cut approach to incorporate this could be by weighting appropriate terms using the proportion o_r of area overlapped with aligned macroblocks. This would still retain the same problem structure. Normally the search space for motion vectors has a very high cardinality. Since the above problem would have to be solved for all such choices of $\{o_r | r=1,..,4\}$, and to relieve an

impractically large effort in gathering operational data, we will focus exclusively on **F1** in what follows.

4) An aggregated formulation can be obtained by replacing constraint (*C3*) in **F1** by (*C3'*). The other constraints stay the same.

(Formulation F2)

$$\min\{\sum_{k,l} d_k(l)X_k(l) + \sum_{i,k,j,l} d_{i,k}(j,l)Z_{i,k}(j,l)\}$$

Subject to:

(C1), (C2), (C4), (C5), and

(C3') $\sum_{i,j} Z_{i,k}(j,l) \le (\sum_i n_i)X_k(l) \ \forall\ k,l$

F2 is correct and is equivalent to **F1** since,

1. If $X_k(l) = 1 \Rightarrow \sum_{i,j} Z_{i,k}(j,l) \le \sum_i n_i$.

This does not constrain $Z_{i,k}(j,l)$ un-necessarily.

2. If $X_k(l) = 0 \Rightarrow Z_{i,k}(j,l) = 0 \ \ \forall\ i, j$ corresponding to choice (k,l). This is true due to constraint $(C3')$.

Both formulations **F1** and **F2** share the same set of variables. There are a total of $\sum_{k=1}^{n} n_k + (\sum_{k=1}^{n} n_k)(\sum_{i=1}^{n} n_i)$ binary variables and $2n + (\sum_{k=1}^{n} n_k)(\sum_{i=1}^{n} n_i) + 1$ constraints in **F1**. **F2,** on the other hand, has a much smaller number of constraints ($2n + 1 + \sum_{k=1}^{n} n_k$). Note that summing *(C3)* over indices *i, j* leads to *(C3')*. Thus, if there is a point in the polyhedron defined by constraints in **F1,** it also lies inside that of **F2** (or Poly(F1) $\subseteq$ Poly(F2)). Alternatively put, although the specification of polyhedron of **F1** needs many more constraints, it boxes the integral points more compactly. As a consequence, if we were to use an iterative scheme (like branch and bound or branch and cut) to solve the problem, formulation **F1** is expected to lead to faster convergence. Consider next the LP relaxation of **F1**. Due to constraint sets *(C1)* and *(C2)* upper bounds on binary variables may be ignored in the LP relaxation. Let f_1 equations in constraint set *(C1)* have fractional positive terms. Hence n-f_1 of them will have pure integral variables. Similarly define

f_2 and n-f_2 for constraint set *(C2)*. Each equation having fractional terms will have at least two fractional variables. Thus, the total number of fractional variables is $\geq 2f_1 + 2f_2$. The number of integral variables for both *(C1)* and *(C2)* is $2n - f_1 - f_2$. Further, the number of non-negative variables in an optimal basic feasible solution (BFS) of a linear program is bounded above by the number of constraints in the problem [18]. Hence we get,

$$2n + 1 + (\sum_{i=1}^{n} n_i)(\sum_{k=1}^{n} n_k) \geq \#(\text{fractional + integral vars.}) \geq (2f_1 + 2f_2) + (2n - f_1 - f_2).$$

$$\text{Combining, } f_1 + f_2 \leq 1 + (\sum_{i=1}^{n} n_i)(\sum_{k=1}^{n} n_k).$$

A similar counting argument on **F2** gives that $f_1 + f_2 \leq 1 + \sum_{k=1}^{n} n_k$. Therefore, although **F1** bounds the feasible integral points more tightly, a solution to its LP relaxation has many more fractional variables than **F2**. Hence, it is possible that a lot fewer branches need to be explored when solving **F2** by branch and bound.

5.2.2 A typical run over an image sequence

The ultimate solution to the dependent quantization problem has to find the jointly optimal solution for all frames between any two consecutive I frames. This will ensure that both historical dependencies (i.e., how the choice of a predicted macroblock being encoded now gets effected by a predicting one encoded in the past) and future dependencies (i.e., how the choices made now affects the prediction performance in future) are taken into account. However such an exhaustive procedure is computationally infeasible and needs buffering of all frames over which the joint optimization is done. Moreover, the data-gathering step to populate **F1** with parameters $r_k(l)$, $d_k(l)$ etc. also becomes extremely impractical.

The above formulation is a simplification and may be used to study the effect of dependencies. It assumes a first order backward dependency (as with P frames) and the recursive effects beyond a single frame are ignored. We solve a sequence of joint 2-frame problems, and then use a heuristic to reconcile the results and determine the quantization choices. Denote by *F1(i, i+1)* the problem F1 solved for frames *i* and *i+1*. This yields a possible choice of quantization parameters for both frames *i* and *i+1*. Let *Quant(i)* be the process of quantizing frame *i* for some

quantization choices. Note that *Quant(i)* can be based on either *F1(i, i+1)* or *F1(i-1, i)*. Let *G(i, i+1)* be the ratio of (PSNR/Bits used) obtained by solving *F1(i, i+1)*.

An example best illustrates how to decide quantization choices over an image sequence. Let us assume we have 4 frames, numbered 1 through 4 that we need to quantize. Frame 1 is intra-coded while the others are assumed to be predictively coded. We follow the following steps:

1) Solve *F1(1, 2)* and *F1(2, 3)*. If *G(1,2)*=G(2,3), *Quant(1)* and *Quant(2)* using the quantization choice from *F1(1,2)*. Here the historical dependency of frame 2 dominates its future affects. If *G(1,2)<G(2,3), Quant(2)* using the choices given by *F1(2,3)*. Given these choices for frame 2, resolve *F1(1,2)* to obtain quantization choice for frame 1. *Quant(1)* using these choices. Note that in this case, the future affects for frame 2 dominate its historical dependency. After step (1), the quantization choice for frame 1 is frozen.

2) Solve *F1(2, 3)* and *F1(3,4)*. The following four cases arise:

- *G(2,3)=G(3,4), G(2,3)=G(1,2)*: *Quant(2)* and *Quant(3)* using the solution of *F1(2,3)*. The dependency effects between frames (2,3) dominate those between frames (1,2) and (3,4). Note that there is no ambiguity between the choice for frame 2 in steps (1) and (2).
- *G(3,4)=G(2,3), G(2,3)<G(1,2)*: *Quant(2)* using the solution of *F1(1,2)*. In step 1, frame 1 would have been quantized using the same problem. *Quant(3)* using *F1(2,3)*. We don't use *F1(3,4)* for quantizing frame 3 since *G(3,4)=G(2,3)*.
- *G(2,3)<G(3,4), G(2,3)=G(1,2)*: *Quant(2)* using the solution of *F1(1,2)*. *Quant(3)* using the solution of *F1(3,4)*.
- *G(2,3)<G(3,4), G(2,3)>G(1,2)*: *Quant(3)* using the solution of *F1(3,4)*. Note that in this case, frame 2 would earlier have been quantized using *F1(2,3)*. Since *G(3,4)>G(2,3)*, we backtrack and re-adjust quantization for frame 2. This is done using quantization choices for frame 3 obtained from *F1(3,4)* and resolving *F1(2,3)* for quantization choices for frame 2. We deliberately don't carry over any rippling effects back to frame 1.

After step (2), the quantization choice for frame 2 is frozen.

3) Proceed as in step (2) to quantize frames 3, 4, 5, … Note that here also we will restrict ourselves to four combinations of *G* to avoid a combinatorial growth in the number of relations. Thus the next step would use *G(2,3)*, *G(3,4)*, and *G(4,5)* only.

5.2.3 The rounding algorithms

This section discusses several methods to solve **F1,** each of which is based on a randomized rounding step applied to a relaxation of **F1**. A randomized rounding scheme (first introduced as a tool in [126][127]) is used to generate an integral solution using the LP relaxation. The rounding scheme may be viewed as a transformation that uses the LP solution to get inside the feasible IP polyhedron, while staying close to integral optimality. Section 5.2.3.1 applies a dependent rounding scheme on the LP relaxation of **F1**. We discuss why a direct LP based method may have problems, and in Section 5.2.3.2 try to further relax the problem using Lagrangian relaxations. Despite both these relaxations, even solving the LP may become too much of a computational burden since we have to do the operation repeatedly when encoding a sequence of frames. The primal-dual method based on recent results of Jain and Vazirani [82] and Goemens and Williamson [63] may be a feasible alternative in such cases.

5.2.3.1 LP-based dependent rounding

We first discuss a dependent rounding scheme based on the LP relaxation of **F1**. The rounding method we use has the following steps:

1) Solve the LP relaxation of **F1**. Let the optimal LP solution be $\tilde{X}_k(l)$ and $\tilde{Z}_{i,k}(j,l)$.

2) Let the rounded integer solution be $X_k^*(l)$ and $Z_{i,k}^*(j,l)$. Use $\tilde{X}_k(l)$ and $\tilde{Z}_{i,k}(j,l)$ to obtain $X_k^*(l)$ and $Z_{i,k}^*(j,l)$ as follows:

- Round $\tilde{X}_k(l)$ under constraint (C1) using,

$$P(X_k^*(l)=1)=\tilde{X}_k(l)$$

$$P(X_k^*(l)=0)=1-\tilde{X}_k(l)$$

- After the above, round $\tilde{Z}_{i,k}(j,l)$ under constraint (C2) using,

$$P(Z_{i,k}^*(j,l)=1 \mid X_k^*(l)=1)=\frac{\tilde{Z}_{i,k}(j,l)}{\tilde{X}_k(l)}$$

$$P(Z_{i,k}^*(j,l)=0 \mid X_k^*(l)=1)=1-\frac{\tilde{Z}_{i,k}(j,l)}{\tilde{X}_k(l)}$$

$$P(Z_{i,k}^*(j,l)=0 \mid X_k^*(l)=0)=1$$

$$P(Z_{i,k}^*(j,l)=1 \mid X_k^*(l)=0)=0$$

Note that with the above rounding functions, constraint (C3) is always satisfied. Further,

$$\begin{aligned} P(Z_{i,k}^*(j,l)=1) &= P(Z_{i,k}^*(j,l)=1 \mid X_k^*(l)=0)P(X_k^*(l)=0)+P(Z_{i,k}^*(j,l)=1 \mid X_k^*(l)=1)P(X_k^*(l)=1) \\ &= 0+\frac{\tilde{Z}_{i,k}(j,l)}{\tilde{X}_k(l)}\tilde{X}_k(l) \\ &= \tilde{Z}_{i,k}(j,l) \end{aligned}$$

and

$$P(Z_{i,k}^*(j,l)=0)=1-\tilde{Z}_{i,k}(j,l)$$

3) If after step (2) $\sum_l X_k^*(l)=0$ for any k, repeat step (2) until $\sum_l X_k^*(l)\geq 1$ for all k. We will call each of these an X-iteration. If $\sum_l X_k^*(l)>1$ for any k, let $L=\{l: X_k^*(l)=1\}$. Choose $l^*=\arg\max\{[1/d_k(l)r_k(l)] \mid l\in L\}$. Deterministically set $X_k^*(l^*)=1$ and $X_k^*(l\neq l^*)=0$. Intuitively, this sets those X's that contribute most profit density in the objective function to 1. A similar strategy is used to ensure that constraint *(C2)* is satisfied. If after step (2) $\sum_{k,l,j} Z_{i,k}^*(j,l)=0$ for any i, repeat step (2) until $\sum_{k,l,j} Z_{i,k}^*(j,l)\geq 1$ for all i. In our experiments, we found that in certain cases this requires far too many Z-iterations. Thus, we threshold the

maximum number of times this step may be carried out. If after the maximum number of Z-iterations there still exists an i for which $\sum_{k,l,j} Z^*_{i,k}(j,l) = 0$, we find the Z's that may be 1 (under given X's and under constraint (C3)). Among these potential choices of Z=1, we deterministically set the one that has the maximum profit density to 1 and the rest to 0. Similarly, if we find that for some i, $\sum_{k,l,j} Z^*_{i,k}(j,l) > 1$, a pruning step like the one used for X's is exercised. Among candidate Z's that may be 1, this step sets the Z with maximum profit density to 1, and the rest to 0 (again under constraint (C3)).

4) Assuming the rounded integer solution from steps (1)-(3), check if constraint (C4) is satisfied. If not, repeat steps (1)-(3) to obtain a new integral solution. If (C4) is satisfied, output the solution and objective value.

Systematic rounding schemes have been fairly successful to generate heuristic solutions for several integer problems (see, for example, [24][126][127] and references therein). The other typical component of a rounding algorithm is a proof on worst case bounds on its performance given feasibility. Most effort in this direction relies on Chernoff's inequality [36]. Using probabilistic bounds on the size of the tail of a sum of iid random variables, Bertsimas and Vohra [24] give an algorithm similar to our rounding scheme for the uncapacitated facility location problem [40][41]. The results of [24] are more exhaustive than ours in that the authors also prove an *O(log(n))* worst case guarantee on the performance of their algorithm. The problem we consider here, however, is much more complicated due to the added packing constraint (C4). Even constraints (C1) and (C2) are more restrictive than those of [24]. Despite several attempts, we have not been able to prove guarantees for dependent rounding methods of the form in Section 5.2.3.1. The main difficulty for formulation F1 stems from constraint C4. An attempt to prove a guarantee on its feasibility (for the rounding scheme considered) leads to a sum of *dependent* Bernoulli random variables for which tail inequalities of the form in [36] are not known. Nonetheless, as will be shown through experimental results, our algorithms perform quite well in practice.

5.2.3.2 LR-based dependent rounding

The approach of Section 5.2.3.1 requires an extra effort to ensure that the constraints (C4 in particular) are satisfied. In this section, we further relax problem F1 using Lagrange multipliers. We consider two natural LRs of **F1**. The first absorbs constraint C4 into the objective function, while the second relaxes constraints C1 and C2.

5.2.3.2.1 Lagrange relaxation 1 (LR1)

Let $Z_1(l) = \sum_{k,l} [d_k(l) + l\, r_k(l)] X_k(l) + \sum_{k,l,i,j} [d_{i,k}(j,l) + l\, r_{i,k}(j,l)] Z_{i,k}(j,l)$

Then **LR1** is given by,

$\max_{l \geq 0} \{L_1(l)\}$ where

$L_1(l) = \min \ (Z_1(l) - l\, R)$

$$\text{st.} \sum_{l} X_k(l) = 1 \text{ for } k = 1,\ldots,n$$

$$\sum_{k,l,j} Z_{i,k}(j,l) = 1 \text{ for } i = 1,\ldots,n$$

$$Z_{i,k}(j,l) \leq X_k(l) \text{ for all } i,k,j,l$$

$$Z_{i,k}(j,l), X_k(l) \in \{0,1\} \text{ for all } i,k,j,l$$

Note that if (x^*, z^*) is optimal for F1, then $L_1(l) \leq Z_1(l) - l\, R\,|_{(x^*,z^*)} \leq F1(OPT)$. Further, because of the first two constraints, we have

$$\frac{l\, R}{2n} 2n = \frac{l\, R}{2n} [\sum_{l,k} X_k(l) + \sum_{i,j,k,l} Z_{i,k}(j,l)]$$

$$\Rightarrow \ Z_1(l) - l\, R = \sum_{k,l} [d_k(l) + l\, r_k(l) - \frac{l\, R}{2n}] X_k(l) + \sum_{i,j,k,l} [d_{i,k}(j,l) + l\, r_{i,k}(j,l) - \frac{l\, R}{2n}] Z_{i,k}(j,l)$$

Note that the structure of the problem remains the same as before. Thus we may use the rounding scheme of Section 5.2.3.1 to solve approximately for $L_1(l)$ except that step (4) in the algorithm is no longer needed. Instead we search for the best Lagrange parameter, $l \geq 0$, to satisfy the resource constraint most tightly.

The sub-gradient method of Held, Wolfe, and Crowder [53] is used to search for the Lagrange parameter. The method starts with an initial value $l_0 = 0$ of l and builds a sequence $\{l_m\}$ of estimates using,

$$l_{m+1} = l_m + t_m(\sum_{k,l} r_k(l)X_k^m(l) + \sum_{i,j,k,l} r_{i,k}(j,l)Z_{i,k}^m(j,l) - R)$$

Here $\{X_k^m(l), Z_{i,k}^m(j,l)\}$ solves **LR1** at the m^{th} iteration. The step size t_m is chosen as:

$$t_m = \frac{b_m(L^* - L_1^m(l_m))}{[\sum_{k,l} r_k(l)X_k^m(l) + \sum_{i,j,k,l} r_{i,k}(j,l)Z_{i,k}^m(j,l) - R]^2}$$

Here L^* is an upper bound on $L_1(l)$ obtained by applying a greedy heuristic to **F1**. The greedy heuristic solves for $X_k(l)$ assuming a total bit-budget of *R/2* to maximize marginal returns [56]. Assuming the above choice of $X_k(l)$, the method of Fox is used again to find $Z_{i,k}(j,l)$ under constraints C2, C3 and modified C4. $b_0 = 2$ and is halved every 5 iterations if $L_1(l)$ does not change.

5.2.3.2.2 Lagrange relaxation 2 (LR2)

The second LR we consider relaxes constraints C1 and C2. Let,

$$Z_2(\vec{u}, \vec{l}) = \sum_k \{\sum_l [d_k(l) + u_k]X_k(l)\} + \sum_i \{\sum_{k,l,j} [d_{i,k}(j,l) + l_i]Z_{i,k}(j,l)\} - \sum_k u_k - \sum_i l_i$$

Then **LR2** is given by,

$$\max_{u_k, l_i \text{ unconstrained}} \{L_2(\vec{u}, \vec{l})\} \text{ where}$$

$$L_2(\vec{u}, \vec{l}) = \min\{Z_2(\vec{u}, \vec{l})\}$$

$$\text{st. } Z_{i,k}(j,l) \le X_k(l) \text{ for all } i,k,j,l$$

$$\sum_{k,l} r_k(l)X_k(l) + \sum_{i,j,k,l} r_{i,k}(j,l)Z_{i,k}(j,l) \le R$$

$$X_k(l),\ Z_{i,k}(j,l) \in \{0,1\} \text{ for all } i,k,j,l$$

There are several reasons that discouraged us from pursuing **LR2** further:

- As there are *2n* Lagrange parameters in **LR2**, it is much more complicated to search for the best one. Contrast this with **LR1** where the search was in a single dimension.

- In our experiments with the algorithm of Section 5.2.3.1, we found that it was more difficult to satisfy constraint C4 as compared to C1 and C2. Since constraint satisfaction was the primary reason that we explored Lagrange relaxations to start with, we found it more beneficial to stick to **LR1**. One could go one step further and Lagrange relax constraint C4 from **LR2**. Although the resulting Lagrange problem would be trivially solvable, the entire computational burden would be relegated to the search of the best multiplier set.

- Unlike $L_1(l)$, there isn't a straightforward method to solve for $L_2(\vec{u}, \vec{l})$ optimally or approximately. A similar LR results for the facility location problem considered in [40]. The techniques of [40], however, do not apply here since for a given $(\vec{u}, \vec{l})$ the following holds only if constraint C4 is maintained:

$$Z_{i,k}(j,l) = 0 \quad \text{if } d_{i,k}(j,l) + l_i \geq 0$$
$$= X_k(l) \text{ if } d_{i,k}(j,l) + l_i < 0$$

Hence, the above Lagrange problem can only be *"approximately"* parameterized in terms of $X_k(l)$.

5.3 Experimental study

The experimental study was carried out on a sequence of 100 frames from the Table Tennis sequence. Each frame was 352x240 pixels in size and was divided into 16x16 macroblocks, leading to a total of 330 macroblocks per frame. Each macroblock was DCT compressed at five quality levels. For simplicity we assume a motion vector of size 0 in all our simulations. Given the quantization choice for the predicting macroblock, we calculated the residual signal and DCT compressed it at five quality levels. This process was repeated for all the macroblocks in all pairs of adjacent frames in an offline data generation stage. This populated 100 linear programs of the form F1. IBM's Optimization Solution (OSL) was used for solving these linear programs. The linear program for each pair of frames had 9900 variables and 8911 constraints. Our objective in these simulations was two-fold. First, we wanted to test the quality of solution obtained by the rounding methods discussed above. Second, we wanted to test if incorporating dependency in making quantization decisions leads to an improvement over independent

decisions on the predicting frame and the resulting residual in the next frame. The experimental results are detailed in the following subsections.

(1) Typical example of dependent RD behavior

In Figure 5.2, we plot the decoded PSNR versus quality factor for a typical predicting macroblock. The quality factor is normalized to be one of five levels (1,...,5). As has been noticed by earlier authors also, we note that the R-D behavior is not concave or even monotonically increasing.

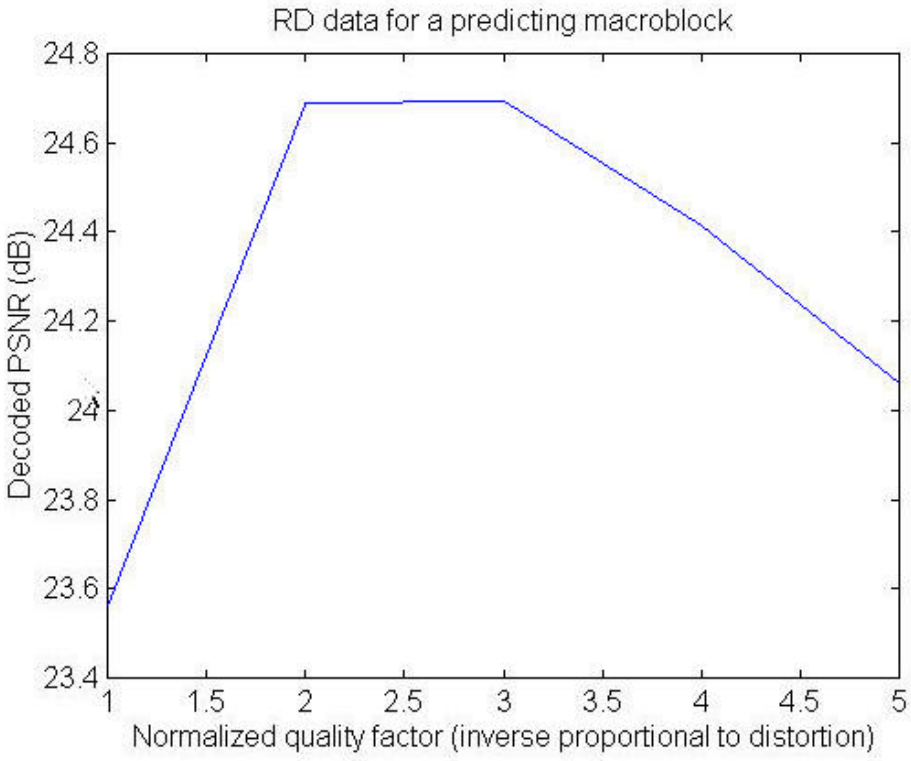
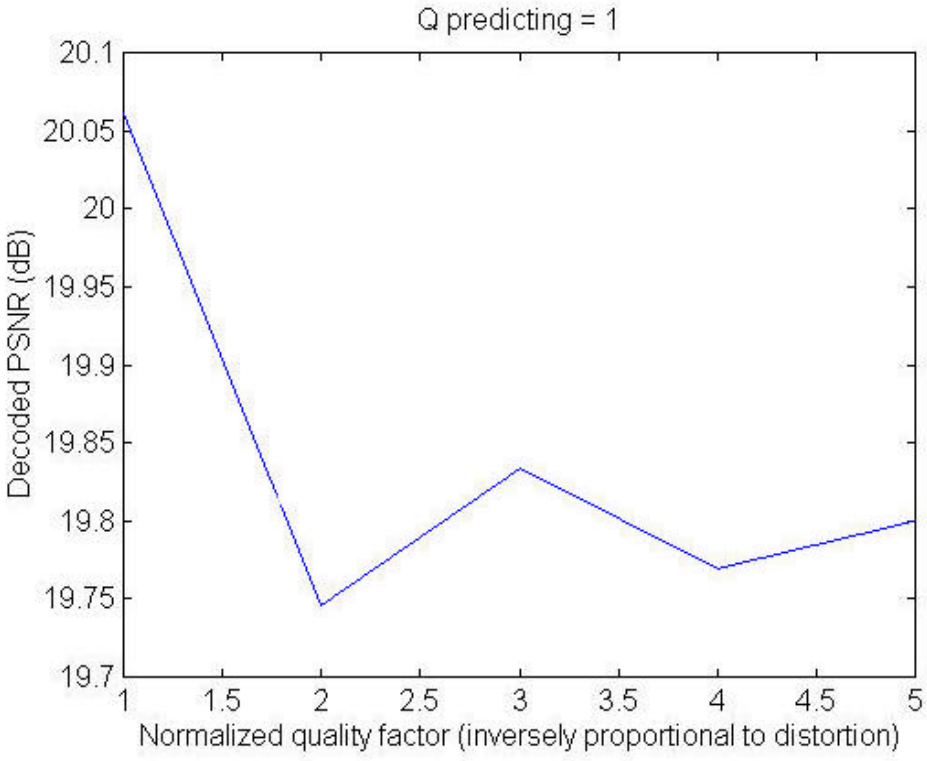
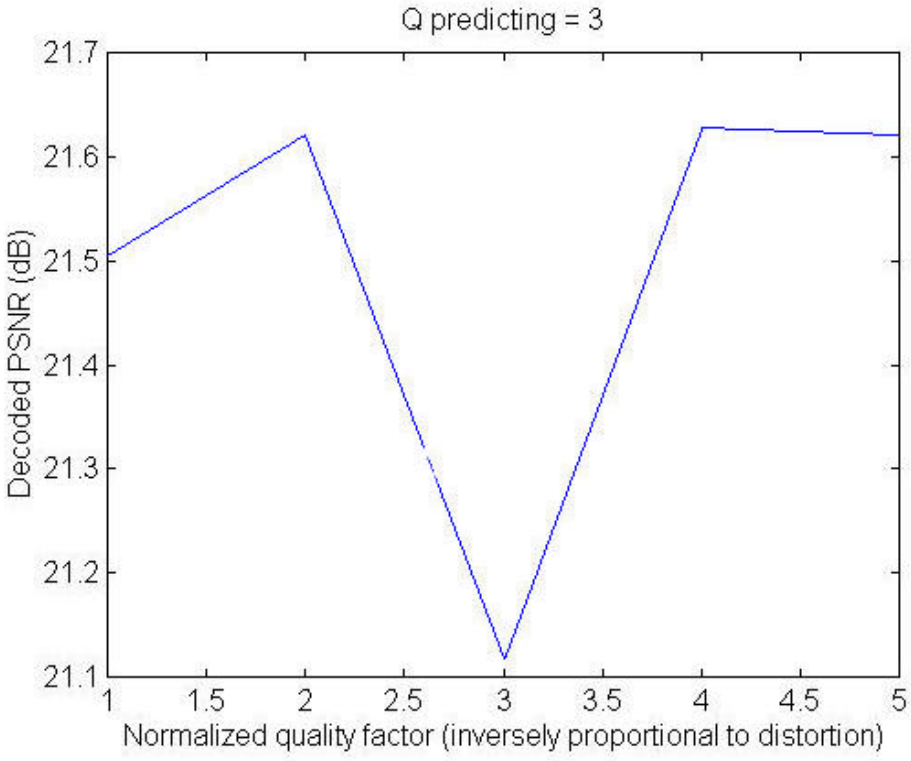


Figure 5.2: R-D behavior of predicting macroblock.

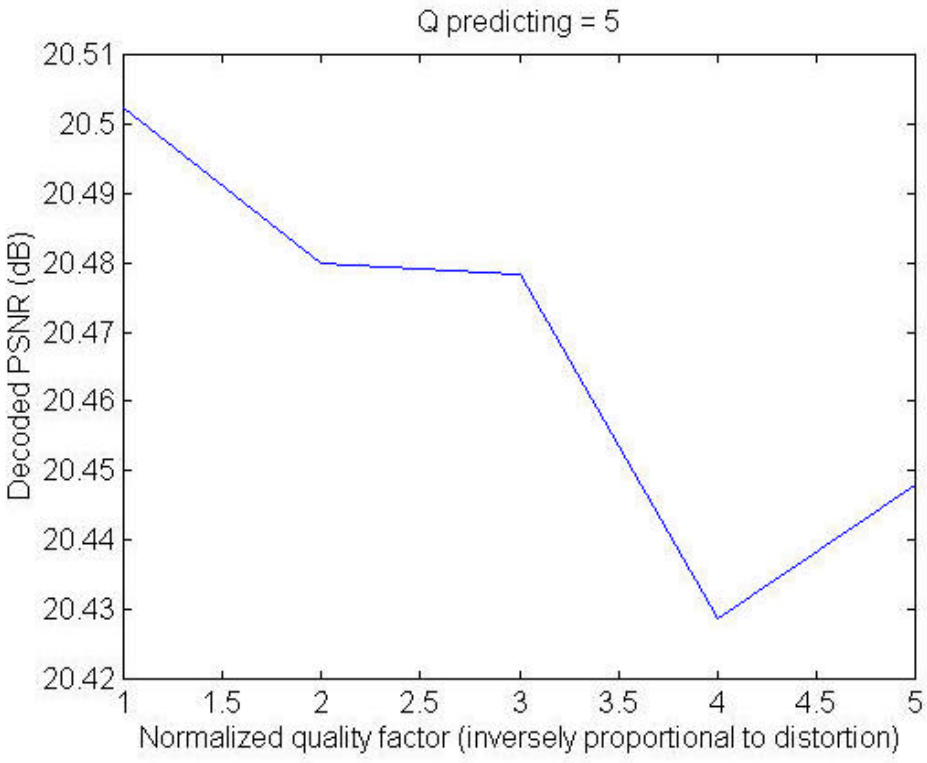
In Figures 5.3(a) to 5.3(c), we plot the decoded PSNR versus quality factor for the predicted macroblock conditioned on a given quantization choice for the predicting macroblock. Again, we note that the behavior is highly non-concave (and is not even monotonic). This contradicts assumptions made by earlier authors [34][115] to solve the dependent quantization problem.


Q predicting = 1
Decoded PSNR (dB)
20.1
20.05
20
19.95
19.9
19.85
19.8
19.75
19.7
1
1.5
2
2.5
3
3.5
4
4.5
5
Normalized quality factor (inversely proportional to distortion)

(a)


Q predicting = 3
Decoded PSNR (dB)
21.7
21.6
21.5
21.4
21.3
21.2
21.1
1
1.5
2
2.5
3
3.5
4
4.5
5
Normalized quality factor (inversely proportional to distortion)

(b)



(c)

Figure 5.3: R-D behavior of predicted macroblock (dependent RD behavior).

(2) Performance of the rounding algorithm of Section 5.2.3.1:

We now discuss the performance of LP based rounding algorithm of Section 5.2.3.1. We know that for a minimization problem, the optimal LP solution forms a lower bound on the optimal IP solution. Ideally, we would like to compare the rounded integral solution (obtained by rounding the LP solution) with the optimal IP solution. Since the IP solution is difficult to obtain, we bound the gap between optimal IP solution and rounded solution using the gap between the LP optimal solution and the rounded solution. We plot this gap as a percentage in Figure 5.4. Results indicate that the rounding method performs 'extremely well' and that for all practical purposes the gap is almost zero.

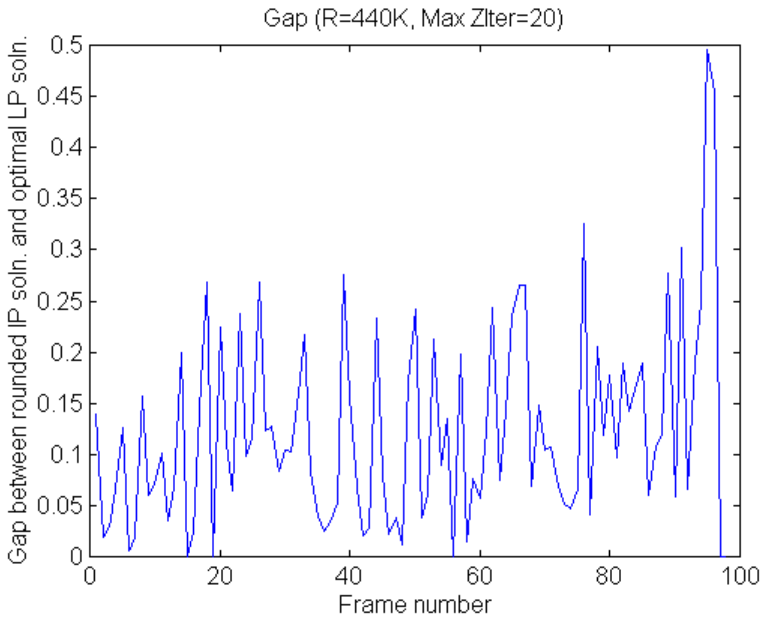
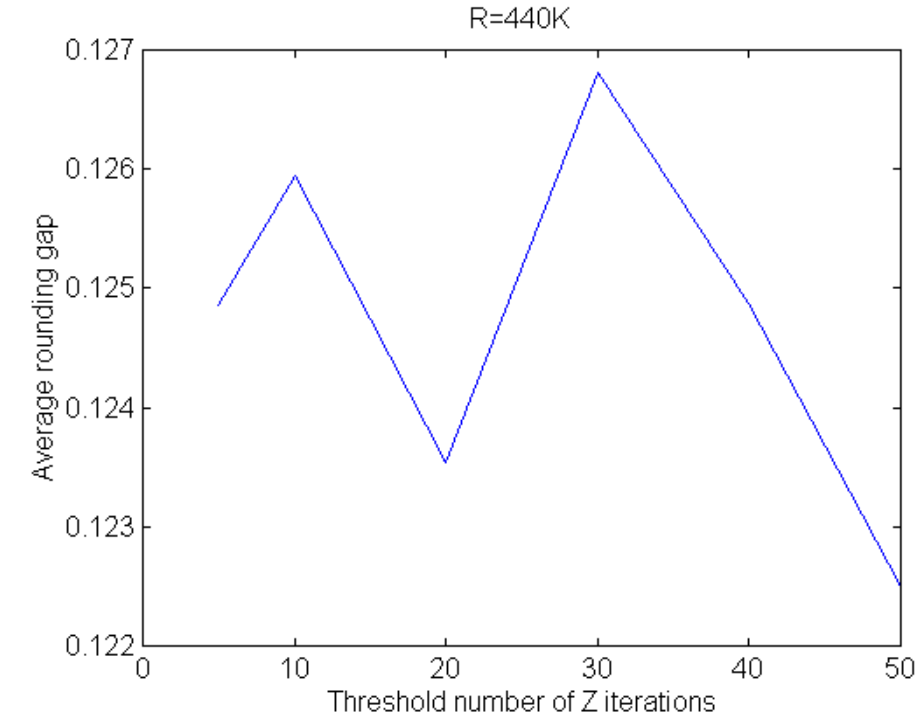


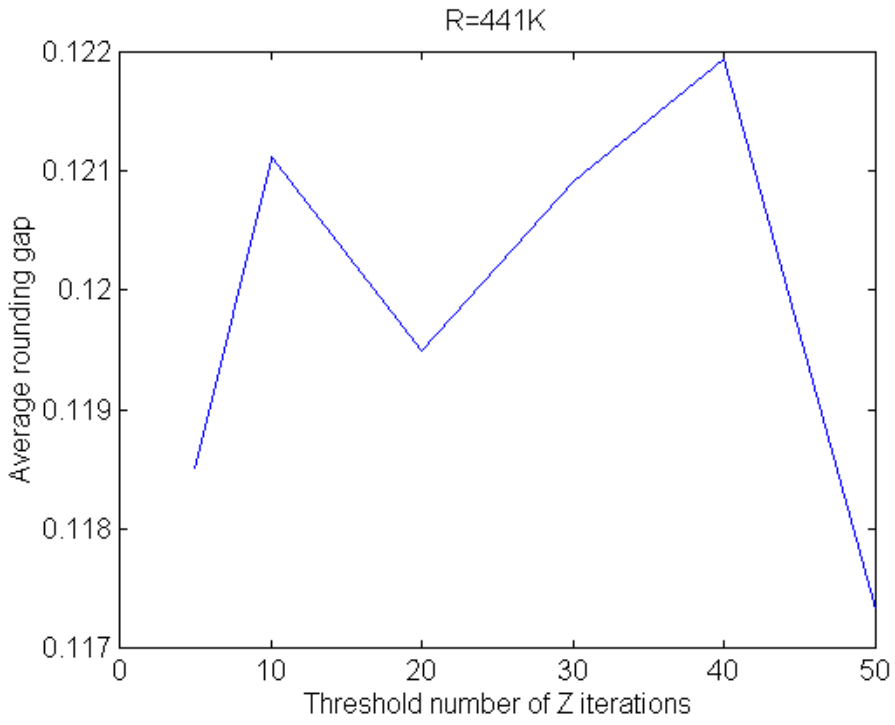
Figure 5.4: Percentage gap between rounded IP solution and optimal LP solution (R=440K bits)

During these simulations we also noted the following:

- The threshold number of X-iterations required for convergence in Step (3) of Section 5.2.3.1 ranged between 1-20.
- We experimented with the sensitivity of the rounding gap to the threshold number of *Z* iterations. As discussed in Step (3) of Section 5.2.3.1, this was used as an escape route if constraint **(C2)** required far too many iterations to converge. In Figure 5.5, we plot the average rounding gap as a function of threshold number of *Z* iterations. As can be seen in these plots, the rounding gap seems to be fairly insensitive to the threshold number of *Z* iterations. In practice, a threshold value of 20 or 30 worked quite well.



(a)

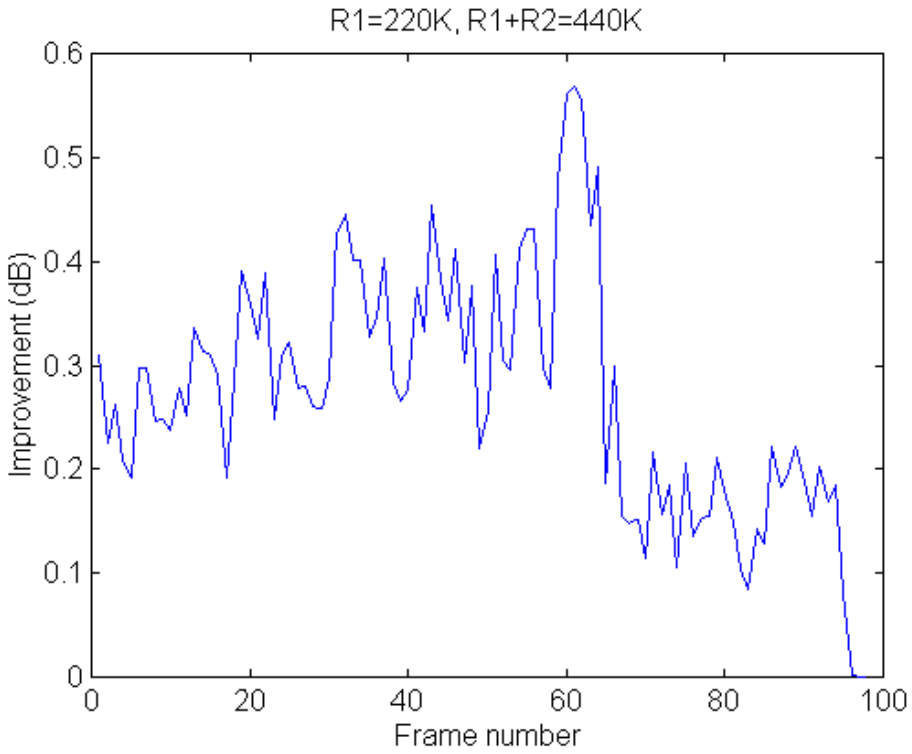


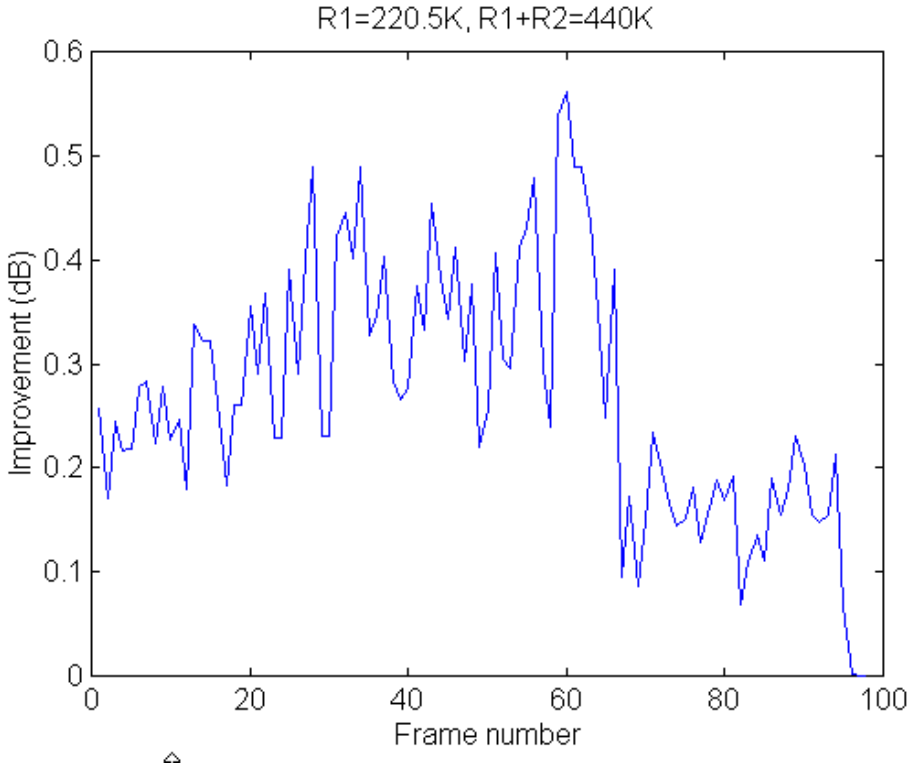
(b)

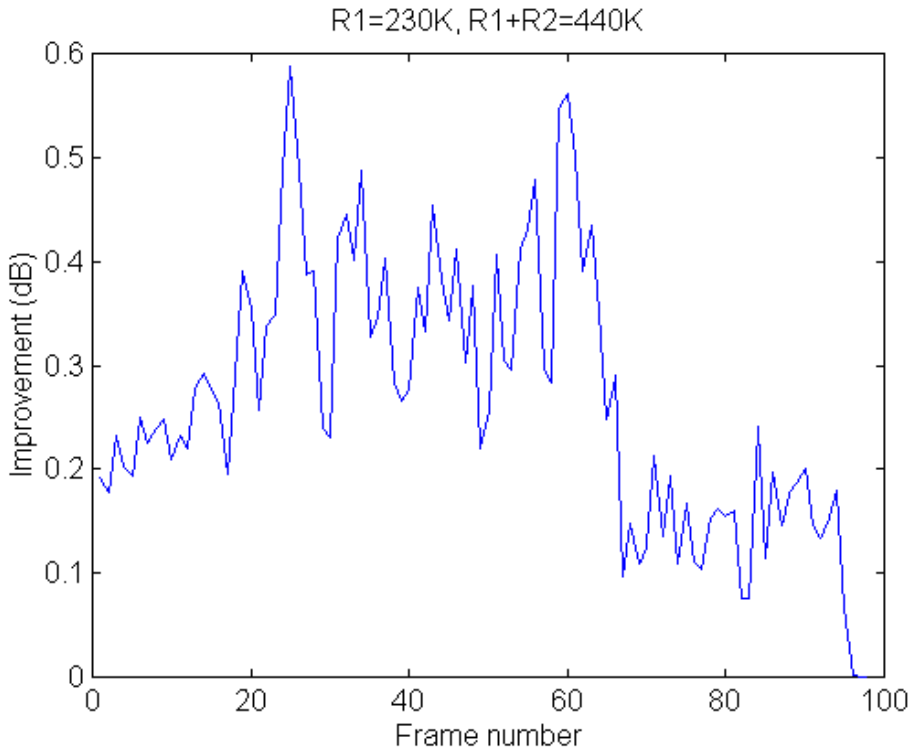
Figure 5.5: Sensitivity of LP rounding algorithm to the threshold number of Z iterations (R=440K, and 441K bits).

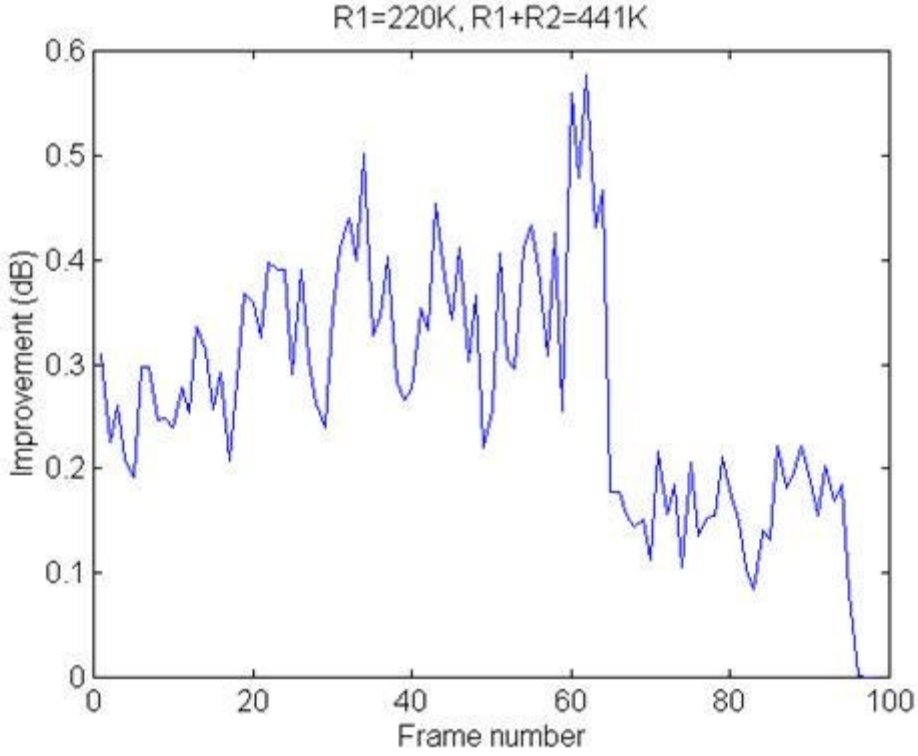
(3) Comparison of causally optimal solution with the dependent quantization approach:

Next we compare the performance of causally optimal algorithm to the dependent quantization approach. For the causally optimal scheme, we independently solve the MCKP problem for the predicting frame. Given the quantization choices for the predicting frame, the joint (two frame) LP is solved, and a rounded solution is obtained based on it. The dependent quantization problem solves the joint two frame problem (of the form F1) with no prior assumptions. In the figures below, we plot the extra improvement obtained with the dependent quantization approach. We experimented with different choices of the total bit budget, and its partitioning between the two frames. From the results we note that on the average an improvement of about 0.2 to 0.5 dB is obtained. Though this number may not be very high, it gives us a reason to expect that as the degree of dependency incorporated in the formulation increases, reasonably high improvements may be obtained. This could justify doing some sort of 'globally optimal' dependent bit allocation between successive I frames (as the memory and computational power required for doing so becomes available in future).


R1=220K, R1+R2=440K
Improvement (dB)
Frame number
0
0.1
0.2
0.3
0.4
0.5
0.6
0
20
40
60
80
100


R1=220.5K, R1+R2=440K
Improvement (dB)
Frame number
0
0.1
0.2
0.3
0.4
0.5
0.6
0
20
40
60
80
100


R1=230K, R1+R2=440K
Improvement (dB)
Frame number
0
0.1
0.2
0.3
0.4
0.5
0.6
0
20
40
60
80
100


R1=220K, R1+R2=441K
Improvement (dB)
Frame number
0
0.1
0.2
0.3
0.4
0.5
0.6
0
20
40
60
80
100

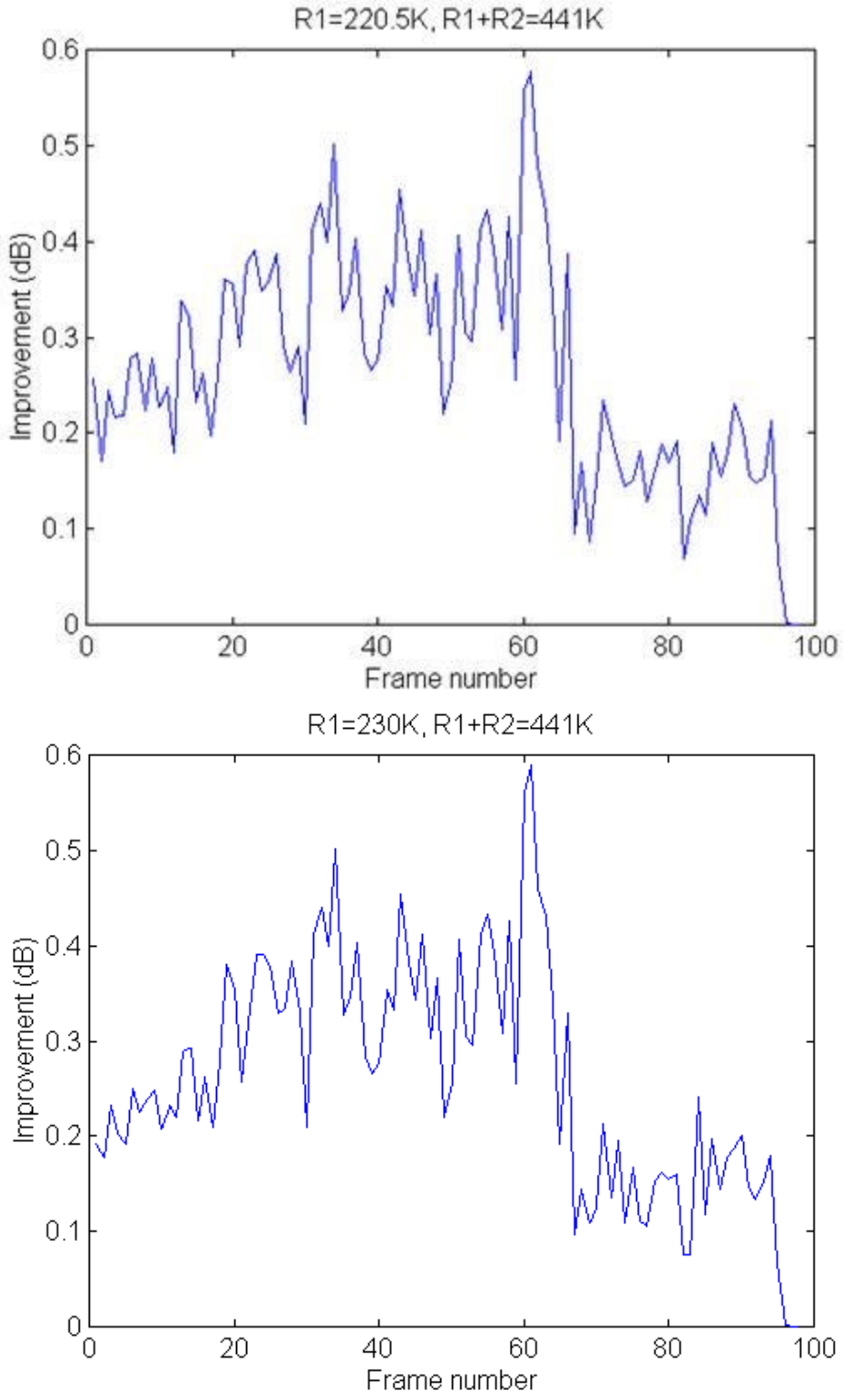
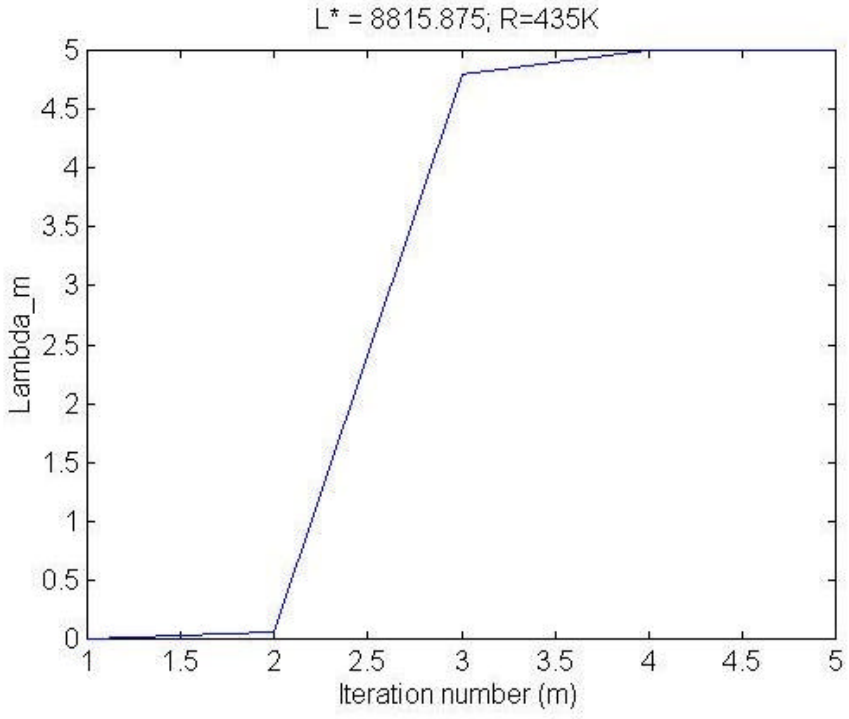


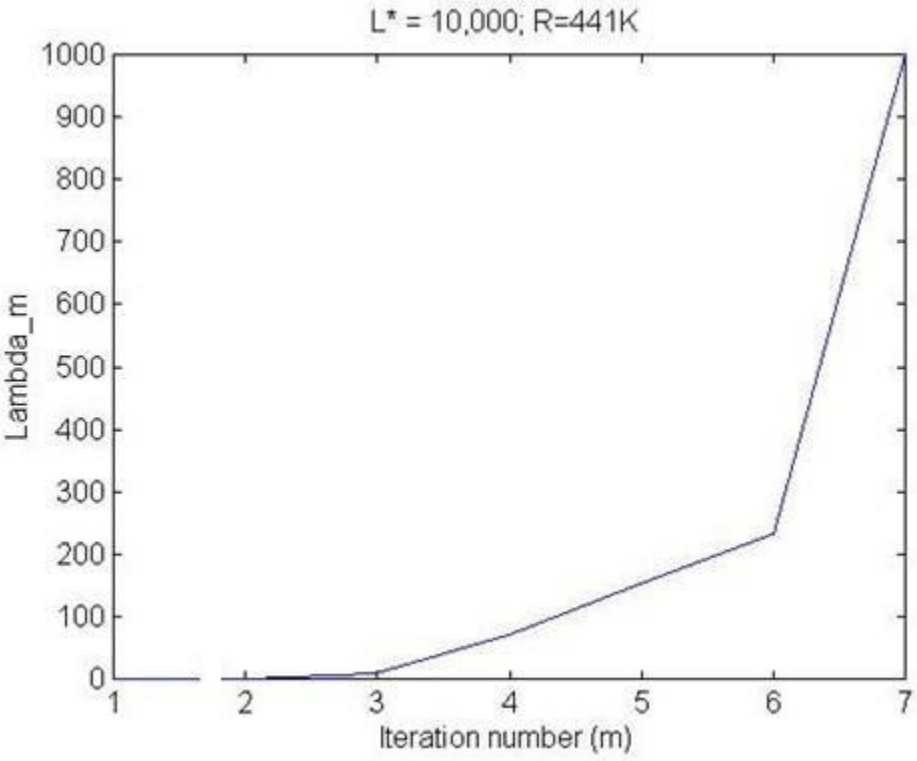
Figure 5.6: Improvement obtained using dependent quantization approach over the causally optimal solution.

(4) Performance of the Lagrange relaxation approach of Section 5.2.3.2:

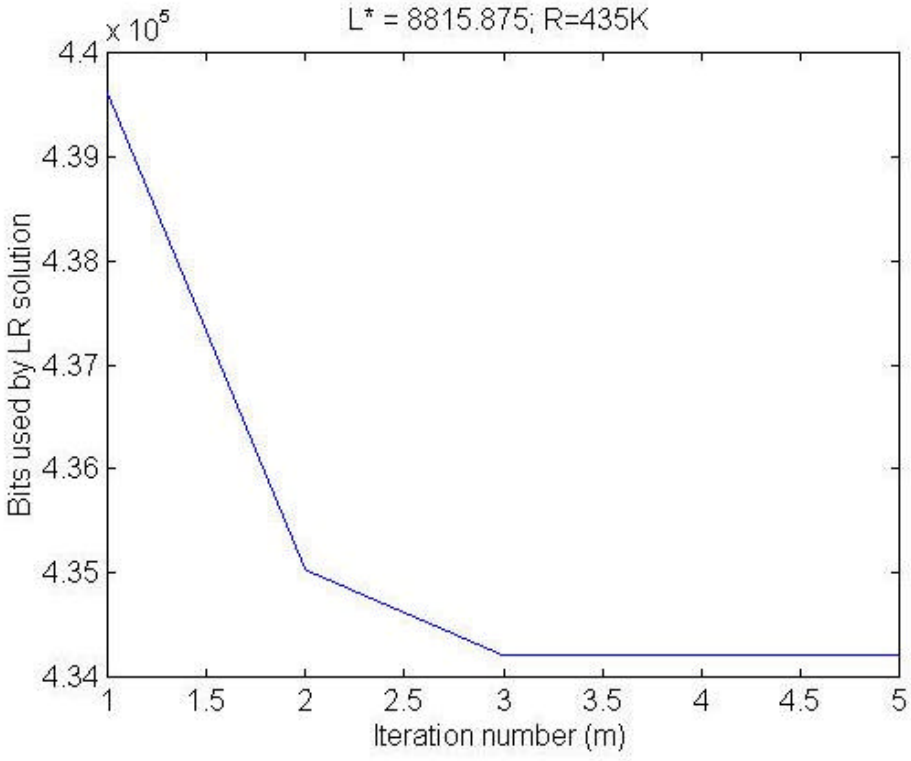
In the simulations conducted for the LP rounding algorithm, we noticed that sometimes the number of iterations needed to satisfy constraint *(C4)* became large. This was particularly true as the bit budget *(R)* was reduced. In such cases, the LR algorithm of Section 5.2.3.2 was used as a backup mechanism. This algorithm solves the Lagrange problem for a sequence of Lagrange parameters. Below we present convergence behavior of the algorithm for a max sum (of PSNRs) problem of the form F1. Figure 5.7(a) plots variation in the Lagrange parameter as function of the iteration number. Figures 5.7 (c) and (e) show the number of bits used and the value of the LR solution obtained as a function of the iteration number. Since the Lagrange problem relaxes the resource constraint, the value of its solution is super-optimal for problem F1 (i.e., it is larger than the optimal IP solution for a max sum form for F1). As the Lagrange parameter is increased the relaxed constraint gets weighted into the objective function. Consequently the gap between the number of bits used by the LR solution and the total bit budget *R* shrinks, and the value of LR solution deteriorates. Because of its very fast convergence, we believe that this LR algorithm gives a very good alternative to the LP rounding approach when it fails to satisfy the problematic resource constraint.



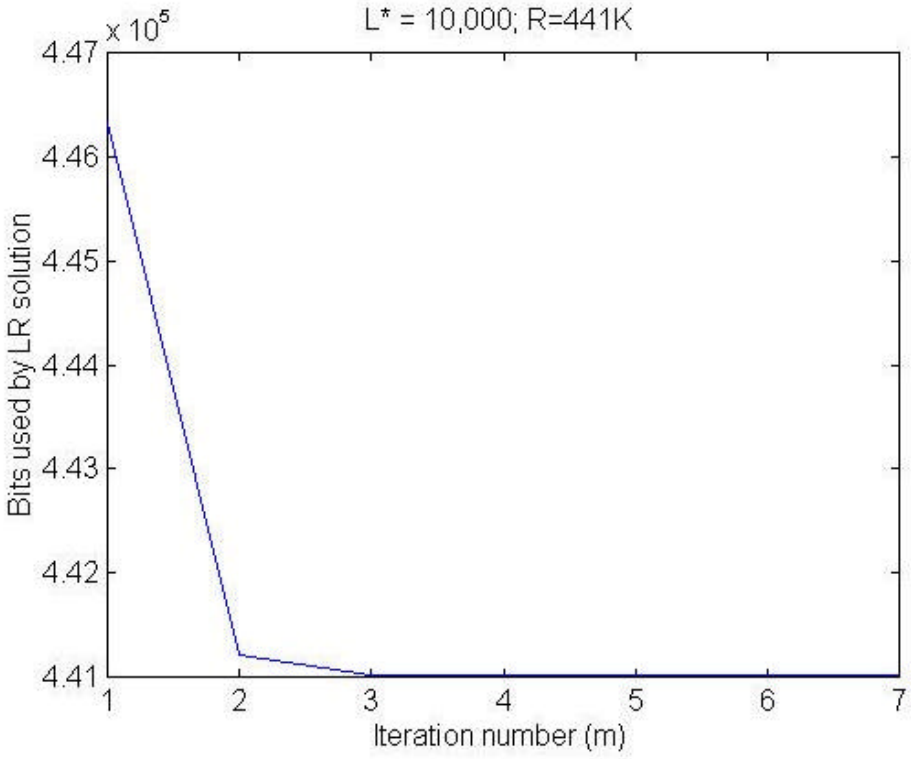
(a)


L* = 10,000; R=441K
Lambda_m
Iteration number (m)
0
100
200
300
400
500
600
700
800
900
1000
1
2
3
4
5
6
7

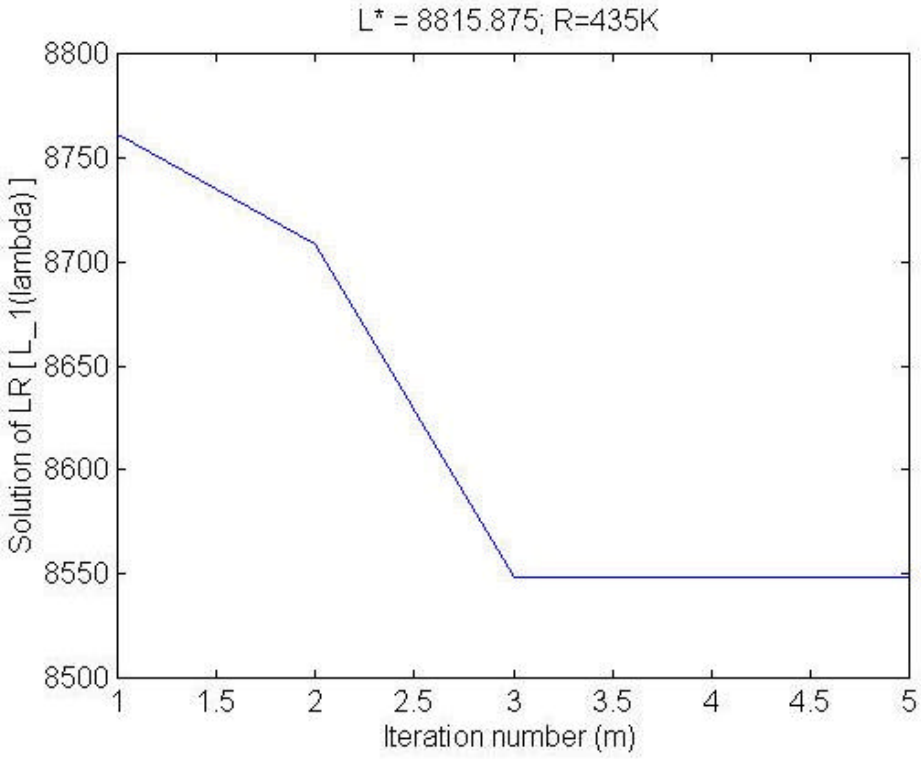
(b)


L* = 8815.875; R=435K
x 10^5
Bits used by LR solution
Iteration number (m)
4.34
4.35
4.36
4.37
4.38
4.39
4.4
1
1.5
2
2.5
3
3.5
4
4.5
5

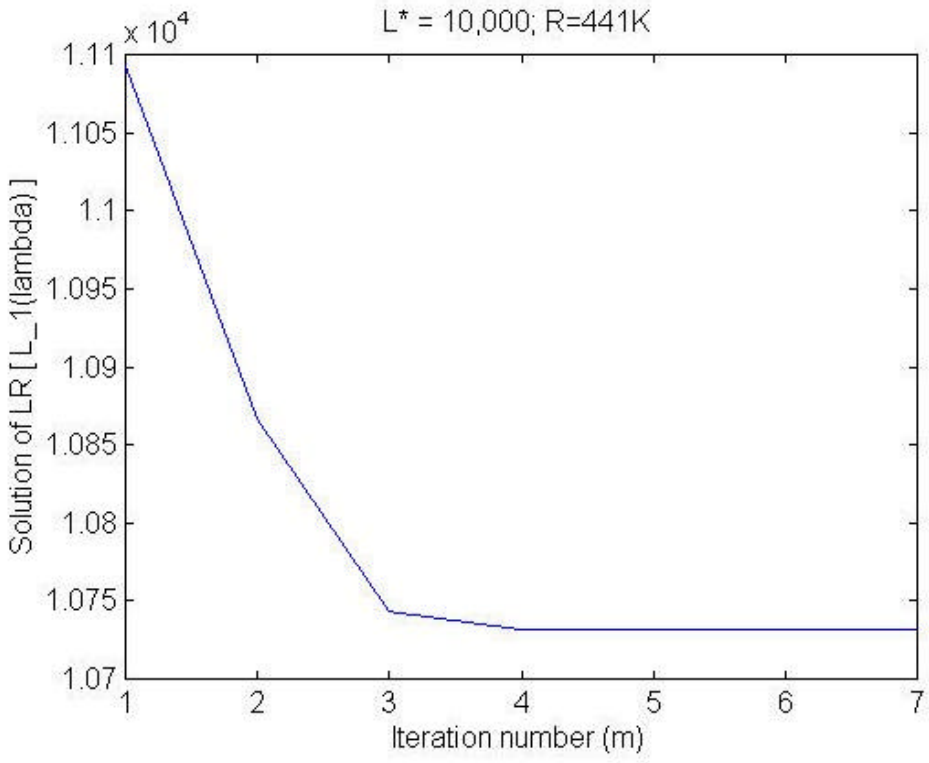
(c)


L* = 10,000; R=441K
x 10^5
4.47
4.46
4.45
4.44
4.43
4.42
4.41
Bits used by LR solution
1
2
3
4
5
6
7
Iteration number (m)

(d)


L* = 8815.875; R=435K
8800
8750
8700
8650
8600
8550
8500
Solution of LR [L_1(lambda)]
1
1.5
2
2.5
3
3.5
4
4.5
5
Iteration number (m)

(e)



(f)

Figure 7: Convergence of Lagrange relaxation algorithm. (R=435K and 441K).

(a,b) *l* as a function of iteration number (m).

(c,d) Sensitivity of the number of bits used by the LR solution as a function of iteration number.

(e,f) Convergence of LR solution as a function of iteration number.

5.4 Summary of the chapter

This chapter studied the bit allocation problem with a restricted dependency structure. In particular, we presented MCKP-like formulations for optimal allocation of bits among predicting and predicted macroblocks. We discussed rounding procedures that may be used to generate approximate solutions after solving some relaxation of the original problem (either the LP relaxation or a Lagrange relaxation). Experimental results show an improvement of about 0.2 to 0.5 dB over the causally optimal solution. From a practical standpoint this validates that the causality assumption is a very justified one to reduce problem complexity for a marginal loss of optimality.

CHAPTER 6

Optimal Dynamic Rate Shaping of Markov-1 Sources

Contents

6.1 Introduction

The concept of Dynamic Rate Shaping (DRS) [45][49][50] was introduced in early 1995 as a technique to adapt the rate of compressed video bit-streams (MPEG-1, MPEG-2, H.261, as well as JPEG) to dynamically varying bit rate constraints. The need for rate adaptation is a very naturally-arising one in best-effort networks such as the Internet. Despite increases in bandwidth availability, the fact that the underlying medium is shared necessitates adaptation at the edges of the network. Furthermore, the recent success of wireless communications, not only in the form of cellular telephony but also wireless LANs, demonstrates that the need for bandwidth adaptation is ever present. Finally, the variety of devices with which users may want or need to access content is so large, that being able to adapt on the fly is extremely important for offering a viable content distribution service. In the past, server operators were concerned about scalability issues – any additional processing done at a server would necessitate a reduction in the server's capacity. Current server speeds, as well as the transition away from free content models, have rendered this argument more or less moot.

DRS bridges the gap between constant and variable bit rate video, allowing a continuum of possibilities between the two. The approach has several applications. It provides an interface (or bitstream filter) between an encoder and a network, with which the encoder's output can be perfectly matched to the network's quality of service characteristics. It can match the rate capabilities of encoders with a variety of decoders. It can facilitate multipoint communication with mobile hosts, without compromising the signal quality of wired participants and can provide smoother trick-mode (fast forward, fast reverse) operation in digital video recorders, and so on. Since the presented algorithms do not require interaction with the encoder, they are fully applicable to pre-coded, stored video (as in, for example, video-on-demand systems). By providing decoupling of the encoder and channel or decoder in terms of rate, universal interoperability can be achieved.

6.1.1 Related Work

At the time when DRS was introduced, techniques had been developed to employ rate control for live sources based on network feedback [47][88]. No general solution was available, however, for prerecorded material. Earlier work such as the approaches used in [106] and [162] to manipulate the rate of compressed H.261 streams were ad-hoc, with no characterization of the performance of the proposed schemes. Our focus is based on techniques where select DCT coefficients are dropped (similar to data partitioning [50]). The other major category of algorithms is that where re-quantization of DCT coefficients is performed. Since the original introduction of DRS, considerable work has been performed on the subject. In [154] a set of basic techniques for "bitstream scaling" are discussed, including the elimination of high frequency components, but they are based on heuristics with no theoretical analysis or claim of optimality. An early re-quantization-based scheme where the quantization step size is determined by local and global activity criteria is presented in [109]. A more detailed theoretical analysis for proper (re)quantizer selection for intra frames is offered in [161]. The subject is further addressed in [5], where the optimal selection of quantizers is analyzed. Along these lines, straightforward decoding plus encoding cascades are presented and optimized in [90][155][163]. Other techniques for rate adaptation include [137], an alternative (but incompatible with existing standards) technique for data partitioning using re-quantization into an even value approximation and an odd remainder part, [4] where the entire operation is very efficiently performed in the frequency (DCT) domain, and [166] where rate shaping is performed by block dropping and intelligent reconstruction at the (non-standard) receiver. Techniques have also been investigated for the proper integration of rate shaping schemes with actual networks and decoders; in [80] we describe a real-time DRS implementation for Internet video including a TCP-friendly rate controller, whereas in [70] buffer conditions for the encoder and transcoder are presented in order to avoid overflow/underflow at the decoder. Most recently, DRS has been applied on MPEG-4, combining fine granular scalability (FGS) and FEC with R-D optimization [33]. Although our interest is primarily "homogeneous" rate shaping, where the source and target coding formats are identical, it is also possible and – indeed – very interesting to examine heterogeneous rate shaping or transcoding (e.g., from MPEG-2 to MPEG-4). Examples include [142][143][160]. Finally, it is also natural to ask whether or not the original coding scheme could

be architected so that it facilitates rate adaptation with minimal additional processing. Scalable coding in this respect is extremely useful, and developments in this area are very promising (e.g., [135] and [157]). We should point out, however, that all currently deployed coding standards are all single layer schemes.

6.1.2 Results

Our earlier work in [45][48][49][50] discussed a family of DRS algorithms for two cases of DRS, namely constrained (CDRS) and general or unconstrained (GDRS). This chapter substantiates the limited observations in these papers by a more extensive set of experimental simulations on actual MPEG-2 bitstreams. We also report on the complexity of these algorithms on today's computers and justify the computational viability of the DRS approach to transcoding. Our findings suggest the following:

- A so-called memoryless algorithm that dispenses the shaping error accumulation from past frames performs very close to the causally-optimal solution and does so over a large spectrum of cluster sizes and rates. This is so both for the constrained as well as the unconstrained versions of the DRS problem.
- Recoding or requantization based approaches have an inferior performance compared to both the causally optimal and the memoryless algorithms for rate shaping ratios less than 2.
- Unconstrained DRS outperforms constrained DRS, but does to only marginally, due perhaps to the effectiveness of the zig-zag scanning order for DCT coefficients.
- Clustering does not decrease the complexity of any of the algorithms discussed by much. One may, therefore, decrease the optimization granularity and choose a distinct (potentially different) breakpoint for each block. Our results show that this leads to a substantially better PSNR performance.
- There is a major difference between the distribution of times spent in various stages of the GDRS and CDRS algorithms. While the optimization step takes a substantial chunk of time for generalized DRS, this step is seen to be extremely efficient for all variations of the constrained DRS problem. A major portion of the time for both CDRS and GDRS gets spent during bitstream decoding and R-D data collection.

The single most definitive experimental observation we shall make in both constrained and unconstrained approaches is that the memoryless algorithm performed extremely close to the causally optimal one. The desire to analytically investigate this property led us to explore the DRS problem with a known source distribution – in our case an AR(1) process. In this chapter we analyze the mixed mode DRS problem by providing a statistical characterization of the source model, the motion compensated frame difference (MCFD) signal it generates, the current frame shaping error and the accumulated motion compensation shaping error from past frames. We derive the rate distortion behavior of the MCFD signal for both large and finite block sizes. Utilizing these results along with facts from Matrix Perturbation Theory [159], we show that the spectrum (eigenvalues) of the current frame residual error alone and that along with the motion compensated accumulated error are separated by bounded constants. Similar arguments follow also for the respective truncated sub-spaces and this helps justify a *split* of the drift space into two subspaces - both of which have strong stochastic dependencies. Whence, we observe that ignoring the term for motion compensated accumulation error from past frames does not alter the spectral characteristics (or the rate distortion behavior) of the objective function too much (or does so in a bounded manner). This corroborates the observed experimental finding that the set of optimal breakpoint values is somewhat invariant to the inclusion or exclusion of the accumulation error term. The importance of this result is that it allows us to use the much simpler memoryless DRS algorithms without any significant distortion penalty.

For the sake of completeness, we repeat the problem formulation from [49] in the next section. For a more thorough treatment, the interested reader may refer to [45][48][49][50]. For publications related to this chapter, see [14][45][46].

6.2 Problem formulation and analysis

6.2.1 Constrained DRS problem definition

We define rate shaping [48][49] as an operation which, given an input compressed video bitstream and a set of rate constraints, produces another compressed video bitstream that

complies with these constraints. For our purposes, both bitstreams are assumed to meet the same syntax specification, and we also assume that a (possibly motion-compensated) block-based transform-coding scheme is used. This includes MPEG-1, MPEG-2, and MPEG-4 as well as H.261/H.263/H.26L and so-called "motion" JPEG. If the rate constraints are allowed to vary with time, the operation will be called *dynamic* rate shaping.

The rate shaping operation is depicted in Fig. 6.1. Note that no communication path exists between the rate shaper and the source of the input bitstream, which ensures that no access to the encoder is necessary. Of particular interest is the source of the rate constraints $B_T(t)$. In the simplest of cases, $B_T(t)$ may be just a constant and known a priori (e.g., the bandwidth of a circuit-switched connection). It is also possible that $B_T(t)$ has an a-priori known statistical characterization (e.g., a policing function). Finally, another alternative is that $B_T(t)$ is generated by the network over which the output bitstream is transmitted; this could potentially be provided by the network management layer, or may be the result of end-to-end bandwidth availability estimates. The objective of a rate-shaping algorithm is to minimize the conversion distortion, i.e.,

$$\min_{B^*(t) \le B_T(t)} \{ \| y - y^* \| \}$$

Note that no assumption is made on the rate properties of the input bitstream, which can indeed be arbitrary. The attainable rate variation (B^* / B) is in practice limited, and depends primarily on the number of B pictures and the original rate of the bitstream. Also, no indication is given on the length of the optimization window, and it too can be arbitrary. Complexity and delay considerations make it desirable that it is kept small, and our interests will focus in the case where the window is up to a complete picture (frame or field).

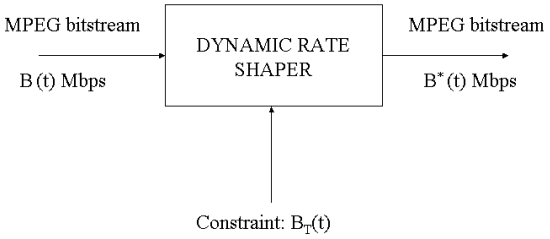


Figure 6.1: Operation of a dynamic rate shaper.

Assuming that a motion-compensated block-based transform coding technique is used to generate the input bitstream and decode the output one, there are two fundamental ways to reduce the rate (1) modifying the quantized transform coefficients by employing coarser quantization, and (2) eliminating transform coefficients. In general, both schemes can be used to perform rate shaping; re-quantization, however, leads to recoding-like algorithms which are not amenable to very fast implementation and (as will be shown later) do not perform as well as selective-transmission ones if the rate shaping ratio remains less than 2. Note that full recoding represents the brute-force approach in effecting rate changes, and falls in this category. In the rest of this chapter we only consider selective-transmission based algorithms. In particular, we examine two different cases: truncation and arbitrary selection. In the former case, a set of DCT coefficients at the end of each block is eliminated. This approach will be referred to as *constrained* DRS. In the latter case, our algorithm is allowed to arbitrarily select DCT coefficients for elimination from the bitstream, and hence will be called *unconstrained* DRS.

In constrained DRS, the number of DCT run-length codes within each block which will be kept is called the *breakpoint*, paralleling the data partitioning terminology [45]. All DCT coefficients above the breakpoint are eliminated from the bitstream. In general DRS, the breakpoint becomes a 64-element binary vector, indicating which coefficients within each 8x8 block will be kept. Figure 6.2 illustrates the difference between the two approaches. Assuming the use of MPEG, and to avoid certain syntax complications[24], we require that at least one DCT coefficient remains

[24] These include recoding the coded block patterns, and re-executing DC prediction loops.

in each block. Consequently, scalar breakpoint values range from 1 to 64, while vector ones will be non-zero.

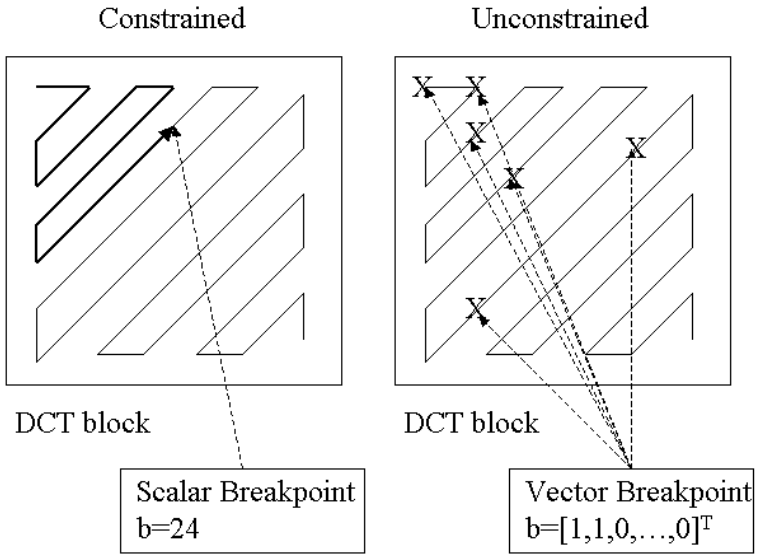


Figure 6.2: Breakpoint definition for constrained and unconstrained DRS

6.2.2 Constrained DRS of Intra coded pictures

In intra-picture rate shaping, there is no temporal dependence between pictures. Consequently, the shaping error will simply consist of the DCT coefficients that are dropped. It can then be shown [49] that the DRS problem can be expressed as follows:

$$\min_{B^*(t)\le B(t)} \{\| y - y^* \|\} \Leftrightarrow \min_{\sum_{i=1}^{N} R_i(b_i)\le B_T(t)} \{\sum_{i=1}^{N} D_i(b_i)\}$$

with

$$D_i(b_i) = \sum_{k\ge b_i} [E_i(k)]^2 \qquad (1)$$

where $b_i \in \{1,..,64\}$ is the breakpoint value for block i (run-length codes from b_i and up will be eliminated), N is the number of blocks considered, $E_i(k)$ is the value of the DCT coefficient of the k-th run in the i-th block, and $R_i(b_i)$ denotes the rate required for coding block i using a

breakpoint value of b_i.

This constrained minimization problem can be converted to an unconstrained one using Lagrange multipliers: instead of minimizing $\sum_i D_i(b_i)$ given $\sum_i R_i(b_i)$, we minimize:

$$\min\{\sum_{i=1}^{N} D_i(b_i) + l \sum_{i=1}^{N} R_i(b_i)\} \tag{2}$$

Note that the two problems are not equivalent; for some value of l, however, which our algorithm will have to find, their solutions become identical [52][144]. The interested reader may refer to [49] for a detailed description of the Lagrangian relaxation algorithm.

Note that since the selection set is discrete in nature, the above *Integer Problem* (IP) [110] may alternatively be relaxed to a linear program [18]. This approach has been well studied in similar problem setups [15] and is known to perform very close to IP optimality in time that is linear in the size of the optimization window. This complexity is provably better than that of Lagrange relaxation [52][53], Dynamic Programming [113][128] or marginal analysis [56] approaches. Similar algorithmic approaches but in different contexts have also been used in [15][113][128][211].

6.2.3 Constrained DRS of Inter coded pictures

When all picture coding types are used (I, P, and B) the problem is significantly more complex. Figure 6.3 shows the block diagram of a MC transform encoder with Huffman (VLC) coding. Using the notation from this figure, we obtain the expressions for the decoded bit-stream with and without rate shaping as follows:

$$\begin{aligned} y_{i+1} &= M_{i+1}(y_i) + e_{i+1} \text{ with } e_0 = y_0 \\ y^*_{i+1} &= M_{i+1}(y^*_i) + e^*_{i+1} \text{ with } e^*_0 = y^*_0 \end{aligned} \tag{3}$$

Here *M(.)* is the motion compensation operator and e_i is the coded prediction error. Thus, the minimization problem for the inter-coded case becomes:

$$\begin{aligned} &\text{Min.}\{||y_i - y^*_i||\} = ||M_i(y_{i-1}) - M_i(y^*_{i-1}) + e_i - e^*_i|| = ||a_i + e_i - e^*_i|| \\ &\text{Subject to: } \sum_{i=1}^{N} R_i(b_i) \le B_T(t) \end{aligned} \tag{4}$$

Here a_i is the accumulated error due to motion compensation based on (optimal) truncation of DCT coefficients in the past frame, while e_i-e_i* is current-frame only distortion due to rate shaping of transformed prediction error. Note that in general $M_i(y_{i-1}) - M_i(y^*_{i-1}) \neq M_i(y_{i-1} - y^*_{i-1})$, i.e., motion compensation is a non-linear operation, because it involves integer arithmetic with truncation away from zero.

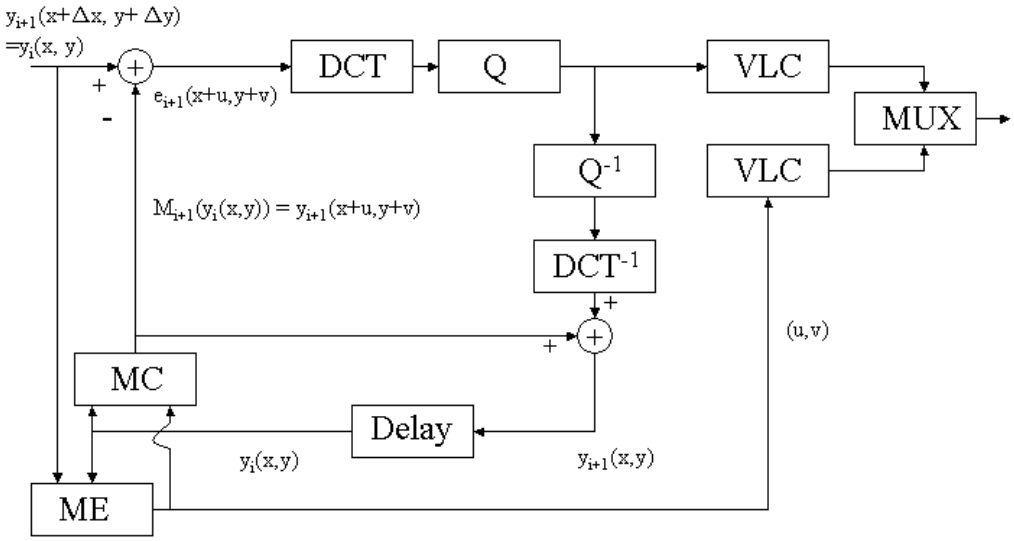


Figure 6.3: Motion compensated transform encoder.

From (3) and (4) we observe that, in contrast with the intra-only case, optimization involves the accumulated error a_i. Furthermore, due to the error accumulation process, rate-shaping decisions made for a given picture will have an effect on the quality and partitioning decisions of subsequent pictures. As a result, an optimal algorithm for (4) would have to examine a complete group of pictures (I-to-I), since breakpoint decisions at the initial I-picture may affect even the last B or P picture. Not only the computational overhead would be extremely high, but the delay would be unacceptable as well.

An attractive alternative algorithm is one that solves (4) on a picture basis, and where only the

error accumulated from past pictures is taken into account; this algorithm will be referred to as *causally optimal*. Note that in order to accurately compute a_i, two prediction loops have to be maintained (one for a decoder that receives the complete signal, and one for a decoder that receives only partition 0). This is because of the non-linearity of motion compensation, which involves integer arithmetic with truncation away from zero. With the penalty of some lack in arithmetic accuracy, these two loops can be collapsed together.

The causally optimal problem can be formulated as follows:

$$\min_{\sum_{i=1}^{N} R_i(b_i) \le B_T} \{\sum_{i=1}^{N} \hat{D}_i(b_i)\} \tag{5}$$

with

$$\hat{D}_i(b_i) = \sum_k A_i(k)^2 + \sum_{k \ge b_i} 2A^i(x(k))E^i(k) + \sum_{k \ge b_i} E^i(k)^2 \tag{6}$$

where N is such that a complete picture is covered, $A^i(k)$ is the k-th DCT coefficient (in zig-zag scan order) of the of the i-th block of the accumulated error a_i, and $x(.)$ maps run/length positions from the prediction error $E^i(.)$ to actual zig-zag scan positions. This minimization problem can be solved using the Lagrangian multiplier approach, with the new definition for the distortion $\hat{D}$.

The complexity of solving (5) is significant, and can be shown to be between that of a decoder and an encoder. In order to examine the benefit of error accumulation tracking, one can apply the intra-only algorithm to the mixed-mode case, since the only difference is the accumulated error term a_i. Surprisingly, the results of this *memoryless* mixed-mode partitioning algorithm are almost identical. Figure 6.4 (reproduced from [49]) shows the relevant PSNR values for the "Mobile" sequence; the difference of the causally optimal and memoryless algorithms is in general less than 0.1 dB and the curves can hardly be distinguished. It turns out that this holds for a wide range of bit rates, although the difference increases slightly to 0.2-0.3 dB. This is a very important result, as it implies that we can dispense completely with the error accumulation calculation and its associated computational complexity, for a minimal cost in performance.

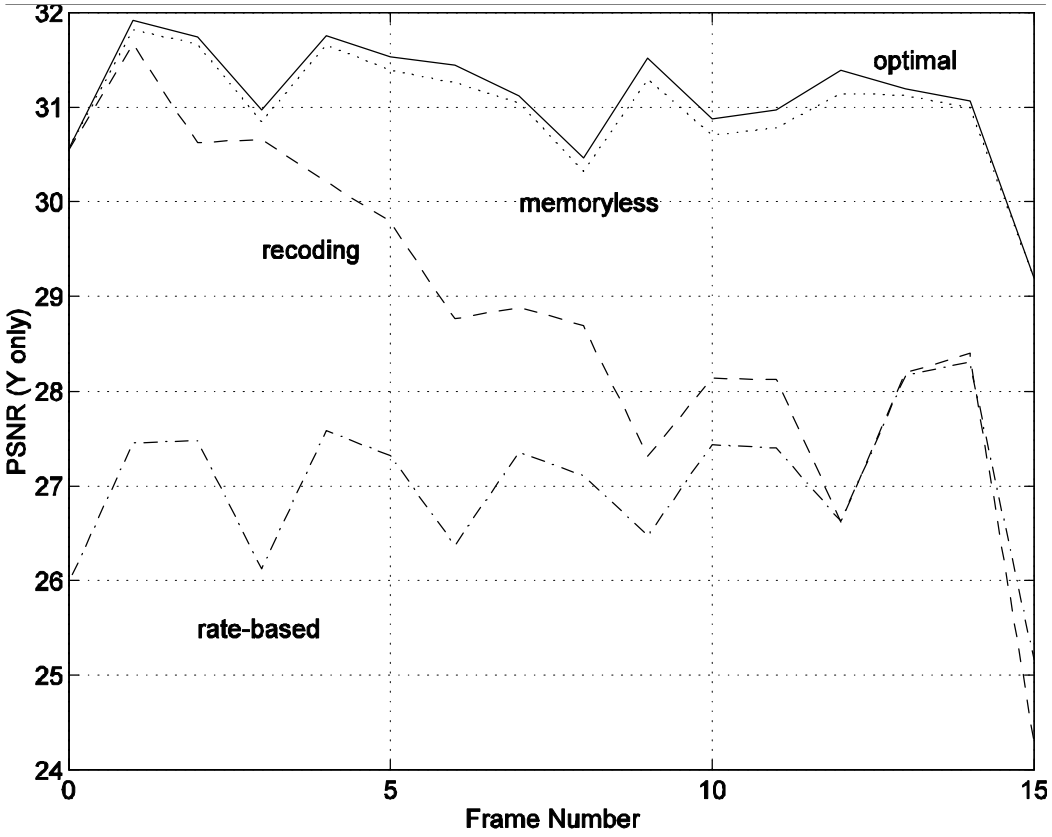
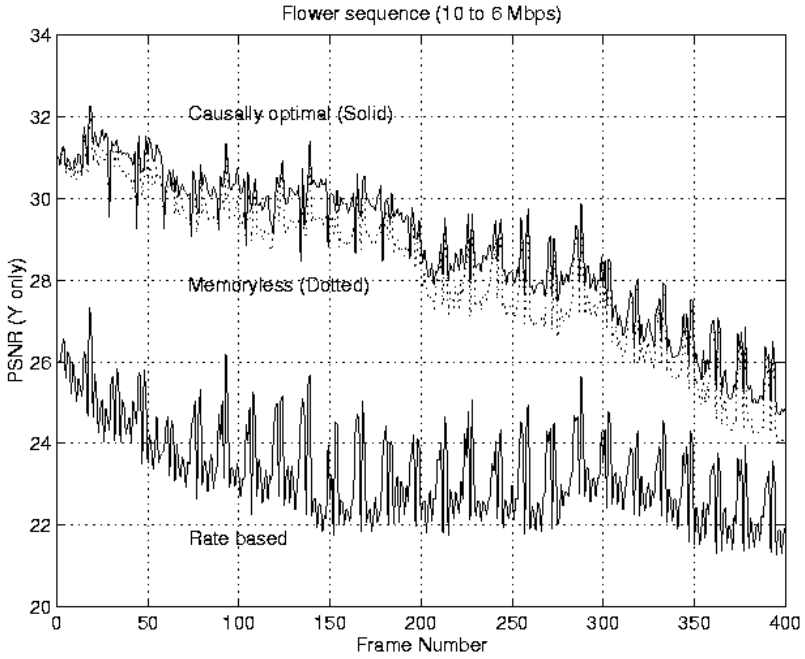


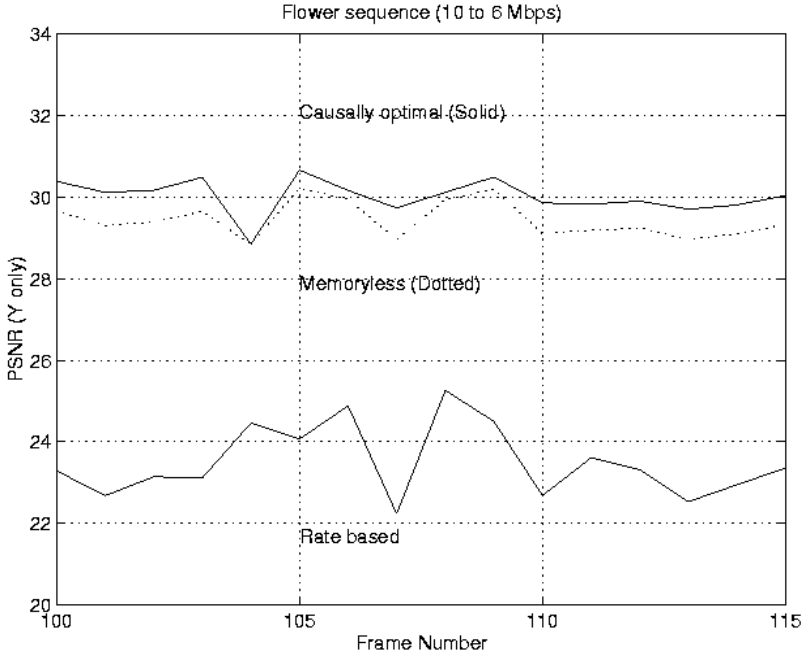
Figure 6.4: Results of various rate-shaping algorithms on the "Mobile" sequence, MPEG-2 coded at 4 Mbps and rate shaped at 3.2 Mbps (reproduced from [49] with author's permission).

To further substantiate our claim that the quality degradation between the causally optimal and memoryless algorithms is perceptually insignificant across a large spectrum of cluster sizes and partition rates, we present a far more extensive set of simulation results in Figure 6.5, Figure 6.6 and Figure 6.7. The compliance bitstreams used in these simulations were downloaded from `ftp://ftp.tek.com/tv/test/streams/Element/index.html` and the MPEG-2 Video Encoder (Version 1.1) from MPEG Software Simulation Group was used to re-encode the streams. Figure 6.5 shows results for various DRS algorithms on 400 frames from the Flower sequence originally encoded at 10 Mbps and shaped to 6.5 and 6 Mbps. Similar results are shown in Figure 6.6 for the Table Tennis sequence. As witnessed for the Mobile sequence shaped by about 80%, these plots reconfirm that the memoryless algorithm is a very good approximation and performs within 1 dB of the causally optimal one. The rate-based algorithm, which ignores the objective function completely, but instead does a *maximal* breakpoint selection

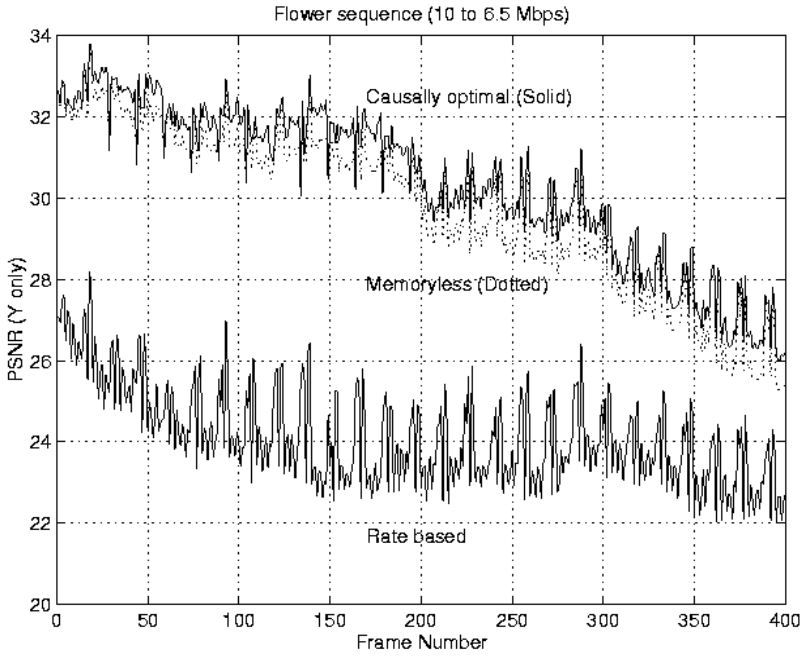
based solely on a sequence of localized rate constraints, is seen to under-perform the memoryless algorithm by about 8-10 dB. Figure 6.7 shows the average PSNR values for a wide spectrum of shaping rates for several commonly used sequences. Yet again, the memoryless algorithm performs extremely close to the causally optimal one, with a difference of only 0.1 to 0.2 dB for a wide range of rate shaping ratios. The performance of the DRS approach is seen to fall off much faster than recoding (with no reuse of motion vectors or other coding parameters). At a ratio of about 2:1 and higher, recoding is seen to out-perform dynamic rate shaping. At such drastic rate shaping ratios, requantization (or recoding) offers a better alternative compared to the (computationally more attractive) DRS approach since the former allows one to re-distribute the quantization error among the various coefficients based on their statistical characteristics at the transcoded rate and not at the original encoding rate.



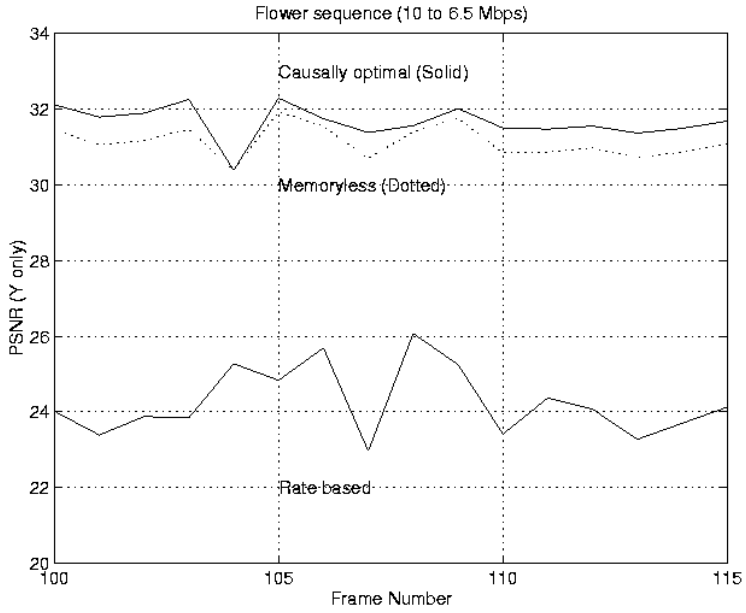
(a)


Flower sequence (10 to 6 Mbps)
Causally optimal (Solid)
Memoryless (Dotted)
Rate based
PSNR (Y only)
Frame Number
34
32
30
28
26
24
22
20
100
105
110
115

(b)

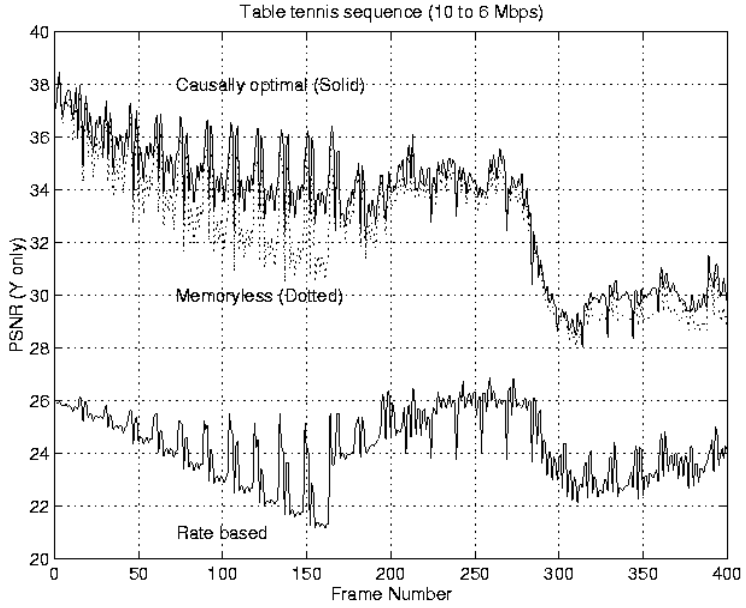

Flower sequence (10 to 6.5 Mbps)
Causally optimal (Solid)
Memoryless (Dotted)
Rate based
PSNR (Y only)
Frame Number
34
32
30
28
26
24
22
20
0
50
100
150
200
250
300
350
400

(c)



(d)

Figure 6.5: PSNR (Y only) for the "Flower" sequence, coded at 10 Mbps and shaped to 6.5 and 6 Mbps using DRS with causally optimal, memoryless or rate based algorithms.



(a)

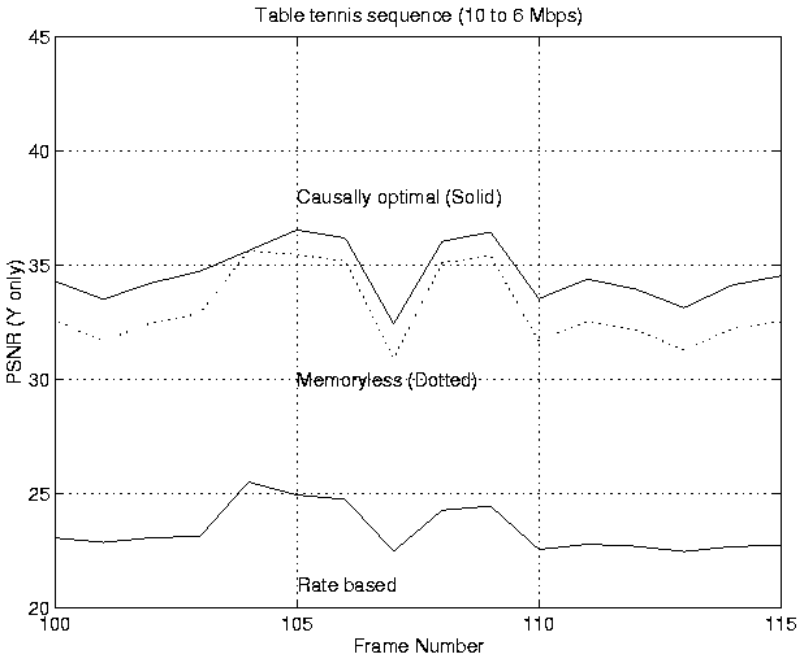


Table tennis sequence (10 to 6 Mbps)
Causally optimal (Solid)
Memoryless (Dotted)
Rate based
PSNR (Y only)
Frame Number
45
40
35
30
25
20
100
105
110
115

(b)

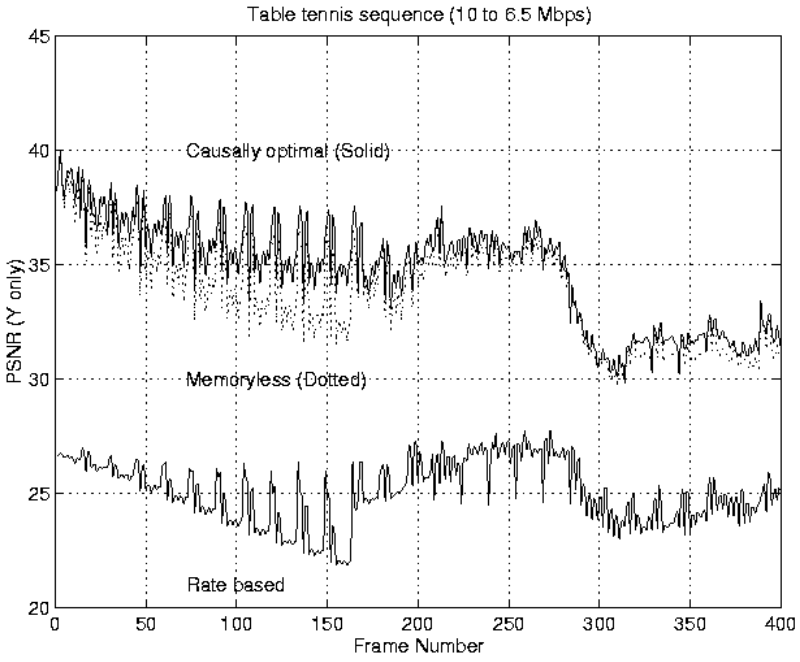
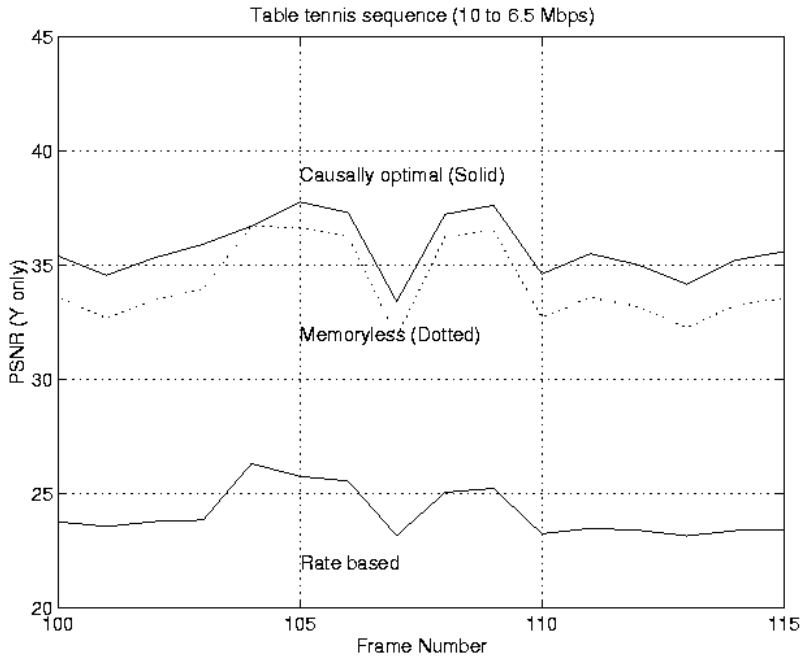


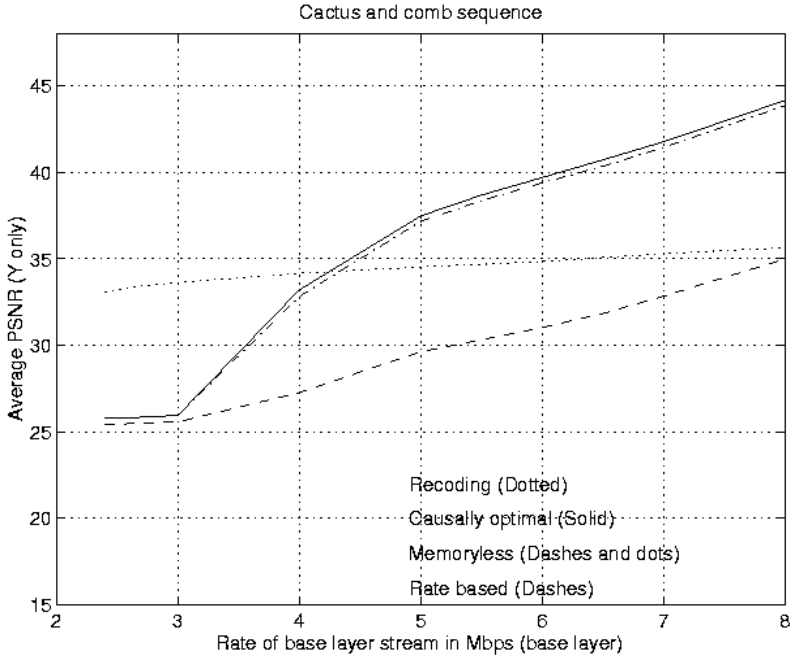
Table tennis sequence (10 to 6.5 Mbps)
Causally optimal (Solid)
Memoryless (Dotted)
Rate based
PSNR (Y only)
Frame Number
45
40
35
30
25
20
0
50
100
150
200
250
300
350
400

(c)

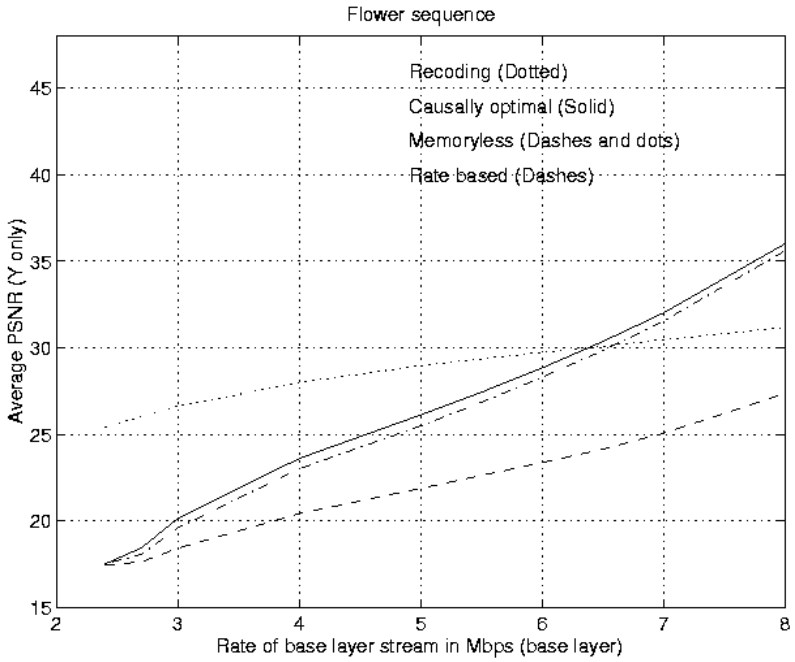


(d)

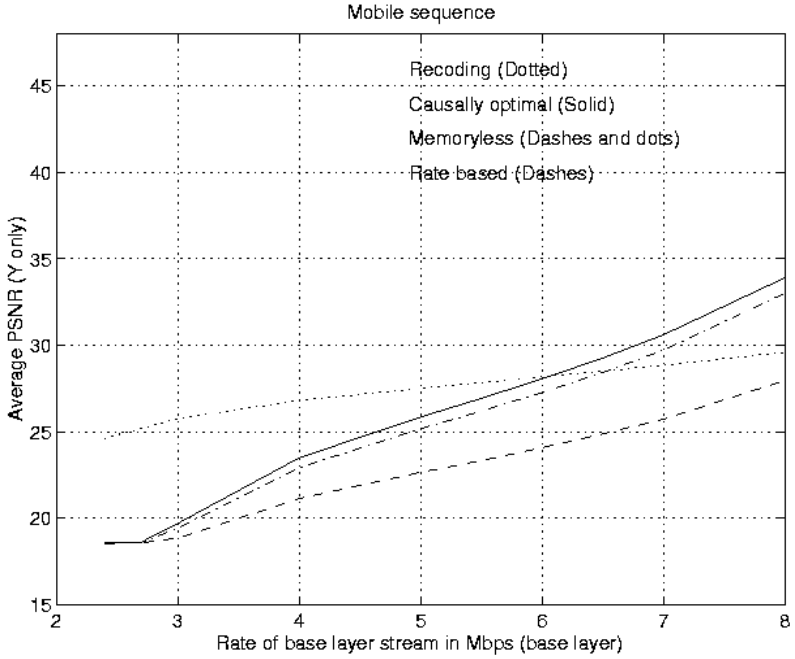
Figure 6.6: PSNR (Y only) for the "Table Tennis" sequence coded at 10 Mbps and shaped to 6.5 and 6 Mbps using DRS with causally optimal, memoryless or rate based algorithms.


Cactus and comb sequence
Average PSNR (Y only)
Rate of base layer stream in Mbps (base layer)
Recoding (Dotted)
Causally optimal (Solid)
Memoryless (Dashes and dots)
Rate based (Dashes)
45
40
35
30
25
20
15
2
3
4
5
6
7
8

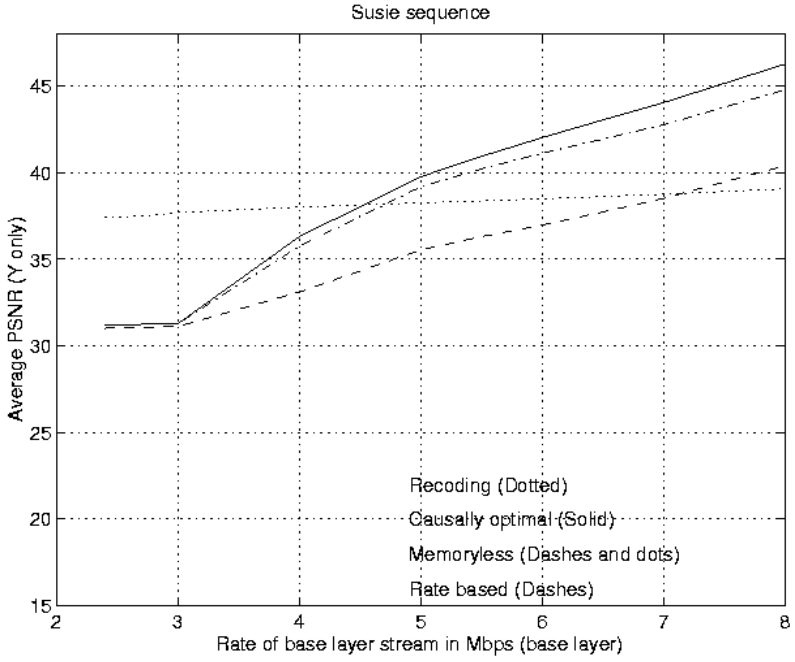
(a)


Flower sequence
Average PSNR (Y only)
Rate of base layer stream in Mbps (base layer)
Recoding (Dotted)
Causally optimal (Solid)
Memoryless (Dashes and dots)
Rate based (Dashes)
45
40
35
30
25
20
15
2
3
4
5
6
7
8

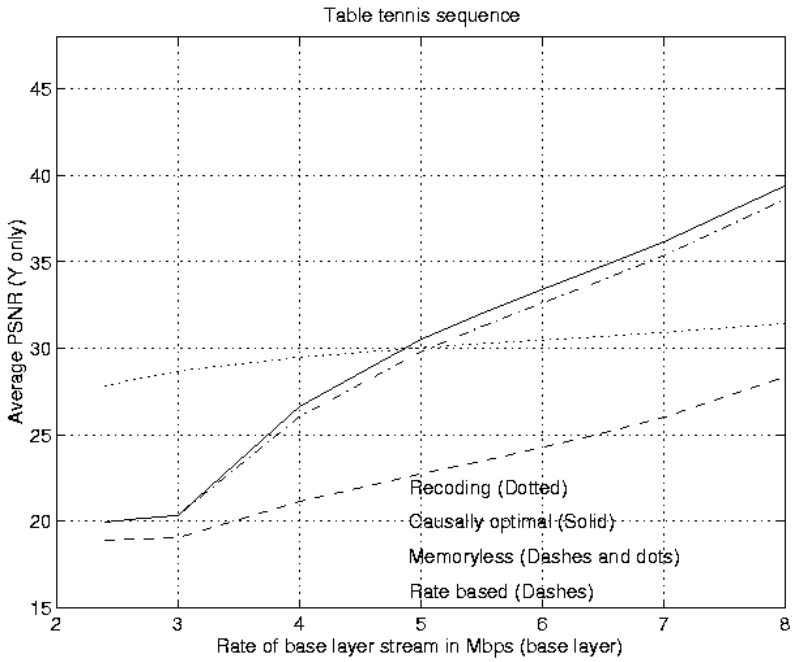
(b)


Mobile sequence
Recoding (Dotted)
Causally optimal (Solid)
Memoryless (Dashes and dots)
Rate based (Dashes)
Average PSNR (Y only)
Rate of base layer stream in Mbps (base layer)
15
20
25
30
35
40
45
2
3
4
5
6
7
8

(c)


Susie sequence
Recoding (Dotted)
Causally optimal (Solid)
Memoryless (Dashes and dots)
Rate based (Dashes)
Average PSNR (Y only)
Rate of base layer stream in Mbps (base layer)
15
20
25
30
35
40
45
2
3
4
5
6
7
8

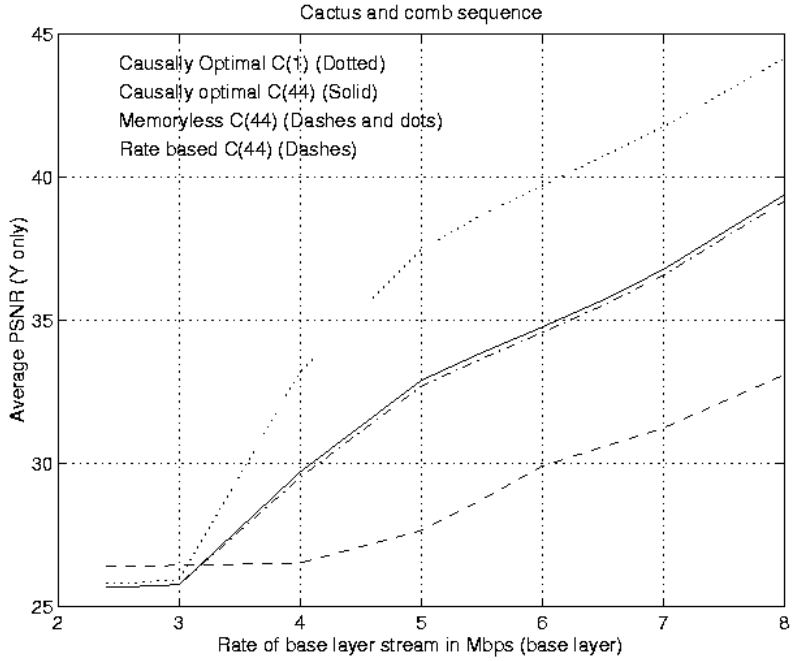
(d)



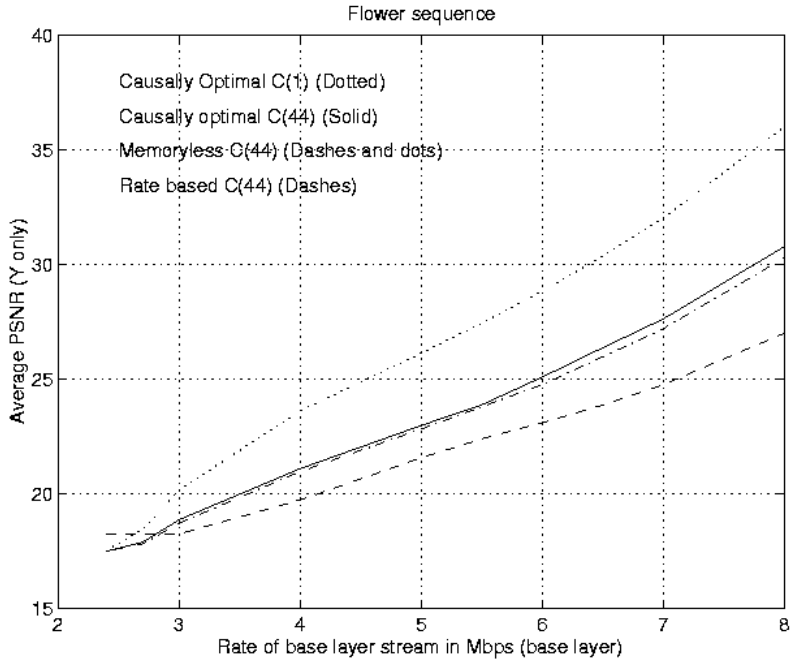
(e)

Figure 6.7: Results of various dynamic rate-shaping algorithms on different MPEG-2 Sequences encoded at 10 Mbps and transcoded to different target rates.

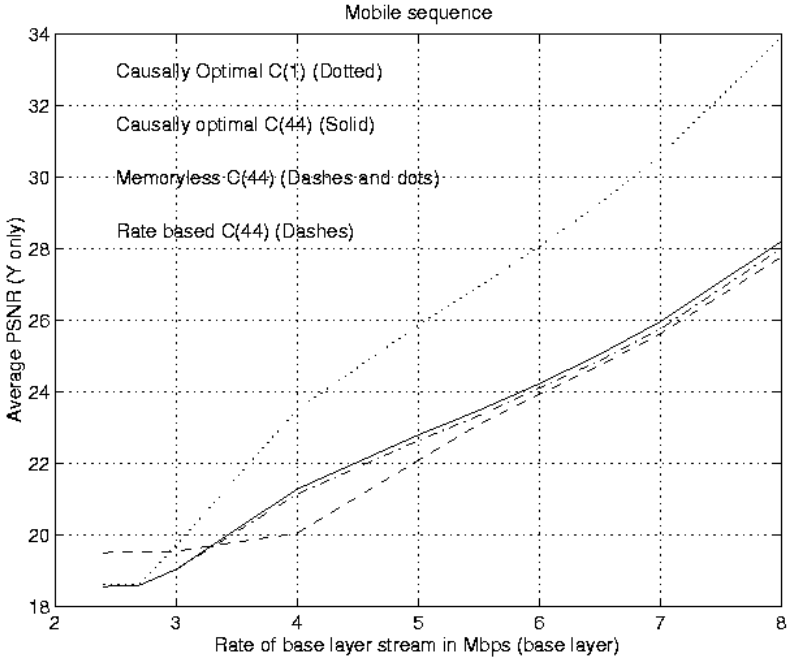
One way to reduce the complexity of the DRS problem is by a so called clustering approach [49] in which a common breakpoint is selected for a set of blocks. Denote by *C(n)* the clustered DRS algorithm where *n* consecutive blocks have a common breakpoint. Figure 6.8 compares the performance of *C(1)* and *C(44)* algorithms for five commonly used test sequences. As can be seen from these plots, *C(1)* outperforms *C(44)* by about 2-5 dB at a rate shaping ratio of about 2:1. The gap is higher for lower rate shaping ratios and shrinks as the rate shaping ratio increases.


Cactus and comb sequence
Causally Optimal C(1) (Dotted)
Causally optimal C(44) (Solid)
Memoryless C(44) (Dashes and dots)
Rate based C(44) (Dashes)
Average PSNR (Y only)
Rate of base layer stream in Mbps (base layer)
45
40
35
30
25
2
3
4
5
6
7
8

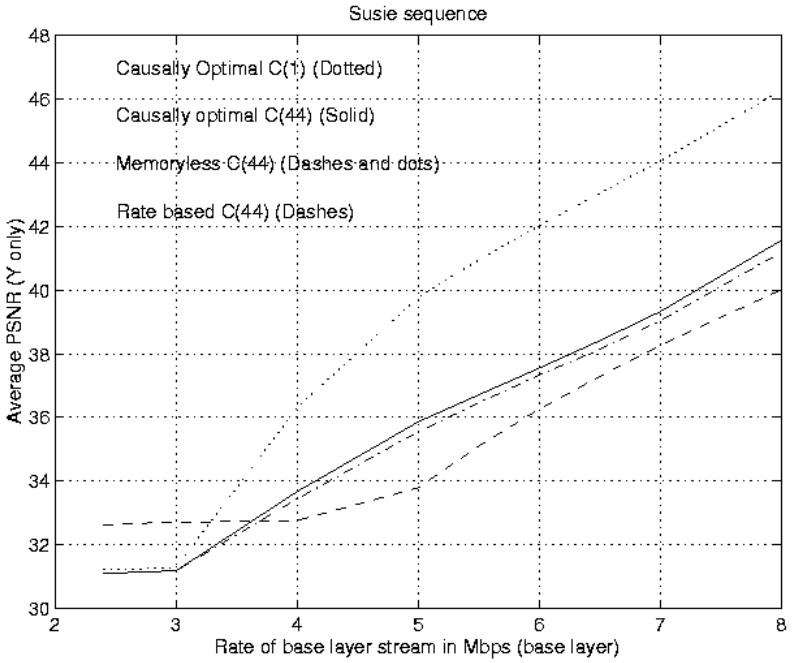
(a)


Flower sequence
Causally Optimal C(1) (Dotted)
Causally optimal C(44) (Solid)
Memoryless C(44) (Dashes and dots)
Rate based C(44) (Dashes)
Average PSNR (Y only)
Rate of base layer stream in Mbps (base layer)
40
35
30
25
20
15
2
3
4
5
6
7
8

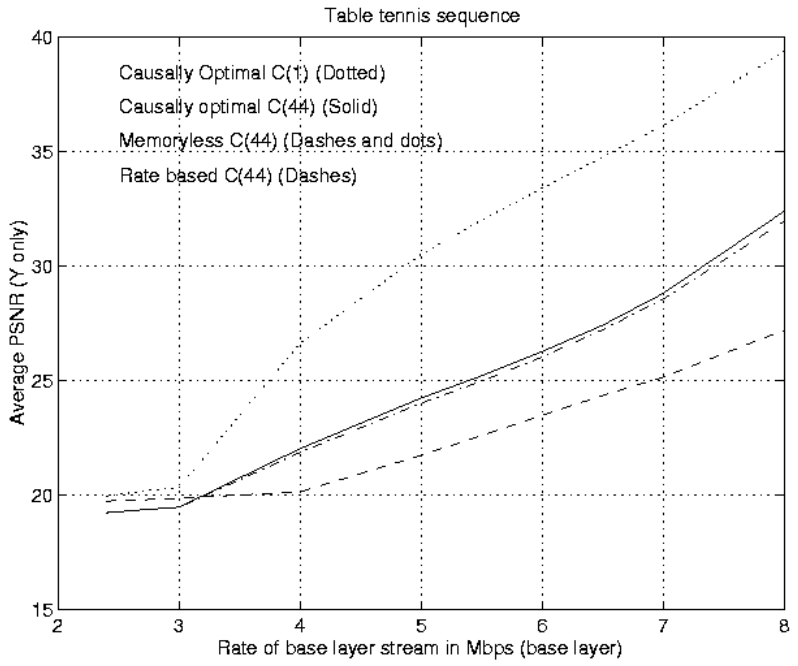
(b)


Mobile sequence
Causally Optimal C(1) (Dotted)
Causally optimal C(44) (Solid)
Memoryless C(44) (Dashes and dots)
Rate based C(44) (Dashes)
Average PSNR (Y only)
Rate of base layer stream in Mbps (base layer)
34
32
30
28
26
24
22
20
18
2
3
4
5
6
7
8

(c)


Susie sequence
Causally Optimal C(1) (Dotted)
Causally optimal C(44) (Solid)
Memoryless C(44) (Dashes and dots)
Rate based C(44) (Dashes)
Average PSNR (Y only)
Rate of base layer stream in Mbps (base layer)
48
46
44
42
40
38
36
34
32
30
2
3
4
5
6
7
8

(d)



(e)

Figure 6.8: Comparison of PSNR performance of C(44) and C(1) for different MPEG-2 Sequences encoded at 10 Mbps and transcoded to different target rates.

Tables 6.1 through 6.5 compare the complexity of solving the *C(1)* DRS algorithm with that of the *C(44)* algorithm for the different test sequences considered. The times we report were measured on an IBM ThinkPad® machine with a 1.80 GHz Intel® Pentium® 4 Mobile CPU and 256 MB of RAM running the Microsoft Windows XP Professional 2002 operating system. We make the following key observations from these tables:

- The rate based and causally optimal DRS algorithms take between 300-400 msec to rate shape each frame. This amounts to an almost real-time performance of around 3 frames per second.
- The memoryless algorithm is considerably more efficient than the causally optimal one and saves around 100 msec of processing time per frame. With the memoryless algorithm one can transcode about 4 frames per second.
- For all the algorithms considered (causally optimal, memoryless and rate based), the *C(1)* times are almost the same as the *C(44)* times. Hence, complexity-wise there is virtually no

difference between *C(1)* and *C(44),* while the former gives a much better performance in terms of Y-PSNR and should be the obvious choice for practical implementations.

- Bitstream decoding and R-D data collection takes about 80% of the processing time and is by far the most resource intensive operation.
- Bitstream output takes around 19-20% of the processing time.
- Finally, the actual optimization algorithm with the Lagrangian iterations and bisection takes less than 1% of the time.

Algorithm	***Stage***			
	Overall Time/Frame (msec)	**Decoding and R-D data collection**	**Bitstream Output**	**Optimization**
C(1) Causally Optimal	378	81.4 %	17.85 %	0.72 %
C(1) Memoryless	247.75	74.36 %	24.52 %	1.11 %
C(1) Rate Based	376.5	82.67 %	16.99 %	0.33 %
C(44) Causally Optimal	379.5	81.4 %	18.6 %	˜ 0 %
C(44) Memoryless	251	67.3 %	32.56 %	0.09 %
C(44) Rate Based	380	79.8 %	20.12 %	˜ 0 %

Table 6.1: Comparison of the complexity of various stages of the *C(1)* and *C(44)* algorithms for the "Cactus and Comb sequence".

Algorithm	***Stage***			
	Overall Time/Frame (msec)	**Decoding and R-D data collection**	**Bitstream Output**	**Optimization**
C(1) Causally Optimal	332.75	80.69 %	18.33 %	0.90 %
C(1) Memoryless	221.5	72.91 %	25.51 %	1.59 %
C(1) Rate Based	330	80.68 %	19.01 %	0.30 %
C(44) Causally Optimal	336.25	81.19 %	34.81 %	˜ 0 %
C(44) Memoryless	225.5	65.08 %	34.81 %	˜ 0 %
C(44) Rate Based	335.5	76.52 %	23.47 %	˜ 0 %

Table 6.2: Comparison of the complexity of various stages of the *C(1)* and *C(44)* algorithms for the "Flower sequence".

Algorithm	***Stage***			
	Overall Time/Frame (msec)	**Decoding and R-D data collection**	**Bitstream Output**	**Optimization**
C(1) Causally Optimal	323.5	80.91 %	18.16 %	0.92 %
C(1) Memoryless	212	70.16 %	28.06 %	1.77 %
C(1) Rate Based	319.25	81.28 %	18.48 %	0.23 %
C(44) Causally Optimal	323	77.55 %	22.37 %	0.078 %

C(44) Memoryless	215.5	68.44 %	31.56 %	˜ 0 %
C(44) Rate Based	325.75	80.04 %	19.95 %	˜ 0 %

Table 6.3: Comparison of the complexity of various stages of the *C(1)* and *C(44)* algorithms for the "Mobile sequence".

Algorithm	***Stage***			
	Overall Time/Frame (msec)	**Decoding and R-D data collection**	**Bitstream Output**	**Optimization**
C(1) Causally Optimal	375.5	82.09 %	17.37 %	0.53 %
C(1) Memoryless	241.5	73.08 %	25.57 %	1.53 %
C(1) Rate Based	373.25	81.5 %	18.35 %	0.13 %
C(44) Causally Optimal	380.75	80.17 %	19.76 %	0.06 %
C(44) Memoryless	244	68.95 %	31.04 %	˜ 0 %
C(44) Rate Based	377.5	80.13 %	19.87 %	˜ 0 %

Table 6.4: Comparison of the complexity of various stages of the *C(1)* and *C(44)* algorithms for the "Susie sequence".

Algorithm	***Stage***			
	Overall Time/Frame (msec)	**Decoding and R-D data collection**	**Bitstream Output**	**Optimization**
C(1) Causally Optimal	349.25	82.61 %	16.82 %	0.57 %
C(1) Memoryless	224.5	71.93 %	26.95 %	1.11 %
C(1) Rate Based	349.5	81.97 %	17.73 %	0.28 %
C(44) Causally Optimal	379.5	79.5 %	20.47 %	˜ 0 %
C(44) Memoryless	230.75	68.69 %	31.31 %	˜ 0 %
C(44) Rate Based	352.5	78.37 %	21.63 %	˜ 0 %

Table 6.5: Comparison of the complexity of various stages of the *C(1)* and *C(44)* algorithms for the "Table Tennis sequence".

6.2.4 Unconstrained rate shaping

In this section, we concentrate on the generalized version of the DRS problem, discussing its performance particularly in comparison to the constrained DRS approach. Let $\vec{b}_i = \{b_i^k \in \{0,1\},\ k = 1,..,K\}^T$ denote the breakpoint vector for block *i*, where b_i^k is 1 iff the *k*-th DCT coefficient run-length of block *i*-th is retained. In order to avoid a perceptually ill-defined distortion, we only include the luminance component in our distortion calculations. As a result, breakpoint decisions will be made on a per macroblock basis, i.e., all blocks of a given

macroblock will use the same breakpoint vector. Rate calculations include, of course, the two-chrominance components. It can then be shown that the optimal DRS problem for intra-coded pictures can be formulated as:

$$\min_{\sum_{i=1}^{N} R_i(\vec{b}_i) \le B_T} \{\sum_{i=1}^{N} D_i(\vec{b}_i)\} \tag{7}$$

with

$$D_i(\vec{b}_i) = \sum_{j \in \Upsilon} \sum_{k} \overline{b_i^k} E_j^i(k)^2 \tag{8}$$

where N is the number of macroblocks over which optimization takes place, Υ denotes the luminance blocks of a macroblock, $R_i(.)$ denotes the number of bits required to code macroblock i with the breakpoint configuration $\vec{b}_i$, $D_i(\vec{b}_i)$ denotes the distortion associated with $\vec{b}_i$, $E_j^i(k)$ is the value of the DCT coefficient of the k-th run in the j-th block of the i-th macroblock, and $\overline{b_i^k}$ is the logical inverse of b_i^k.

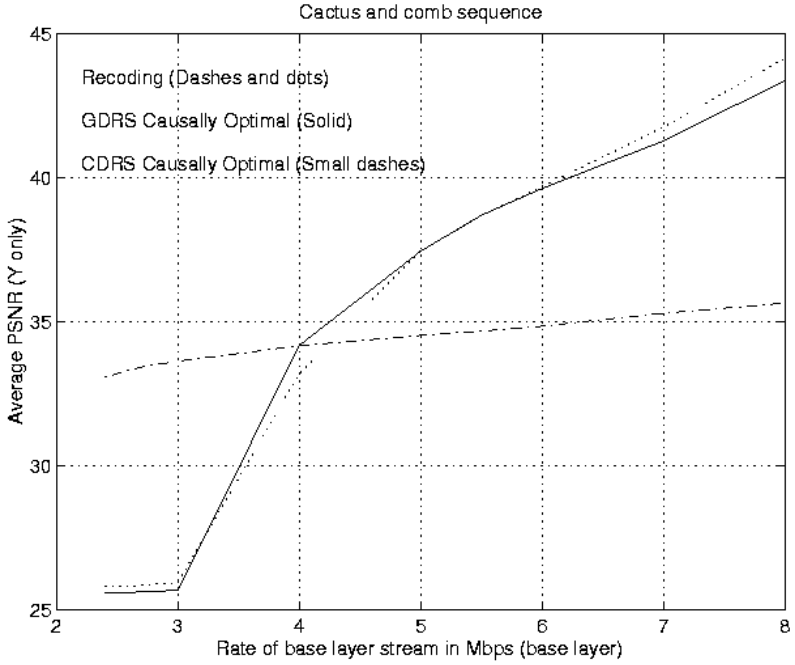
Using the causality argument and taking into account only the accumulated error, it can be shown that (7), (8) can be generalized to include P and B pictures by defining the distortion as follows:

$$D_i(\vec{b}_i) = \sum_{y \in \Upsilon} \left[\sum_{k} A_j^i(k)^2 + \sum_{k} 2\overline{b_i^k} A_j^i(x_j^i(k)) E_j^i(k) + \sum_{k} \overline{b_i^k} E^i(k)^2 \right] \tag{9}$$

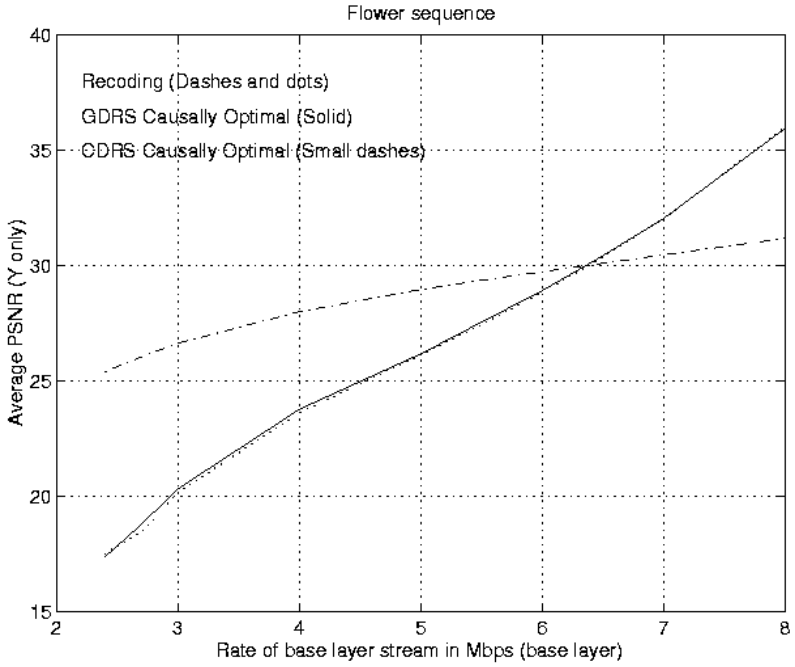
where $A_j^i(k)$ is the k-th DCT coefficient (in zig-zag scan order) of the i-th block of the j-th macroblock of accumulated error, and $x_j^i(.)$ maps run-length positions to zig-zag scan positions.

Yet again, the constrained minimization problem of (7) and (9) can be converted to an unconstrained one using Lagrange multipliers. A fast, iterative algorithm for the determination of the optimal l can be found based on bisection [52] or a descent algorithm [53]. For more details on this algorithm, refer to [46][49]. A key characteristic of this algorithm is that it operates in the convex hull of the "operational" *R(D)* curves of each block. Initial experimental

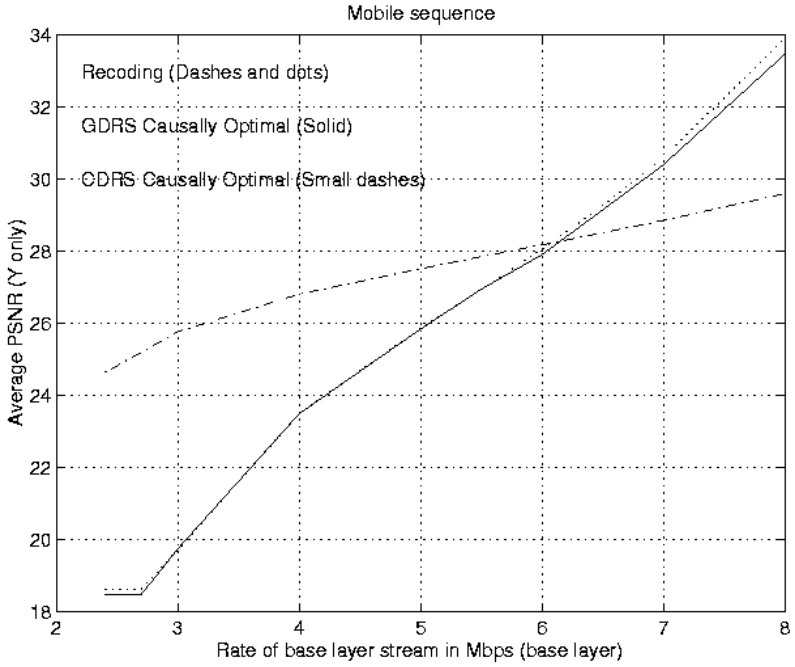
results for optimal unconstrained DRS with the "Mobile" sequence coded (using MPEG-2) at 4 Mbps and rate-shaped at 3.2 Mbps were reported in [49]. As with CDRS, these results showed that the memoryless algorithm performs almost identical to the optimal one. To supplement these observations, we present a more extensive set of simulation results comparing the performance of GDRS and CDRS in Figure 6.8. As can be seen from these plots, unconstrained and constrained DRS perform almost identically. This is likely to be due to the efficiency of the zig-zag scanning order for DCT coefficients. Alternatively, one may argue as well that the non-sequential (somewhat random) selection of breakpoints produces a distribution of (run-length, value) pairs for which the original set of VLC tables were not optimized, hence giving worse R-D performance. Since VLC tables are pre-specified by the standard, one generally does not have a freedom to re-optimize in this dimension if a standard-compliant transcoded bitstream is required.



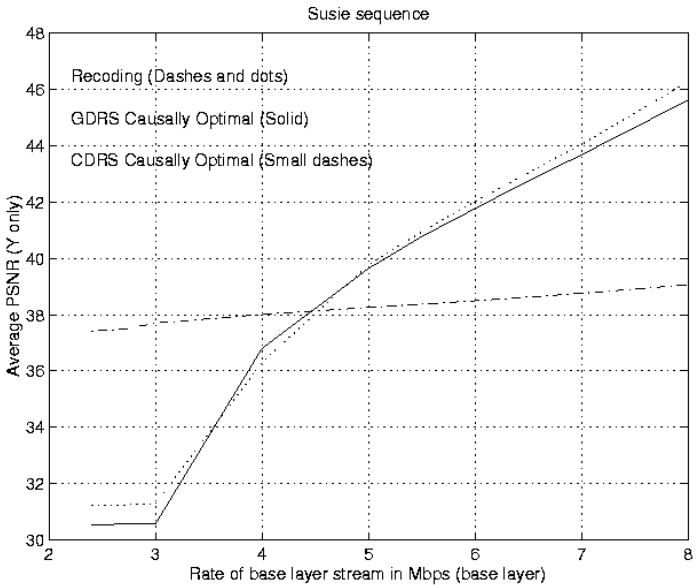
(a)


Flower sequence
Recoding (Dashes and dots)
GDRS Causally Optimal (Solid)
CDRS Causally Optimal (Small dashes)
Average PSNR (Y only)
Rate of base layer stream in Mbps (base layer)
40
35
30
25
20
15
2
3
4
5
6
7
8

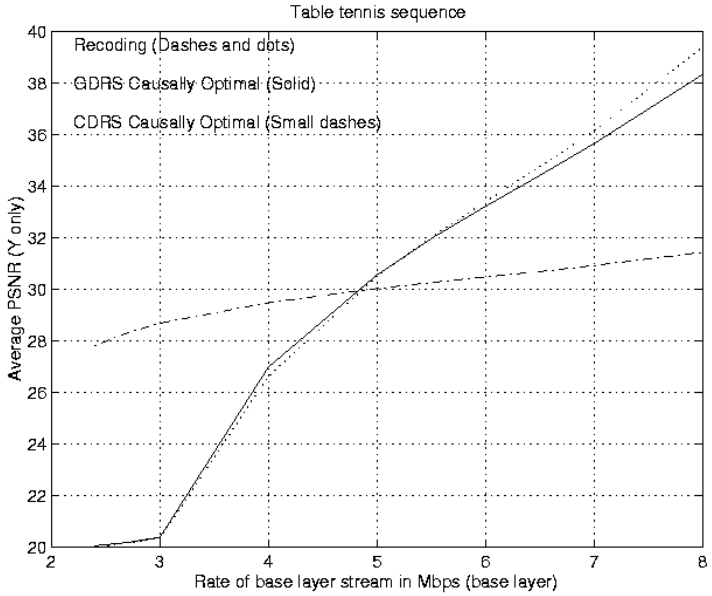
(b)


Mobile sequence
Recoding (Dashes and dots)
GDRS Causally Optimal (Solid)
CDRS Causally Optimal (Small dashes)
Average PSNR (Y only)
Rate of base layer stream in Mbps (base layer)
34
32
30
28
26
24
22
20
18
2
3
4
5
6
7
8

(c)



(d)



(e)

Figure 6.9: Comparison of PSNR performance of CDRS and GDRS for different MPEG-2 Sequences encoded at 10 Mbps and transcoded to different target rates. Also shown is the performance of a recoding based scheme.

Tables 6.6 through 6.10 compare the complexity of *CDRS C(1)* algorithm with the *GDRS* algorithm for the different test sequences considered. The times we report were measured on a machine with a 1.80 GHz Intel® Pentium® 4 Mobile CPU and 256 MB of RAM running the Microsoft Windows XP Professional 2002 operating system. We make the following key observations from these tables:

- For both the causally optimal and memoryless algorithms, *CDRS* is much more efficient than *GDRS*. The ratio of the *CDRS* to *GDRS* times is roughly 1:2.5 for the causally optimal approach and about 1:3.5 for the memoryless algorithm. Since the *GDRS* and *CDRS* algorithms have virtually no difference performance-wise, *CDRS* would be the obvious choice for all practical implementations.
- The distribution of times spent in the various stages of a *GDRS* transcoder is drastically different from that of a *CDRS* transcoder. For the causally optimal *GDRS* approach, the processing time gets distributed as 35%, 10% and 55% for the decoding and R-D data collection, bitstream output and optimization stages, respectively. For the memoryless *GDRS* algorithm, the distribution is approximately 25%, 10% and 65% for these stages. Contrast this with the 80%, 19%, and 1% distribution for the all the *CDRS* algorithms and note that the *CDRS* schemes spend virtually no time in the actual optimization algorithm.

Algorithm	***Stage***			
	Overall Time/Frame (msec)	**Decoding and R-D data collection**	**Bitstream Output**	**Optimization**
CDRS Causally Optimal	378	81.4 %	17.85 %	0.72 %
CDRS Memoryless	247.75	74.36 %	24.52 %	1.11 %
CDRS Rate Based	376.5	82.67 %	16.99 %	0.33 %
GDRS Causally Optimal	829	40.41 %	11.49 %	48.10 %
GDRS Memoryless	694.25	29.46 %	13.25 %	57.30 %

Table 6.6: Comparison of the complexity of various stages of the *CDRS* and *GDRS* algorithms for the "Cactus and Comb sequence".

Algorithm	***Stage***			
	Overall Time/Frame (msec)	**Decoding and R-D data collection**	**Bitstream Output**	**Optimization**
CDRS Causally Optimal	332.75	80.69 %	18.33 %	0.90 %
CDRS Memoryless	221.5	72.91 %	25.51 %	1.59 %
CDRS Rate Based	330	80.68 %	19.01 %	0.30 %

GDRS Causally Optimal	990.5	29.40 %	10.50 %	60.1 %
GDRS Memoryless	881.75	20.69 %	11.62 %	67.68 %

Table 6.7: Comparison of the complexity of various stages of the *CDRS* and *GDRS* algorithms for the "Flower sequence".

Algorithm	***Stage***			
	Overall Time/Frame (msec)	**Decoding and R-D data collection**	**Bitstream Output**	**Optimization**
CDRS Causally Optimal	323.5	80.91 %	18.16 %	0.92 %
CDRS Memoryless	212	70.16 %	28.06 %	1.77 %
CDRS Rate Based	319.25	81.28 %	18.48 %	0.23 %
GDRS Causally Optimal	1000.5	27.81 %	10.32 %	61.87 %
GDRS Memoryless	894.25	18.48 %	10.48 %	71.04 %

Table 6.8: Comparison of the complexity of various stages of the *CDRS* and *GDRS* algorithms for the "Mobile sequence".

Algorithm	***Stage***			
	Overall Time/Frame (msec)	**Decoding and R-D data collection**	**Bitstream Output**	**Optimization**
CDRS Causally Optimal	375.5	82.09 %	17.37 %	0.53 %
CDRS Memoryless	241.5	73.08 %	25.57 %	1.35 %
CDRS Rate Based	373.25	81.5 %	18.35 %	0.13 %
GDRS Causally Optimal	820.75	41.18 %	11.48 %	47.33 %
GDRS Memoryless	686.5	28.69 %	12.05 %	59.25 %

Table 6.9: Comparison of the complexity of various stages of the *CDRS* and *GDRS* algorithms for the "Susie sequence".

Algorithm	***Stage***			
	Overall Time/Frame (msec)	**Decoding and R-D data collection**	**Bitstream Output**	**Optimization**
CDRS Causally Optimal	349.25	82.61 %	16.82 %	0.57 %
CDRS Memoryless	224.5	71.93 %	26.95 %	1.11 %
CDRS Rate Based	349.5	81.97 %	17.73 %	0.28 %
GDRS Causally Optimal	936.75	32.66 %	10.06 %	57.25 %
GDRS Memoryless	813	22.32 %	11.90 %	65.77 %

Table 6.10: Comparison of the complexity of various stages of the *CDRS* and *GDRS* algorithms for the "Table Tennis sequence".

6.3 Problem Analysis for Rate Shaping of Markov-1 Sources

Motivated by the performance of the memoryless algorithm, this section details the behavior of the above minimization problem for the motion compensated (inter) frame case assuming an auto-regressive (AR) source model [14]. By analyzing the statistical and rate-distortion behavior of the different components of this minimization problem, the following key result is mathematically shown: *The set of optimal breakpoint values for any frame is somewhat invariant to the accumulated motion compensated shaping error from past frames and may be very reasonably approximated using the current frame shaping error alone*.

6.3.1 Source Model

The following sections characterize the statistical and R-D behavior of each of the different components of equations (3) and (4). We model the pixel values of intra-frame images using a first order Markov process [84]. An Autoregressive-1 (AR(1)) process is a wide-sense stationary process characterized by an exponentially decreasing auto-correlation function (ACF). In the 2-dimensional case the ACF $R(x,y)=R(0,0)\rho_x^{|x|}\rho_y^{|y|}$, which assuming that $\rho_x=\rho_y=\rho$ yields $R(x,y)=R(0,0)\rho^{|x|+|y|}$. A correlation coefficient (ρ) of 0.9-0.98 is known to model natural images quite well [32].

6.3.2 MCFD Model

Next, we need to model the statistical behavior of the motion residual signal (*e* in equations (3) and (4) above) assuming an AR(1) model for the source signal *y*. Chen and Pang [31][32] show that the ACF of the MCFD for an AR(1) source can be viewed as the convex combination of two uncorrelated zero-mean wide-sense stationary processes, i.e.,

$$E_n = \alpha Y_n + (1-\alpha)Z_n$$

where Y_n is the same as the original source AR(1) process with parameters (ρ, σ^2) and Z_n is an independent white-noise sequence. More formally, the variance-normalized ACF R_e of e in either direction is given by:

$$\mathrm{R_e} = \begin{bmatrix} 1 & A(a)r & A(a)r^2 & .... & A(a)r^{N-1} \\ A(a)r & 1 & A(a)r & ... & A(a)r^{N-2} \\ A(a)r^2 & A(a)r & 1 & ... & A(a)r^{N-3} \\ . & & & ... & A(a)r^{N-4} \\ . & & & & \\ A(a)r^{N-2} & A(a)r^{N-3} & ... & 1 & A(a)r \\ A(a)r^{N-1} & A(a)r^{N-2} & ... & A(a)r & 1 \end{bmatrix} = \mathrm{R_{e1}} + \mathrm{R_{e2}} \tag{10}$$

where,

$$\mathrm{R_{e1}} = A(a) \begin{bmatrix} 1 & r & r^2 & r^3 & .... & r^{N-1} \\ r & 1 & r & r^2 & ... & r^{N-2} \\ r^2 & r & 1 & r & ... & r^{N-3} \\ . & & & 1 & ... & r^{N-4} \\ . & & & & & \\ r^{N-2} & r^{N-3} & r^{N-4} & ... & 1 & r \\ r^{N-1} & r^{N-2} & r^{N-3} & ... & r & 1 \end{bmatrix} \tag{11}$$

and

$$R_{e2} = (1 - A(a)) \begin{bmatrix} 1 & 0 & 0 & 0 & .... & 0 \\ 0 & 1 & 0 & 0 & ... & 0 \\ 0 & 0 & 1 & 0 & ... & 0 \\ . & & & 1 & ... & 0 \\ . & & & & & \\ 0 & 0 & 0 & ... & 1 & 0 \\ 0 & 0 & 0 & ... & 0 & 1 \end{bmatrix} \tag{12}$$

To obtain the above expressions, the authors of [31] model errors in motion estimation using a uniform pdf. over *[-a, a]*. The parameter *a* may be regarded as the inaccuracy in motion estimation and is typically 0.5 for motion estimation with single pixel accuracy and 0.25 for half pixel accuracy. *A* is a function of *a* and *r* and is given by:

$$A(a) = \frac{1 - 2\left(\frac{1}{2a \ln r}\right)^2 (r^a - r^{-a})(r^a - 1)}{1 - \left(\frac{r^a - 1}{\ln r}\right)^2} \leq 1 \tag{13}$$

with $0 < a \le x$ or y depending on the direction Figure 6.10 below plots *A(a)* as a function of the inaccuracy (a) in motion estimation. Note that A is a decreasing function of both r and of the accuracy (1-a) in motion estimation.

The ACF of equation (10) has been normalized using the variance $E[R_e(0,0)]$ given by:

$$E[\mathrm{R_e}(0,0)] = 2R(0,0)\left[1-\left(\frac{r^a-1}{\ln r^a}\right)^2\right] \tag{14}$$

Given the above facts, the two dimensional ACF can be reasonably approximated as a separable combination of horizontal and vertical normalized ACF and is given by:

$$E[R_e(x,y)] = E[R_e(0,0)]R_{e,x}R_{e,y} \tag{15}$$

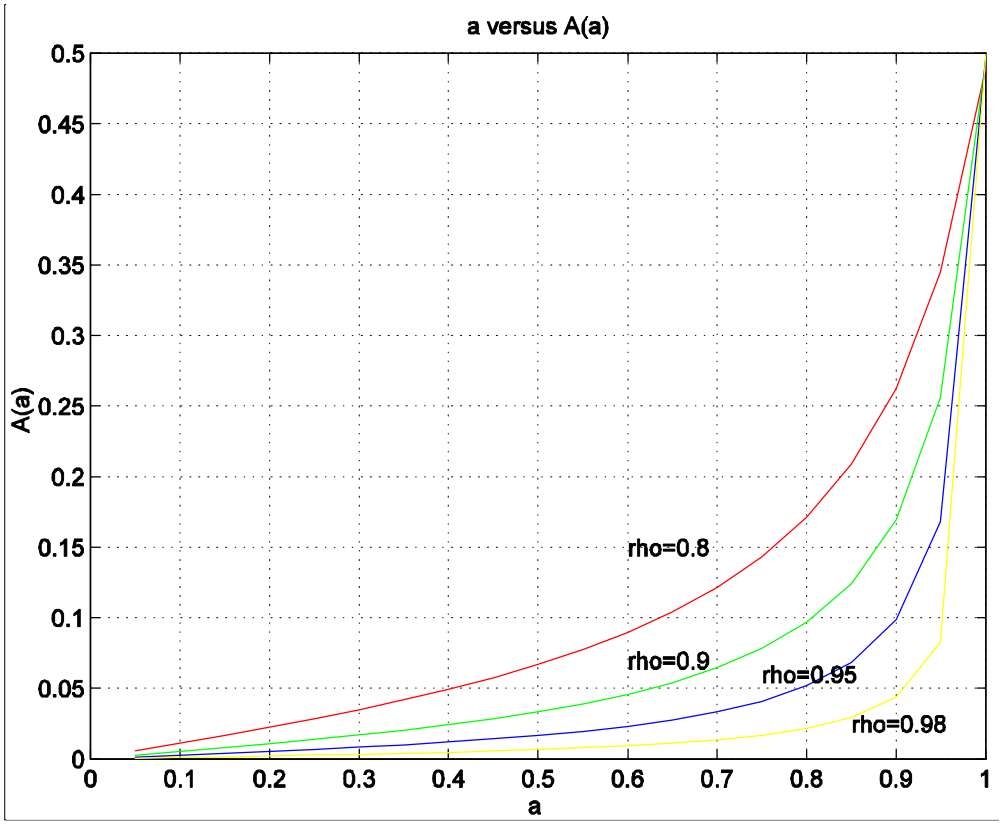


Figure 6.10: ***A(a)*** **as a function of the inaccuracy in motion estimation (a)**

6.3.3 Rate distortion behavior

The next step in finding the optimal DRS solution involves a characterization of the rate distortion behavior of the MCFD signal *e*.

a) Large Block Size

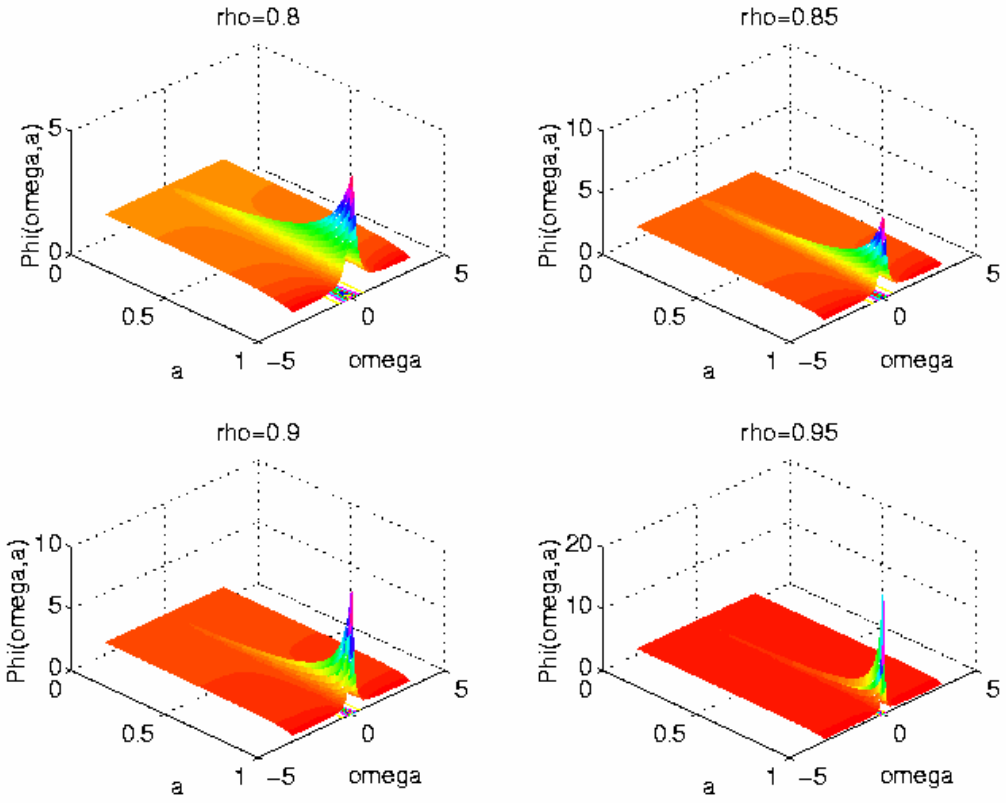
Berger [22] shows that the MSE rate distortion function of a Gaussian source with a Toeplitz ACF that corresponds to a spectral density function $F(w)$ has the following parametric representation for large block sizes:

$$D_q = \frac{1}{2p}\int_{-p}^{p} \min[q, \Phi(w)]dw \text{ and}$$
$$R(D_q) = \frac{1}{4p}\int_{-p}^{p} \max[0, \log\frac{\Phi(w)}{q}]dw \quad \text{where } \Phi(w) = \sum_{k=-\infty}^{\infty} \mathrm{R_e}(k)e^{-jkw} \tag{16}$$

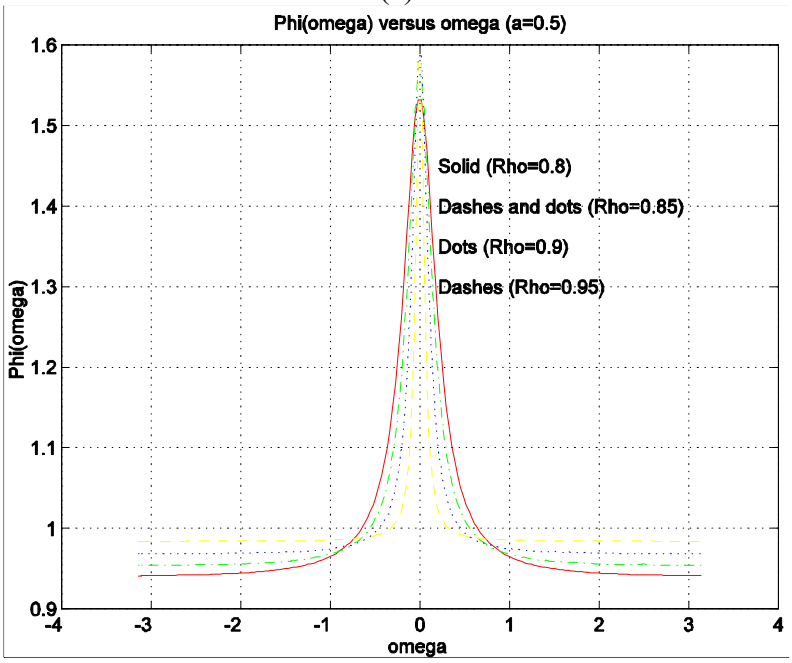
The *R(D)* curve can be generated using the parameter $q = -\frac{1}{2}\frac{\partial R}{\partial D}$. Using the above expression for normalized ACF in a single direction yields:

$$\Phi(w) = 1 + 2A(a)\mathrm{Real}\{\sum_{k=0}^{\infty}(re^{-jw})^k - 1\} = 1 + 2rA(a)\left[\frac{\cos(w) - r}{1 + r^2 - 2r\cos(w)}\right] \tag{17}$$

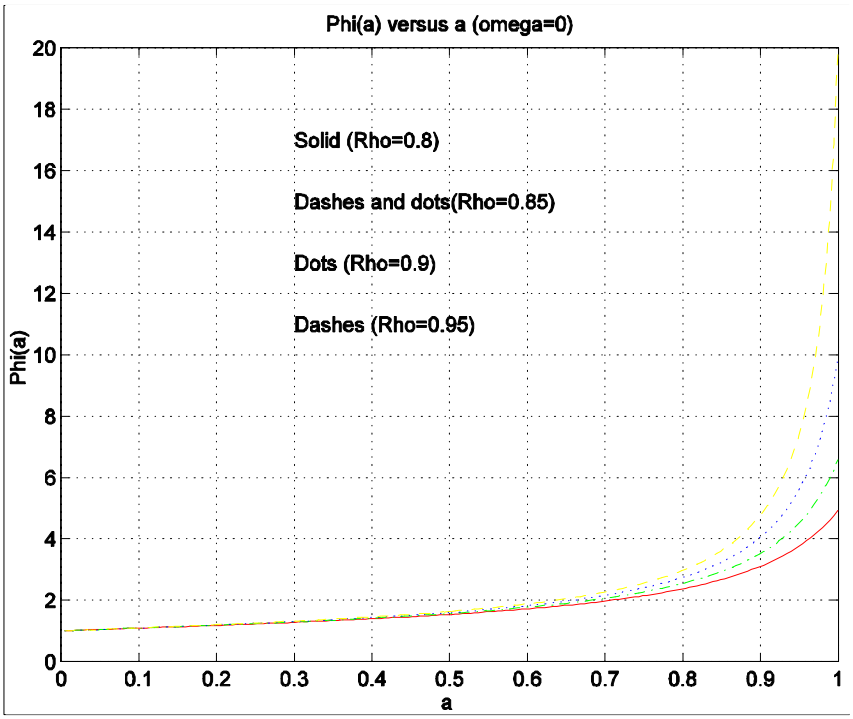
Figure 6.11 plots the spectral density function for typical values of r. As can be seen from Figure 6.11 (c) plots, as the inaccuracy in motion estimation increases, the spectral density for a given source correlation coefficient increases. For a given (constant) inaccuracy in motion estimation, as the source correlation increases, the spectral density at any frequency increases. From Figure 6.11 (b) we note that as the source correlation coefficient, lower frequencies have higher spectral density. The spectral density gets concentrated around lower frequencies as the source correlation increases.


rho=0.8
rho=0.85
rho=0.9
rho=0.95
Phi(omega,a)
a
omega

(a)


Phi(omega) versus omega (a=0.5)
Solid (Rho=0.8)
Dashes and dots (Rho=0.85)
Dots (Rho=0.9)
Dashes (Rho=0.95)
Phi(omega)
omega

(b)



(c)

Figure 6.11: Spectral density function (a) Spectral density function as a function of the frequency (omega) and the inaccuracy in motion estimation (a) (b) The spectral density function as a function of omega (c) The spectral density function as a function of the inaccuracy in motion estimation (a).

Small Distortion:

For small q or equivalently small distortion, $D_q=q$ and $R(D_q)=R(q)=R(D)$ where:

$$R(D) = \frac{1}{2p}\int_0^p \log\left[\frac{1}{D}+\frac{2A(a)\,r}{D}\left\{\frac{\cos w - r}{1+r^2-2r\cos w}\right\}\right]dw$$
$$= \frac{1}{2p}\int_0^p [\log(a_1+a_2\cos w)-\log(a_3+a_4\cos w)]dw \qquad (18)$$

with

$$a_1 = 1+r^2(1-2A(a)),\ a_2 = 2r(A(a)-1),\ a_3 = D(1+r^2),\ a_4 = -2Dr \qquad (19)$$

While we omit final expressions, we observe that this may be solved for explicitly using the fact that [152],

$$\int_0^p \ln(a \pm b\cos w)dw = p\,\ln\left[\frac{a+\sqrt{a^2-b^2}}{2}\right] \text{ for } a \geq b$$

Note that $a_1 - a_2 = (1+r)(1+r-2A(a)r) > 0$ if $A(a) < (1+r)/2r$, which is true since $r < 1$. Further, $a_3 - a_4 = D(1+r)^2 > 0$. Figure 6.12 plots the rate distortion function for small distortion for typical values of r and $A(a)$. Note that the rate decreases as the distortion increases and that as the correlation coefficient (r) increases, the rate needed to achieve a given distortion in the MCFD signal decreases.

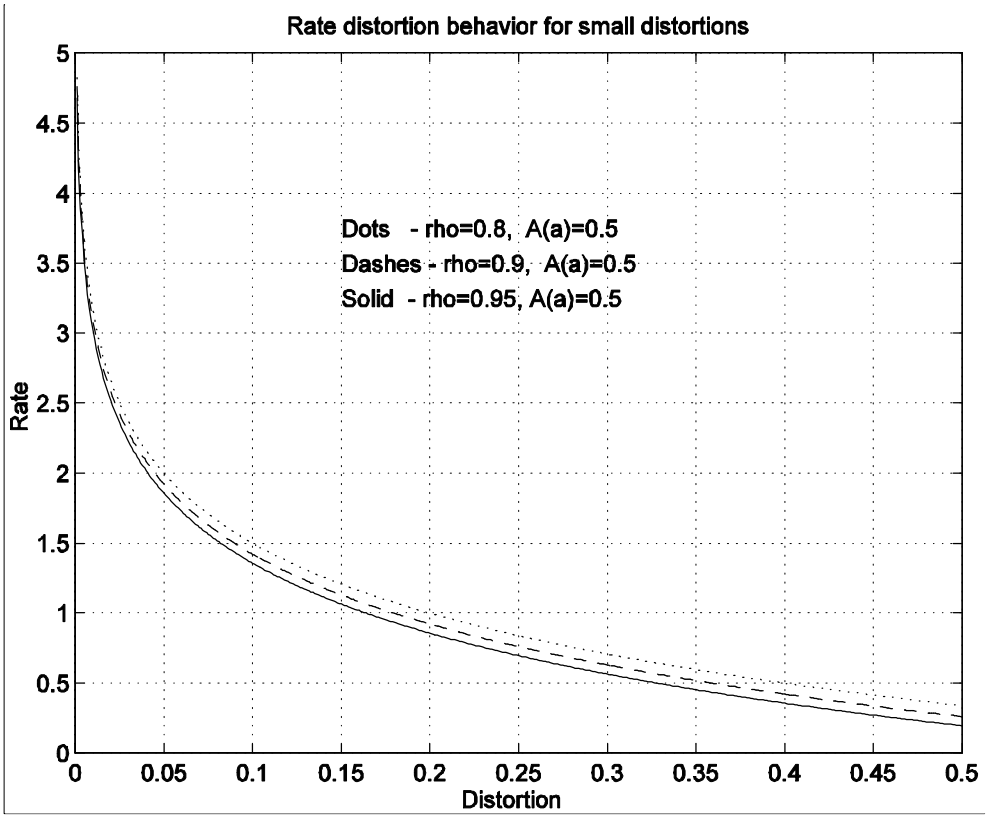


Figure 6.12: Rate distortion function for small distortion.

Figure 6.13 shows three dimensional plots of the rate required to encode the MCFD signal at a given distortion as a function of both *a* and r. For a given source correlation coefficient r, as the inaccuracy *a* in motion estimation increases, *A(a)* increases. Consequently, the white noise term contributing to the ACF of the MCFD signal becomes less dominant. The statistical behavior of

the MCFD signal starts to get closer to the original AR(1) source model and it becomes easier to encode it – thus resulting in small rate requirements. As was observed from Figure 6.12 earlier, we again see that as the source correlation coefficient increases, the rate required for encoding the MCFD signal at a given distortion decreases. One would certainly expect such behavior in the source domain. The fact that it is even true after motion compensation is suggestive of the fact that strong statistical properties carry over from the source signal to the MCFD signal. Finally, we observe that the rate requirement is a strong function of the source correlation coefficient and its dependency on the inaccuracy in motion estimation *(a)* is rather weak.

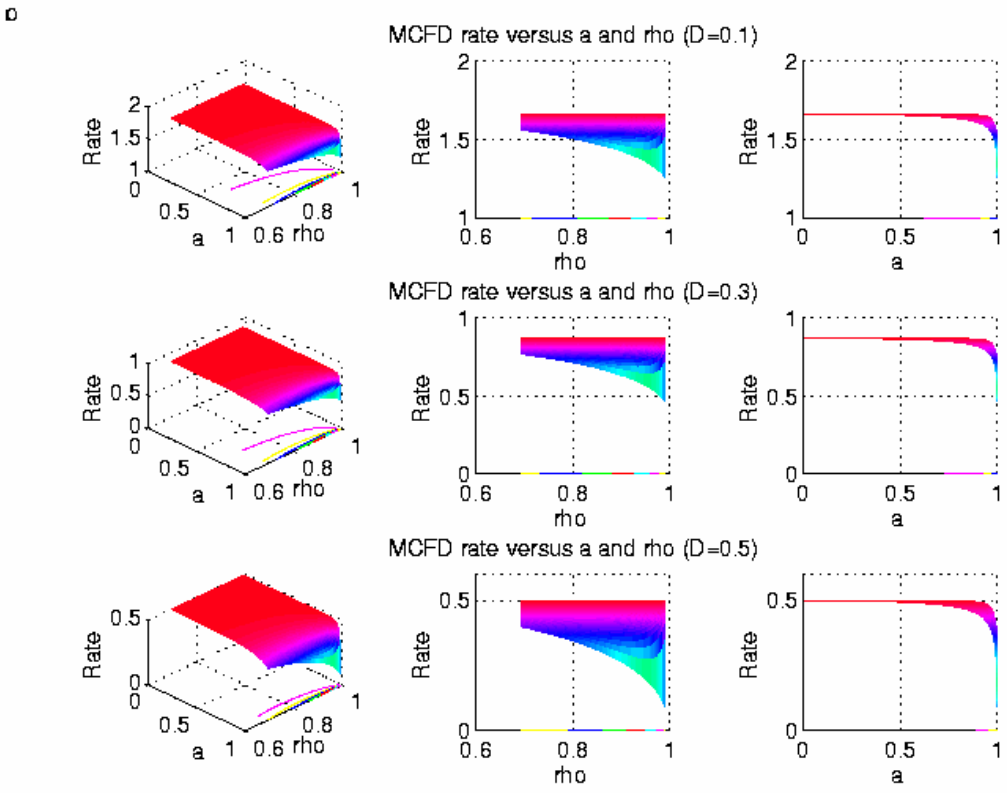


Figure 6.13: Rate (for a given distortion) of the MCFD signal as a function of the inaccuracy in motion estimation (a) and the source correlation coefficient.

Large Distortion:

Consider next large values of D or q. Note that $F(w)$ is even and is an increasing function for $\omega \in [-\pi,0]$ and a decreasing one for $\omega \in [0,\pi]$. Let $w_q>0$ be such that $F(w_q)=q$, then

$$\begin{aligned} D_q &= 1/p \int_0^p \min[q, \Phi(w)]dw \\ &= \frac{q w_q}{p} + \frac{(p - w_q)(1 - A(a))}{p} + \frac{(r^2 - 1)A(a)}{p} \int_p^{w_q} \left[\frac{1}{1 + r^2 - 2r\cos w} \right] dw \end{aligned} \tag{20}$$

From [152], we have that for $b^2 \geq c^2$,

$$\int \frac{dw}{b + c\cos w} = \frac{2}{\sqrt{b^2 - c^2}} \tan^{-1}\left[\frac{(b-c)\tan(w/2)}{\sqrt{b^2 - c^2}} \right] \tag{21}$$

Thus,

$$D_q = \frac{q w_q}{p} + \frac{(p - w_q)(1 - A(a))}{p} - \frac{2A(a)}{p} \tan^{-1}\left[\frac{(1+r)}{(1-r)} \tan(w_q/2) \right] \tag{22}$$

Unlike for small D, $R(D)$ must be left in a "non-explicit" parameterized form now:

$$R(D_q) = 1/2p \int_0^{w_q} \log\left[\frac{1}{q} + \frac{2r\,A(a)(\cos w - r)}{q(1 + r^2 - 2r\cos w)} \right] dw \tag{23}$$

Figure 6.14 plots the rate distortion function for large values of distortion for typical values of r and $A(a)$. Note that again as the correlation coefficient (r) increases, the rate needed to achieve a given distortion in the MCFD domain decreases.

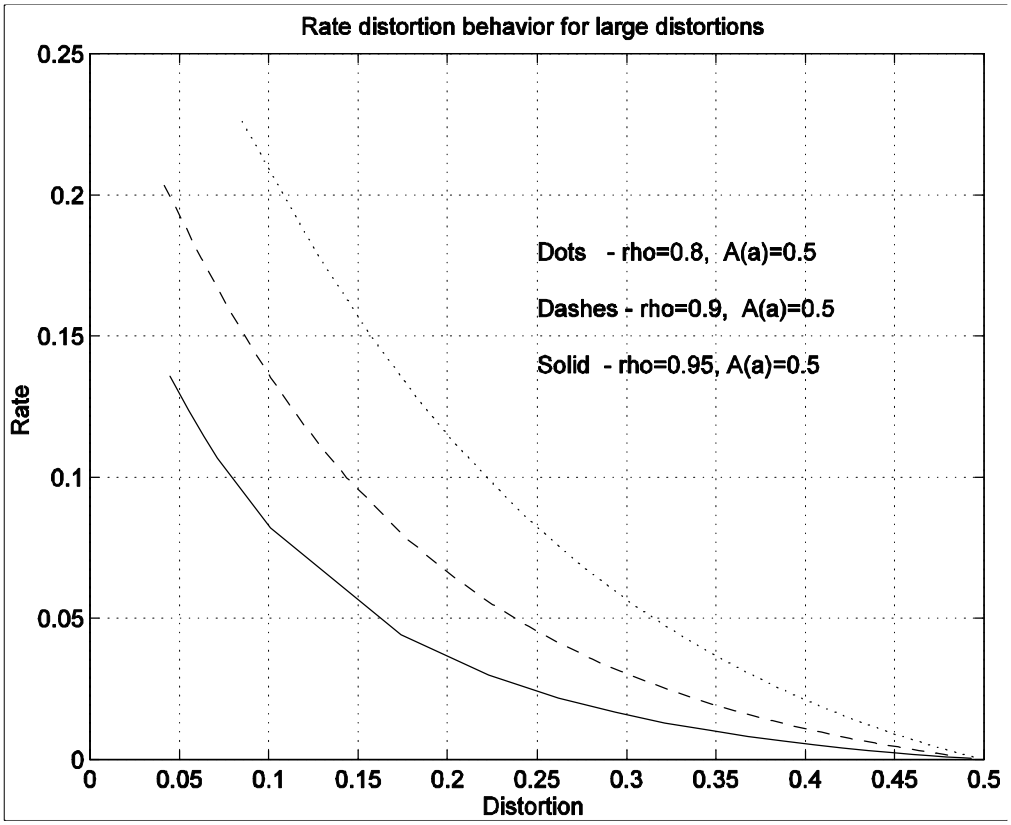


Figure 6.14: Rate distortion function for large distortions.

b) Finite Block Size

For finite block sizes, the parametric rate-distortion function [22] is given by:

$$D_q = \frac{1}{N}\sum_{k=0}^{N-1}\min(q, l_k) \text{ and } R_q = \frac{1}{N}\sum_{k=0}^{N-1}\max[0, \frac{1}{2}\log(l_k / q)] \tag{24}$$

Here l_k are the MC-DCT coefficient variances. Note also the following special cases:

$$\begin{aligned} &\text{If } q \le \min(l_k),\ D_q = q \text{ and } R(D) = \frac{1}{2N}\log\left[\prod_{k=0}^{N-1}\frac{l_k}{D}\right] \\ &\text{If } q = \max(l_k),\ R(D_q) = 0 \text{ and } D_q = \frac{1}{N}\sum_{k=0}^{N-1} l_k \end{aligned} \tag{25}$$

6.3.4 Current frame shaping error

Until now, we have modeled the MCFD signal *e* and its R-D behavior. We now return to the current-frame shaping error $e\text{-}e^*$. Recall that for AR(1) sources with high r, the rate distortion

behavior for coding DCT coefficients is very close to the statistically optimal Karhunen-Louve Transform (KLT) (see Chapter 6 in [132]). We therefore take (the somewhat round-about) recourse to the KLT to approximate DCT. It is known that the orthogonal basis functions for KLT are the eigenvectors of the signal's ACF matrix. Let the eigenvectors E_i and eigen-values l_i solve the eigen-problem for R_e, i.e.,

$$[E]^{-1}\,\mathrm{R_e}[E] = diag[\,l_0, l_1, ..., l_{N-1}] \text{ with } [E] = [E_0, E_1, ..., E_{N-1}] \tag{26}$$

If C_i is the projection of the residual signal along E_i, the shaping error in terms of KLT is given by:

$$\sum_{i=b}^{N} (C_i E_i)^2 = \sum_{i=b}^{N} |\,C_i\,|^2 = \sum_{i=b}^{N} E_i^T e e^T E_i \tag{27}$$

with an average value of:

$$||\,e - e^*\,|| = \sum_{i=b}^{N} E_i^T\,\mathrm{R_e}\,E_i = \sum_{i=b}^{N} l_{\,i} \tag{28}$$

Using the above equations and the fact that e is jointly Gaussian, we conclude that e-e^* is also Gaussian and can be represented in a statistically optimal manner in terms of $(E_i,\ l_i)|_{i=b}$. A further simplifying fact [32] is that since,

$$[E]^{-1} R_e [E] = [E]^{-1}(R_{e1} + R_{e2})[E] = [E]^{-1} R_{e1}[E] + R_{e2} \tag{29}$$

the optimal eigenvectors for e are in fact the same as those of the original AR(1) source signal. Since the eigenvectors and eigenvalues for the AR(1) source may be computed using transcendental equations (see equations 3.2.15 and 3.2.16 in [132]), one may back-substitute the eigenvectors thus obtained into (29) to obtain the eigenvalues for R_e.

6.3.5 Accumulation error

The only portion in equations (3) and (4) that we have not yet characterized is the accumulation error term, $a_i = M_i(y_{i-1}) - M_i(y^*_{i-1})$. If we assume that motion compensation is done via pixel translation, ignore the errors due to integer truncation, and assume that motion vectors obtained with y and y^* are the same and identical to those for y-y^*, we can write $a_i \approx M_i(y_{i-1} - y^*_{i-1})$. The motion compensation operator only translates pixel values. Hence, from a statistical point of view, the random process that models $y_{i-1} - y^*_{i-1}$ should model a_i sufficiently well as well.

6.3.6 Properties of the optimal solution

Due to the above arguments, a degenerate case of motion compensation is worthy of special attention. Let M_i be the identity operator corresponding to zero motion vectors. Solving (3) recursively yields that:

$$y_i - y_i^* = y_{i-1} - y_{i-1}^* + e_i - e_i^* = \{ \sum_{j=0,..,i-1} (e_j - e_j^*) \} + e_i - e_i^* \quad (30)$$

Here the first parenthesized term is the accumulation error while the second term is the current frame shaping error. If $Z=X+Y$ and X and Y are un-correlated, it is easy to see that $R_{Z,Z}=R_{X,X}+R_{Y,Y}$. Thus, if the current frame shaping error signal is temporally uncorrelated and identically distributed, we get that the accumulation error term affects the ACF by a scalar multiple. Consequently, using R-D expressions for large blocks given before, we see that the R(D) function gets perturbed by an additive constant. Therefore, under the above assumptions, the set of optimal breakpoint values would not get affected if we were to exclude the accumulation error term altogether to start with.

One may observe a similar result, though with much weaker assumptions, by using the following fact from matrix perturbation theory [159]:

Lemma: If A and $A+E$ are n-by-n symmetric matrices, $l_k(X)$ is the k^{th} largest eigenvalue of X, then for $k=1,..,n$:

$$\left| l_k(A+E) - l_k(A) \right| \le \left\| E \right\|_2 \le \sqrt{n} \left\| E \right\|_1 = \sqrt{n} \max_{j=1,..,n} \sum_{i=1}^{n} \left| E_{i,j} \right| \quad (31)$$

Using this with $A=A(a)Y_{e1}=R_{e1}$ and $E=R_{e2}$, yields:

$$\left| l_k(A+E) - l_k(A) \right| \le (1 - A(a))\sqrt{n}, \text{ or} \quad (32)$$

$$A(a)\, l_k(Y_{e1}) - (1 - A(a))\sqrt{n} \le l_k(R_e) \le A(a)\, l_k(Y_{e1}) + (1 - A(a))\sqrt{n} \quad (33)$$

Since $A(a) \pounds 1$, we see that all the eigenvalues for R_e and Y_{e1} are a within constant bound of each other. Thus, using (33), we see that the $R(D)$ function gets perturbed by at most a constant amount if we were to drop the $M_i(y_{i-1})$ term (from $M_i(y_{i-1})+e_i$) and computed the $R(D)$ function based only on e_i. Using exactly similar arguments on truncated sub-spaces of y^* and e^* lets us

drop the $M_i(y^*_{i-1})$ term. Again, we are led to the fact that the optimal break-point values don't change (much) by excluding the accumulation error term.

Yet another perturbation inequality [159] leads to even tighter bounds.

Lemma: Let A and $A+E$ be n-by-n symmetric matrices, and let $l_k(X)$ be the k[th] largest eigenvalue of X, then:

$$l_k(A)+l_n(E)\le l_k(A+E)\le l_k(A)+l_1(E) \text{ for } k=1,..,n. \quad (34)$$

Using this with $A=R_{e1}\equiv A(a)Y_{e1}$ and $E=R_{e2}$ yields:

$$A(a)\,l_k(Y_{e1})+(1-A(a))\le l_k(R_e)\le A(a)\,l_k(Y_{e1})+(1-A(a)) \text{ for } k=1,..,n \quad (35)$$

where we use that $l_1(E)=l_n(E)=1-A(a)$. This further implies that:

$$l_k(R_e)=A(a)\,l_k(Y_{e1})+1-A(a) \text{ for } k=1,..,n. \quad (36)$$

Note that statistically $M_i(Y_{i-1})$ behaves like Y_i. Thus, we consider the above lemma with $A=(1+A(a))Y_{e1}$ and $E=R_{e2}$, yielding:

$$(1+A(a))\,l_k(Y_{e1})+(1-A(a))\le l_k(Y_{e1}+R_e)\le(1+A(a))\,l_k(Y_{e1})+(1-A(a)) \quad (37)$$

for $k=1,..,n$, which further implies that,

$$l_k(Y_{e1}+R_e)=(1+A(a))\,l_k(Y_{e1})+1-A(a) \text{ for } k=1,..,n. \quad (38)$$

and that,

$$l_k(Y_{e1}+R_e)=l_k(Y_{e1})+l_k(R_e) \quad (39)$$

or using (36),

$$l_k(Y_{e1}+R_e)=\left[1+\frac{1}{A(a)}\right]l_k(R_e)-\left[1-\frac{1}{A(a)}\right] \quad (40)$$

Equation (40) shows that dropping the accumulation error term would (due to the log term) affect the Rate Distortion function by an additive constant only, leading us to the conclusion that the set of optimal breakpoints would stay (more or less) the same after dropping this term.

6.4 Summary of the chapter

The concept of Dynamic Rate Shaping was discussed as an adaptation mechanism between coded video rate characteristics and transport service capabilities, and analyzed in an operational rate-distortion context. We build on earlier experimental results that show that a memoryless version of the algorithm performs almost identical to the causally optimal one, hence significantly simplifying the implementation complexity, with no compromise in terms of quality. This is so even with the memoryless version of GDRS algorithm. The desire to better understand the good performance of the memoryless problem led us to analyze the behavior of the optimal solution to the DRS problem assuming an AR(1) source model. By deriving the statistical and rate-distortion characteristics of different components of the inter-frame problem we showed that the set of optimal breakpoint values for any frame is somewhat invariant to the accumulated motion compensated shaping error from past frames and may be very reasonably approximated using the current frame shaping error alone. This result is significant because it opens up the way to construct simpler memoryless algorithms that give minimal penalty in achieved picture quality, not just for this, but possibly other types of problems such as those in [3] and [105].

CHAPTER 7

Conclusions and future work

Contents

In this Thesis, we explored problems that arise at various stages of an audio-visual communication system. We addressed (a) the optimal intra and inter frame quantization problem, (b) the modeling of object based systems both at the server and the client side, (c) the content based approach for resource allocation and error control, and (d) the structure of the optimal dynamic rate shaping problem for Markovian sources. We took the approach of abstracting the minimal essentials that help to model the problem in a mathematically tractable form, yet retain a high correlation with reality. We also introduced the video coding community to several new mathematical tools that may aid in solving a wide array of problems. While the treatment presented above is fairly complete, it leaves ample room for improvement and extensions.

7.1 Approximately optimal bit allocation

An interesting outcome of the MCKP formulation of Chapter 2 is that some known approximation algorithms [30][72][138] can be used to solve bit allocation problems. We believe that these have not been studied well enough for applications in compression and may have enormous impact on it. An ε-Polynomial Time Approximation Scheme (PTAS), although a theoretical nicety for the most part, gives a smooth tradeoff between an algorithm's complexity and its performance. Using a PTAS, one can solve the bit allocation problem to an arbitrary degree of precision with a proportionate increase in computational complexity (and hence, delay) at an encoder. The user may pre-specify the encoding complexity that he is willing to incur; based on which the algorithm scales to satisfy an error-ratio relative to the optimal solution, irrespective of source statistics or the problem instance. Alternatively put, based on an a-priori error bound from the optimal solution, the scheme would scale gracefully from very low to exponentially high complexity. Hence one may view a PTAS as a set of algorithms to tradeoff encoding complexity versus distortion while quantizing in an operational R-D. We reiterate an important outcome of using approximation algorithms is that their performance is guaranteed irrespective of the problem instance or the source density being quantized. Of equal, if not more practical interest, is a known 4/5-approximation algorithm that has fairly minimal complexity. Each of these algorithms yields a *lossy universal* compression (quantization) scheme. We will discuss these results and its implications on quantization theory in more detail in a follow-up paper [13].

7.2 Scheduling objects

The early-tardy model of Chapter 3 could be expanded in several directions. A functionality that our model ignores (yet one that is commonly supported in multimedia documents) is user interaction. Users may change the playout of a presentation by interacting with the scene. This could happen by stopping or starting some elementary streams, by adding locally stored objects, or by signaling a remote server to modify the scene. Similar behavior can also be caused by quasi-static objects including objects with hyperlinks, D-HTML documents with portions that may be activated or de-activated interactively, and objects having local or distributed dependent

objects. Events caused by these occur at a coarse time granularity, are asynchronous, and fairly unpredictable. They cause the temporal requirements of some objects to change at render time. In order to facilitate the use of an ET-like framework to handle this, one avenue could be to extend incremental approaches like the dispatching rules. Another direction may be to build some fault tolerance within the model to allow a cushion for unpredictable variations.

Yet another functionality that is becoming commonplace is the integration and coherent playback of objects present in distributed and federated network environments. New issues like heterogeneity, type and location of synchronization control, hierarchical synchronization frameworks, and load balancing have to be taken into account in distributed systems. The incorporation of some of the above into the ET model could form an interesting direction to work in. Finally, as we have mentioned earlier also, our framework handles buffer limitations implicitly using an earliness penalization term instead of using explicit buffering constraints. Other works such as the JINSIL project [150], Escobar-Molano [51] and Kalva and Eleftheriadis [86][87] took an explicit account of these to find an optimal resource schedule for continuous display of structured video. Their primary focus in these works had been on buffer and bandwidth allocation issues resulting from variations in number of atomic objects and their compression ratios. Similar results in [101][139] also report significant improvement in network utilization by doing work-ahead smoothing and optimal pre-fetching under decoder buffering and delay constraints. We believe that it would be an interesting exercise to study if ET problems with additional buffering constraints can be optimally yet easily solved.

7.3 Memoryless algorithms and dependent allocation

In the last two chapters of this Thesis we studied the affects of dependency in quantization and dynamic rate shaping problems, respectively. We saw experimentally that the improvement obtained by relaxing the causality assumption was fairly marginal for the dependent quantization problem. The author believes that a rigorous approach like that of Chapter 6 could be used to explain this behavior as well. Finally, the key result of Chapter 6 opens up the way to construct much simpler memoryless (approximate) algorithms for traditional drift problems such as those in [3] and [105].

APPENDICES

Contents

A.1 Branch and bound method [110]

The Branch and Bound method is a recursive procedure for solving any pure or mixed integer-programming problem. Without loss of generality we consider a max IP. As we show in Figure 8.1, the Branch and Bound method solves the problem by repetitively solving linear relaxation of integer programs. The optimal solution to a linear relaxed problem forms an upper bound on all feasible solutions to the integer program. The procedure consists of two rules – a Branching Rule and a Fathoming or Stopping Rule. In the branching step, two sub-problems are solved each preventing a fractional variable (X_i in Figure 8.1) from taking on its fractional value $(I+f)$ in the optimal solution to the LP. Note that children of a node can only lead to worse solutions than the parent. Thus, branching need not be done if the linear program gives a solution with integral variables (where necessary) or a value of the LP relaxed objective function that is less than the current best integer solution. A formal statement of the method is:

Branching Rule:

STEP 1. Solve the corresponding linear programming problem by ignoring any integer constraints, for example, if we require that $X_i \in \{0,1\}$, we would put the constraint $0 \leq X_i \leq 1$ for i=1,..,n in the LP relaxation. The linear program may be solved in one of many ways, e.g., using the simplex method or a faster interior point method.

STEP 2. Choose any variable, say X_i, which takes on a fractional value (say, I + f) in the solution to Step 1. If for the original IP X_i must take only integral values only, we generate two sub-problems specified below, and repeat the Branching rule for each of them.

Sub-problem 1: Linear programming problem in **STEP 1** + constraint $X_i \leq I$

Sub-problem 2: Linear programming problem in **STEP 1** + constraint $X_i \geq I+1$

Fathoming or Bounding Rule:

A node in the tree is fathomed if either of the following is true:

RULE 1. The sub-problem is infeasible.

RULE 2. The solution to the linear relaxation has an integer-valued solution for the necessary variables. Note that this solution is feasible for the original IP. If the value of the objective function is higher than the current best lower bound, the current lower bound is set to this value.

RULE 3. The value of the objective function is less than the current lower bound. Note that the variables need not be integers to do this fathoming step.

Although Branch-and-Bound is a generic procedure, its complexity is exponential. In case of KP**,** solving the linear program involves only a sorting step. Once the candidate objects are sorted, they are added in decreasing order of profit density. When the bandwidth constraint cannot be satisfied, a fractional amount of the *critical object* is added using up the entire residual bandwidth. Since there is only one fractional object, the two sub-problems also get simplified to the original problem plus a constraint for including or excluding the critical object. This leads to two smaller problems with exactly the same structure as the original one. Using an iterative

argument one gets an $O(2^n)$ complexity for Branch and Bound for KP. Using similar arguments, we can show that the complexity of this method for MCKP is $O(2^{\sum_i n_i})$.

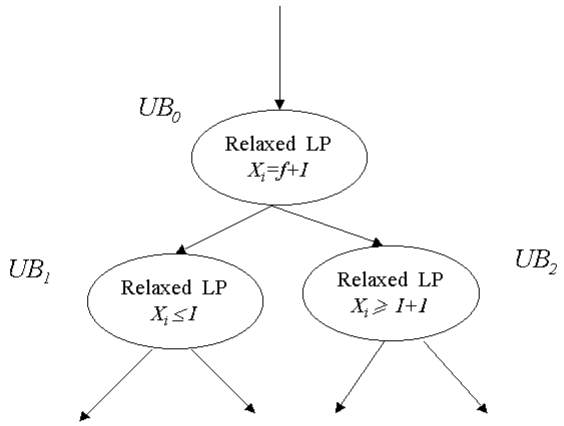


Figure 8.1: The Branch and Bound Method.

A.2 Dynamic programming for MCKP [110][148]

As with the simple knapsack problem, dynamic programming gives a pseudo-polynomial time algorithm for MCKP. We use the following notation:

$f_k(m)$ = maximum objective function value if m units of bandwidth are available and we are allowed to use only items in *sets 1,..,k.*

m = amount of currently available bandwidth, $m \leq B$

$s_i^* = \min_{j=1,..,n_i} \{s_{ij}\}$ for $i=1,..,N$

Using the above, a state is defined by the pair (k, m) and has a value

$$f_k(m)=max\{ \sum_{i=1}^{k}\sum_{j=1}^{n_i} p_{ij}X_{ij} \mid \sum_{i=1}^{k}\sum_{j=1}^{n_i} s_{ij}X_{ij} \leq m; \sum_{j=1}^{n_i} X_{ij} = 1 \text{ for each } i \leq k \text{ and } X_{ij} \in \{0,1\} \}$$

With this definition of state, we can write the following DP recursion:

$f_1(m) = -\infty$ for $m=0,..,s_1^*-1$.

$= \max_{j=1,..,n_1} \{ p_{1j} \mid s_{1j} \leq m \}$ for $m= s_1^*,..,B$.

and for $k=2,..,N$,

$f_k(m) = -\infty$ for $m = 0,.., \sum_{i=1}^{k} s_i^* - 1$

$= \max_{j=1,..,n_k} \{ f_{k-1}(m-s_{kj}) + p_{kj} \mid s_{kj} \leq m \}$

for $m = \sum_{i=1}^{k} s_i^* ,..., B$

The optimal objective function is given by $f_N(B)$ and the optimal set $\{X_{ij}\}$ can be found by backtracking the algorithm. Obviously, the space and time complexity of this method is $O(B\sum_{i=1}^{N} n_i)$. As this may become large fast, DP is not suitable for solving big problems. We may, however, prune out some states from each set to reduce complexity. For example, if there are two scaled versions O_1 and O_2 of an object such that O_1 has higher bitrate and lower PSNR than O_2, then O_1 will never be included in an optimal solution and hence can be pruned out.

A.3 Proof of propositions in Chapter 3 [119]

Consider proposition (b) in Section 3.4 of Chapter 3. We need to show that target display time of objects coincides with the completion time of some object. Assume otherwise. This would mean that we have a situation like the one shown in Figure 8.2. Let the objective value of this schedule be *f(d)*. Form a new schedule by shifting objects in set P_1 to the right by d_1. Call the new objective function value $f(d_1)$. It is easy to see that :

$$f(d_1) = f(d) + (\sum_{j \in P_2} b_j + b_i - \sum_{j \in P_1} a_j - a_i) d_1$$

Similarly, form a new schedule from the original one by shifting objects in set P_2 to the left by d_2. Call the new objective function value $f(d_2)$. Then,

$$f(d_2) = f(d) - (\sum_{j \in P_2} b_j + b_i - \sum_{j \in P_1} a_j - a_i) d_2$$

Since both d_1 and d_2 are greater than zero, one of the two new schedules must be better than the original one depending on whether $\sum_{j \in P_2} b_j + b_i - \sum_{j \in P_1} a_j - a_i$ is less than or greater than 0.

Other propositions in Section 3.4 can be proven in a similar way.

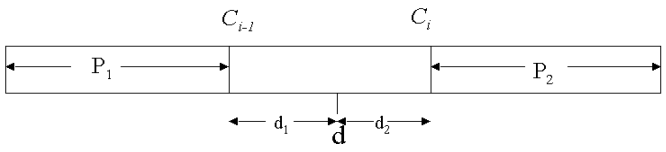


Figure 8.2: The target display time should coincide with the completion time of some object.

REFERENCES

[1] E. Amir. and S. McCanne and M. Vetterli, "A Layered DCT Coder for Internet Video", Proc. ICIP, Laussanne, Switzerland, Vol. 1, pp. 13-16, Sep. 1996.

[2] R. Aravind. and R. Civanlar, R. and A.R. Reibman, "Packet Loss Resilience of MPEG-2 Scalable Video Coding Algorithms", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 6, No.5, pp. 426-435, Oct. 1996.

[3] J.F. Arnold, M.R. Frater, Y. Wang, "Efficient Drift-Free Signal-to-Noise Ratio Scalability," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 10, No. 1, pp. 70-82, Feb, 2000.

[4] P. Assuncao, and M. Ghanbari, "A frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bit streams," *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 8, No. 8, 1998, pp. 953–967.

[5] P. Assuncao and M. Ghanbari, "Optimal transcoding of compressed video," *Proceedings, IEEE Int'l Conf. on Image Processing*, October 1997, Santa Barbara, CA, Vol. 1, pp. 739–742.

[6] E. Ayanoglu etal., "Forward error control for MPEG-2 video transport in a wireless LAN", ACM/Baltzer Mobile Networks and Applications Journal, Vol. 1, pp. 235-244, Dec. 1996.

[7] O. Avaro, A. Eleftheriadis, C. Herpel, G. Rajan, and L. Ward, "MPEG-4 Systems: Overview", pp. 331-365, Multimedia Systems, Standards and Networks, Editors A. Puri, T. Chen, Marcel Dekker, New York.

[8] K. Baker and G.D. Scudder, "Sequencing with earliness and tardiness penalties: A review," Operations Research, Vol. 38, pp. 22-36, 1990.

[9] K. Baker and G.D. Scudder, "On the assignment of optimal due date", Journal of Operations Research Society, Vol. 40, No. 1, pp. 93-95, 1989.

[10] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz, "Improving TCP/IP Performance over Wireless Networks" Proceedings 1st ACM International Conference On Mobile Computing and Networking, pp. 1-11, Nov. 1995.

[11] P.Batra and A.Eleftheriadis, "A framework for optimal scheduling of structured and streaming media", Columbia University, EE Dept./ADVENT Technical Report #2000-03. To be submitted, IEEE Transactions on Multimedia, 2003.

[12] P.Batra and S-F. Chang, "Effective algorithms for video transmission over wireless channels", Signal Processing: Image Communication, Vol. 12, pp. 147-166, 1998. An earlier, but quite different version is in: CU/CTR Technical Report 476-97-10, 1997.

[13] P. Batra, "Complexity-Distortion tradeoffs in quantization", In Preparation, October 2002.

[14] P. Batra and A. Eleftheriadis, "Analysis of dynamic rate shaping of Markov-1 sources", IEEE Int'l Conf. on Image Processing, Sep 2003.

[15] P. Batra, "Modeling and efficient optimization for object based scalability and some related problems," IEEE Transactions on Image Processing, Vol. 9, No. 10, pp.1677-92, Oct. 2000. An earlier version is in: EE Dept./ADVENT Technical Report #1998-05.

[16] P. Batra, A.Eleftheriadis, and J. Sethuraman, "Alternative formulations for bit allocation with dependent quantization", In review with IEEE Transactions on Image Processing, 2002.

[17] P. Batra and A. Eleftheriadis, "Alternative formulations for bit allocation with dependent quantization", Proc. ICASSP, 2002.

[18] M. Bazaraa, J. J. Jarvis, and H. D. Sherali, "Linear Programming and Network Flows", 2nd Edition, John Wiley and Sons, 1990.

[19] J. Bean, "Genetics and Random Keys for Sequencing and Optimization," ORSA Journal of Computing, Vol. 6, pp. 154-160, 1994.

[20] R. Bellman and S. Dreyfus, "Applied Dynamic Programming", Princeton University Press, Princeton, N. J., 1962.

[21] J. Belzer, Liao, Villasenor, "Adaptive Video Coding for Mobile Wireless Networks", Proc. ICIP, Oct. 1994.

[22] T. Berger, “Rate Distortion Theory: A Mathematical Basis for Data Compression,” Englewood Cliffs, NJ: Prentice Hall, 1971.

[23] E. Berklekamp, R Peile, and S.P. Pope, “The Application of Error Control to Communications", IEEE Communications Magazine, Vol. 25, No. 4, pp. 44-57, Apr. 1987.

[24] D. Bertsimas and R Vohra, “Rounding algorithms for covering problems”, Mathematical Programming, Vol. 80, pp. 63-89, 1998.

[25] H. Burton and D.D. Sullivan, "Error Control Coding", Proceedings of the IEEE, Vol. 60, No. 11, pp. 1293-1301, Nov. 1972.

[26] M.C. Buchanan and P.T. Zellweger, “Automatically generating consistent schedules for multimedia applications”, Multimedia Systems, Vol. 1, No.2, pp. 55-67, 1993.

[27] J. Cain and D.N. McGregor, “A Recommended Error Control Architecture for ATM Networks with Wireless Links" IEEE Journal on Selected Areas in Communications, Vol. 15, No. 1, pp. 16-28, Jan. 1997.

[28] K. Candan, B. Prabhakaran, and V.S. Subrahmanian, “Collaborative Multimedia Documents: Authoring and Presentation,” International Journal of Intelligent Systems, Vol. 13, 1059-1111, 1998.

[29] N. Chaddha and S. Diggavi, “A Frame-work for Joint Source-Channel Coding of Images over Time-Varying Wireless Channels" Proc. ICIP, Laussanne, Switzerland, Vol. 1, pp. 1-5, Sep. 1996.

[30] A. Chandra, D. S. Hirschberg, and C. K. Wong, “Approximate algorithms for some generalized knapsack problems”, Theoretical Computer Science, Vol. 3, pp. 293-304, 1976.

[31] C.-F. Chen and K.K. Pang, “Hybrid Coders With Motion Compensation,” Multidimensional Systems and Signal Processing, Vol. 3, pp. 241-266, 1992.

[32] C.-F. Chen and K.K. Pang, “The Optimal Transform of Motion-Compensated Frame Difference Images in a Hybrid Coder,” IEEE Transactions on Circuits and Systems – II: Analog and Digital Signal Processing, Vol. 40, No. 6, pp. 393-397, 1993.

[33] T. P.-C. Chen and T. Chen, “Fine-grained rate shaping for video streaming over wireless networks,” unpublished manuscript (submitted to ICASSP 2003), available at `http://amp.ece.cmu.edu/people/Trista`.

[34] J. Chen and D.W. Lin “Optimal bit allocation for coding of video signals over ATM networks,” IEEE Journal on Selected Areas in Communications, pp. 1002-15, Vol. 15, No. 6, Aug. 1996.

[35] T. Cheng and M.C. Gupta, "Survey of scheduling research involving due date determination decisions," European Journal of Operations Research, Vol. 38, pp. 156-166, 1989.

[36] H. Chernoff, “A measure of asymptotic efficiency for tests based on the sum of observations”, Annals of Mathematical Statistics, Vol. 23, pp. 493-509, 1952.

[37] T. Chiang and H-J Lee, “ISO/IEC MPEG98/M3816, Study of FCD text for rate control”, July 1997.

[38] J. Chuang, “Comparison of Two ARQ Protocols in a Rayleigh Fading Channel” IEEE Transactions on Vehicular Technology, Vol. 39, No. 4, pp.367-73, Nov. 1990.

[39] R. Clarke, “Digital Compression of Still Images and Video", Academic Press, London, England, 1995.

[40] G. Cornuejols, M.L. Fisher, and G.L. Nemhauser, “Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms”, Management Science, Vol. 23, No. 8, pp. 789-810, April. 1977.

[41] G. Cornuejols, G.L. Nemhauser, and L.A. Wolsey, “The uncapacitated facility location problem”, In P. Mirchandani and R. Francis, Eds., Discrete Location Theory, John Wiley and Sons, John Wiley and Sons, New York, 1990.

[42] R. Cruz, “A Calculus of Network Delay, Part 1: Network Elements in Isolation", IEEE Transactions on Information Theory, Vol. 37, No. 1, pp. 114-31, Jan. 1991.

[43] J. Davis and J.J. Kanet, "Single machine scheduling with early and tardy completion costs," Naval Research Logistics, Vol. 40, No. 1, pp. 85-101, 1993.

[44] M. Dyer, “An *O(n)* algorithm for the multiple-choice knapsack linear program”, Mathematical Programming, Vol. 29, pp. 57-63, 1984.

[45] A. Eleftheriadis and P. Batra, “Optimal data partitioning of MPEG-2 coded video”, To appear, IEEE Transactions on Circuits and Systems for Video Technology, 2004.

[46] A. Eleftheriadis and P. Batra, “Dynamic rate shaping of compressed digital video”, Being revised for IEEE Transactions on Multimedia, 2003.

[47] A. Eleftheriadis, S. Pejhan and D. Anastassiou, "Algorithms and Performance Evaluation of the Xphone Multimedia Communication System", Proc. ACM Multimedia, pp. 311-20, Aug. 1993.

[48] A. Eleftheriadis and D. Anastassiou, "Meeting arbitrary QoS constraints using Dynamic Rate Shaping of Coded Digital Video," NOSSDAV, pp. 96-106, Durham, New Hampshire, April 1995.

[49] A. Eleftheriadis, "Dynamic Rate Shaping of Compressed Digital Video," Ph.D. Thesis, Department of Electrical Engineering, Columbia University, New York, NY, 1995.

[50] A. Eleftheriadis and D. Anastassiou, "Optimal Data Partitioning of MPEG-2 Coded Video," *Proceedings, 1st IEEE International Conference on Image Processing* (ICIP-94), Austin, Texas, November 1994, pp. I.273-I.277.

[51] M. Escobar-Molano, S. Gandeharizadeh, and D. Ierardi, "An Optimal Resource Scheduler for Continuous Display of Structured Video Objects," IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 3, June 1996.

[52] H. Everitt III, "Generalized Lagrange multiplier method for solving problems of optimum allocation of resources", Operations Research, Vol. 11, pp. 399-417, 1963.

[53] M. Fisher, "The Lagrangian relaxation method for solving integer programming problems," Management Science, Vol. 27, pp. 1-18, Jan. 1981.

[54] G. Forney, "The Viterbi Algorithm", Proc. IEEE, Vol. 61, pp. 268-78, Mar. 1973.

[55] B. Fox and D. M. Landi, "Searching for the multiplier in one-constraint optimization problems", Operations Research, Vol. 18, pp. 253-262, 1970.

[56] B. Fox, "Discrete optimization via marginal analysis", Management Science, Vol. 13, No. 3, pp. 210-216, Nov. 1966.

[57] M. Garrett and M Vetterli, "Joint source/channel coding of statistically multiplexed real-time services on packet networks" IEEE/ACM Transactions on Networking, Vol. 1, No. 1, pp. 71-80, Feb. 1993.

[58] M. Ghanbari and V. Seferidis, "Cell-Loss Concealment in ATM Video Codecs", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 3, No. 3., Jun 1993.

[59] M. Ghanbari, "Postprocessing of Late Cells for Packet Video", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 6. No. 6, pp. 669-78, Dec. 1996.

[60] M. Ghanbari, “Two-Layer Coding of Video Signals for VBR Networks”, IEEE Journal on Selected Areas in Communications, Vol. 7, No. 5, pp. 771-81, Jun. 1989.

[61] H. Gharavi and M. Partovi, "Multilevel Video Coding and Distribution Architectures for Emerging Broadband Digital Networks", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 6, No. 5, pp. 459-69, Oct. 1996.

[62] F. Glover, "Tabu Search: A Tutorial," Interfaces, Vol. 20, No. 4, pp. 74-94, 1990.

[63] M. Goemens and D.P. Williamson, “The primal-dual method for approximation algorithms and its application to network design problems”, In Approximation Algorithms for NP-hard Problems, D. Hochbaum, Ed., PWS, 1997.

[64] H.263 Standard, "Line transmission of non-telephone signals", Jul. 1995.

[65] N. Hall and M.E. Posner, "Earliness-tardiness scheduling problems, I: Weighted deviation of completion times about a common due date," Operations Research, Vol. 39, pp. 836-846, 1991

[66] R. Han and D.G. Messerschmitt, “Asymptotically Reliable Transport of Multimedia/Graphics Over Wireless Channels" Proceedings Multimedia Computing and Networking, San Jose, CA, pp.29-31, Jan. 1996.

[67] L. Hanzo and J. Streit, "Adaptive Low Rate Wireless Videophone Schemes", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 5, No. 4, pp. 305-318, Aug.1998.

[68] T. Hsing etal. "A Real-Time Software Based End-to-End Wireless Visual Communications Simulation Platform", Polytechnic University Symposium on Multimedia Communications and Video Coding, Nov. 1995.

[69] C. Hsu. and A. Ortega and M. Khansari, “Rate control for robust video transmission over wireless channels”, Proc. VCIP, San Jose, Feb. 1997.

[70] C. Hsu, A. Ortega, "Joint Selection of Source and Channel Rate for VBR Video Transmission under ATM Policing Constraints", Vol. 15, No. 6, pp.1016-28, Aug. 1997.

[71] J. Huang and P.M. Schultheis, “Block quantization of correlated Gaussian random variables”, IEEE Transactions on Communication Systems, Vol. 11, No. 3, pp. 289-296, Sep. 1963.

[72] O. Ibarra and C. E. Kim, “Fast approximation algorithms for the knapsack and sum of subset problems”, Journal of the ACM, Vol. 22, pp. 463-468, 1975.

[73] IEEE Journal on Selected Areas in Communications, Special Issue on "Synchronization Issues in Multimedia Communications," Guest Eds. N.D. Georganas, R. Steinmetz, and T. Nakagawa, Jan. 1996.

[74] Information Technology – Generic coding of Moving Pictures and Associated Audio, ITU-T Draft Recommendation H.262, ISO/IEC 13818 Draft International Standard (MPEG-2), 1994.

[75] ISO/IEC JTC1/SC29/WG11 N1808 MPEG97, "Description of Core Experiments on Error Resilience Aspects in MPEG-4 Video", 1997.

[76] ISO/IEC JTC1/SC29/WG11, N2416, "Coding of audio-visual objects: Systems" FCD ISO/IEC 14496-1. Oct. 1998.

[77] ISO/IEC JTC1/SC29/WG11, N2417, "Coding of audio-visual objects: Video" FCD ISO/IEC 14496-2. Oct. 1998.

[78] ISO/IEC 14772-1, "The Virtual Reality Modeling Language", 1997 http://www.vrml.org/Specification/VRML97.

[79] ITU-T Recommendation H.263, "Video coding for low bit rate communication", May 1996.

[80] S. Jacobs and A. Eleftheriadis, "Streaming Video using TCP Flow Control and Dynamic Rate Shaping," *Journal of Visual Communication and Image Representation*, Special Issue on Image Technology for World-Wide-Web Applications, Vol. 9, No. 3, September 1998, pp. 211–222.

[81] A. Jain, "Fundamentals of Digital Image Processing," Prentice Hall, Inc., Englewood Cliffs, N.J., USA, 1995.

[82] K. Jain and V.V. Vazirani, "Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation", To appear, Journal of the ACM, 2001.

[83] W. Jakes Jr., "Microwave Mobile Communications", John Wiley and Sons, 1994.

[84] N.S. Jayant and P. Noll, "Digital Coding of Waveforms – Principles and Applications to Speech and Video," Englewood Cliffs, NJ: Prentice Hall, 1984.

[85] H. Kalva et.al. "Implementing multiplexing, streaming, and server interaction for MPEG-4," IEEE Transactions on CSVT, pp. 1299-311, Vol. 9, No. 8, Dec. 1999.

[86] H. Kalva and A. Eleftheriadis, “Algorithms for Multiplex Scheduling of MPEG-4 Presentations”, In Review with IEEE Trans. On CSVT.

[87] H. Kalva and A. Eleftheriadis, “Scheduling Interactive MPEG-4 Presentations”, In Review with IEEE Trans. On CSVT.

[88] H. Kanakia, P. P. Mishra, and A. Reibman, “An Adaptive Congestion Control Scheme for Real-Time Packet Video Transport,” *Proceedings, ACM SIGCOMM Conference*, September 1993, pp. 20-31.

[89] G. Karlsson and M. Vetterli, “Packet Video and its Integration into the Network Architecture", IEEE Journal on Selected Areas in Communications, Vol. 7, No. 5, pp. 739-51, Jun. 1989.

[90] G. Keesman, R. Hellinghuizen, F. Hoeksema, and G. Heideman, “Transcoding of MPEG bitstreams,” Signal Processing: Image Communication, Vol. 8, No. 6, September 1996, pp. 481–500.

[91] M. Khansari and M. Vetterli, “Layered Transmission of Signals over Power-Constrained Wireless Channels" Proc. ICIP, Washington D.C., pp. 23-26, Oct. 1995.

[92] M. Khansari et.al., “Low Bit-Rate Video Transmission over Fading Channels for Wireless Microcellular Systems", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 6, No. 1, pp. 1-11, Feb. 1996.

[93] M. Kim and J. Song, “Multimedia documents with elastic time,” ACM Multimedia Conference, 1995.

[94] W. Lam and A Reibman, "An error concealment algorithm for images subject to channel errors" IEEE Transactions on Image Processing, Vol. 4, No. 5, pp. 533-542, May 1995.

[95] B. Lamparter and M. Kalfane, “The Implementation of PET”, TR-95-047, International Computer Science Institute, Berkeley, CA. Aug. 1995.

[96] G. Lawton, "Genetic algorithms for schedule optimization," AI Expert, pp. 23-27, May 1992.

[97] D. LeGall, “MPEG: A video compression standard for multimedia applications” Communications of the ACM, 34(4): 46-58, April 1991.

[98] T. Little and A. Ghafoor, “Multimedia Synchronization Protocols for Broadband Integrated Services,” IEEE Journal on Selected Areas in Communications, pp. 1368-1382, Vol. 9, No. 9, 1991.

[99] H. Liu and M. Zarki, “Data partitioning and Unbalanced Error Protection for H.263 Video Transmission over Wireless Channels", Polytechnic University Symposium of Multimedia Communications and Video Coding, New York, NY, Nov. 1995.

[100] J. Markel and A.H. Gray, “Linear prediction of Speech”, 3rd Edition, New York: Springer Verlag, 1982.

[101] J. McManus and K. Ross, “Video-on-demand over ATM: Constant-rate transmission and transport,” IEEE Journal on Selected Areas in Communications, Vol. 14, pp. 1087-1098, Aug. 1996.

[102] J. Meng and S-F. Chang, “CVEPS: A Compressed Video Editing and Parsing System", ACM Multimedia Conference, Boston, MA, Vol. 2419, Nov. 1996.

[103] J. Meng etal, “Scene Change Detection in a MPEG Compressed Video Sequence", IS&T/SPIE Symposium Proceedings, San Jose, California, Vol. 2419. Feb. 1995.

[104] J. Modestino, D.G. Daut, and A.L. Vickers, "Combined Source-Channel Coding of Images using the Block Cosine Transform", IEEE Transactions on Communications, COM-29, No. 9, Sep. 1981.

[105] R. Mokry and D. Anastassiou, “Minimal Error Drift in Frequency Scalability for Motion-Compensated DCT coding,” IEEE Transactions on Circuits and Systems for Video Technology, Vol. 4, No. 4, pp. 392-406, Aug. 1994.

[106] D. G. Morrison, M. E. Nilsson, and M. Ghanbari, “Reduction of the Bit-Rate of Compressed Video While in its Coded Form,” *Proceedings, Packet Video Workshop*, 1994, pp. D17.1–17.4.

[107] MPEG-2 Systems Standard, ISO/IEC 13818-1 MPEG-2 H.222.0, "Generic coding of moving pictures and associated audio: Systems", Nov. 1994.

[108] MPEG-2 Video Standard, ISO/IEC 13818-2 MPEG-2 H.262, Video, Nov. 1994.

[109] Y. Nakajima, H. Hori, and T. Kanoh, “Rate conversion of MPEG coded video by re-quantization process,” *Proceedings, IEEE Int’l Conf. on Image Processing*, October 1995, Vol. 3, pp. 408–411.

[110] G. Nemhauser and L. A. Wolsey, “Integer and combinatorial optimization”, New York: Wiley, 1988.

[111] A. Netravali and B.G. Haskell, "Digital Pictures: Representation, Compression, and Standards," New York, Plenum Press, 2nd Edition, 1995.

[112] A. Noerpel, Y. Lin, and H. Sherry, "PACS: Personal Access Communications System - a tutorial" IEEE Personal Communications, Vol. 3, No.3, pp. 32-43, Nov. 1996.

[113] A. Ortega, "Optimization techniques for adaptive quantization of image and video under delay constraints," Ph.D. Thesis, Department of Electrical Engineering, Columbia University, New York, NY, 1994.

[114] A. Ortega and M. Khansari, "Rate Control for Video Coding over Variable Bit Rate Channels with Applications to Wireless Transmission", Proc. ICIP, Washington DC, pp. 23-26, Oct. 1995.

[115] A. Ortega, K. Ramchandran, and M. Vetterli, "Optimal trellis-based buffered compression and fast approximations", IEEE Trans. On Image Processing, Vol. 3, No. 1, pp. 26-39, Jan. 1994.

[116] P. Ow and T.E. Morton, "The single machine early tardy problem," Management Science, Vol. 35, pp. 177-191, 1988.

[117] S. Paek and S-F Chang, "Video server retrieval scheduling and resource reservation for variable bit rate scalable video", IEEE Transactions on CSVT, Vol. 10, No. 3, April 2000.

[118] K. Pahlavan and A. Levesque, "Wireless Information Networks", John Wiley and Sons, Inc., 1995.

[119] S. Panwalkar, M.L. Smith, and A. Seidman, "Common due date assignment to minimize total penalty for the one machine scheduling problems," Operations Research, Vol. 30, pp. 391-399, 1982.

[120] Papers in ACM Multimedia Conference, 1993, 1997 and 1998.

[121] V. Parthasarathy, J.W. Modestino and K.S. Vastola, "Design of a Transport Coding Scheme for High-Quality Video over ATM Networks", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 7, No. 2, pp. 358-76, Apr. 1997.

[122] S. Pejhan, M. Schwartz and D. Anastassiou, "Error Control Using Retransmission Schemes in Multicast Transport Protocols for Real-Time Media", IEEE/ACM Transactions on Networking, Vol. 4, No. 3. pp. 413-27, June 1996.

[123] M. Pinedo, "Scheduling Theory, Algorithms, and Systems," Prentice Hall, Englewood Cliffs, N.J. 1995.

[124] A. Puri and T. Chen Eds. "Multimedia Systems, Standard, Networks," In preparation, Marcel Dekker, 1999.

[125] A. Puri etal., "Considerations in ISO MPEG-4 and ITU-T Mobile Video Standards", Proceedings 3rd. International Workshop on Mobile Multimedia Communications (MoMuC-3)", Princeton, NJ, Sep. 1996.

[126] P. Raghavan and C.D. Thompson, "Randomized rounding: provably good algorithms and algorithmic proofs", Combinatorica, Vol. 7 pp. 365-374, 1987.

[127] P. Raghavan, "Probabilistic construction of deterministic algorithms: approximating packing integer programs", Journal of Computer and System Sciences, Vol. 37, pp. 130-143, 1988.

[128] K. Ramchandran, "Joint Optimization Techniques in Image and Video Coding with Applications to Multi-resolution Digital Broadcast," Ph.D. Thesis, Department of Electrical Engineering, Columbia University, New York, NY, 1993.

[129] K. Ramchandran and M Vetterli, "Best wavelet packet in the rate-distortion sense", IEEE Trans. on Image Processing, Vol. 2, No. 2, pp. 160-175, Apr. 1993.

[130] K. Ramchandran etal., "Multi-resolution Broadcast for Digital HDTV Using Joint Source/Channel Coding", IEEE Journal on Selected Areas in Communications, Vol. 11, No. 1, pp. 6-23, Jan. 1993.

[131] K. Ramchandran, A. Ortega, and M. Vetterli, "Bit allocation for dependent quantization with applications to multi-resolution and MPEG video coders", IEEE Trans. On Image Processing, Vol. 3, No. 5, pp. 533-545, Sep 1994.

[132] K.R. Rao and P. Yip, "Discrete Cosine Transform: Algorithms, Advantages, Applications," Academic Press, San Diego, California, 1990.

[133] T. Rappaport, "Wireless Communications, Principles and Practice", Prentice Hall, New Jersey, 1996.

[134] J. Reason et.al. "Asynchronous Video: Coordinated Video Coding and Transport for Heterogeneous Networks with Wireless Access", Mobile Computing, Korth and Imielinski, Kluwer Academic Publishers, Boston, MA, 1995.

[135] R. Rejai, M. Handley, and D. Estrin, "Layered quality adaptation for Internet video streaming," *IEEE Journal on Selected Areas in Communications,* Quality of Service (QoS) in the Internet, Vol. 18, No. 12, December 2000, pp. 2530–2498.

[136] A. Reibman and B. Haskell, “Constraints on Variable Bit Rate Video for ATM Networks” IEEE Transactions on Circuits and Systems for Video Technology, Vol. 2, No. 4, pp. 361-72, Dec. 1992.

[137] K.-C. Roh, K.-D. Seo, and J.-K. Kim, “Data Partitioning and coding of DCT coefficients based on re-quantization for error-resilient transmission of video,” Signal Processing: Image Communication, Vol. 17, No. 8, September 2002, pp. 573–586.

[138] S. Sahni, “Approximate algorithms for the 0/1 knapsack problem”, Journal of the ACM, Vol. 22, pp. 115-124, 1975.

[139] J. Salehi, Z. Zhang, J.J. Kurose, and D. Towsley, “Supporting stored video: reducing rate variability and end-to-end resource requirements through optimal smoothing,” ACM Sigmetrics, pp. 222-231, May 1996.

[140] G. Schuster and A. Katsaggelos, “Rate-distortion based video compression -- Optimal video frame compression and Object Boundary Encoding”, Kluwer Academic Publishers, 1997.

[141] A. Segall, “Bit allocation and entropy for vector sources”, IEEE Trans. On Information Theory, Vol. IT-22, pp. 162-169, Mar. 1976.

[142] K.-D. Seo, S.-C. Heo, and J.-K. Kim, “Adaptive rate control algorithm based on logarithmic R-Q model for MPEG-1 to MPEG-4 transcoding,” Signal Processing: Image Communication, Vol. 17, No. 10, November 2002, pp. 857–876.

[143] T. Shanableh and M. Ghanbari, “Heterogeneous video transcoding to lower spatio-temporal resolutions and different encoding formats,” *IEEE Trans. on Multimedia*, Vol. 2, No. 2, June 2000, pp. 101–110.

[144] Y. Shoham and A. Gersho, “Efficient bit allocation for an arbitrary set of quantizers”, IEEE Trans. on Signal Processing, Vol. 36, pp. 1445-53, Sept. 1988.

[145] H. Song and K. M. Lee, “Adaptive rate control algorithms for low bit rate video under networks supporting bandwidth renegotiation,” Signal Processing: Image Communication, Vol. 17, No. 10, November 2002, pp. 759–780.

[146] Signal Processing: Image Communication, Special Issue on MPEG-4, Parts 1 and 2: Invited and Submitted Papers, Vol. 10, Nos. 1-4, May-July 1997.

[147] J. Signes, Y. Fisher, and A. Eleftheriadis, “MPEG-4: Scene Representation and Interactivity”, pp. 407-447, Multimedia Systems, Standards, and Networks, Eds. A. Puri and T. Chen, Marcel Dekker, Inc. New York.

[148] P. Sinha and A. A. Zoltners, “The Multiple-Choice Knapsack Problem”, Operations Research, Vol. 27, No. 3, May-June 1979.

[149] I. Sodagar, Hung-Ju Lee, P. Hatrack, Ya-Qin Zhang “Scalable wavelet coding for synthetic/natural hybrid images”, IEEE Transactions on Circuits & Systems for Video Technology, vol.9, no.2, March 1999, pp.244-54.

[150] J. Song, A. Dan, and D. Sitaram, “Efficient Retrieval of Composite Multimedia Objects in the *JINSIL* Distributed System,” pp. 260-271, ACM Sigmetrics, Seattle, WA, 1997.

[151] R. Stedman, H. Gharavi, L Hanzo and R. Steele, “Transmission of Sub-Band Coded Images via Mobile Channels", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 3, pp. 15-26, Feb. 1993.

[152] H. Stocker and J.W. Harris, “Handbook of Mathematics and Computational Science,” Springer, New York, NY, 1998.

[153] J. Streit and L. Hanzo, “Quadtree-Based Reconfigurable Cordless Videophone Systems", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 6, No. 2, pp. 225-37, Apr. 1996.

[154] H. Sun, W. Kwok, and J. W. Zdepski, “Architectures for MPEG Compressed Bitstream Scaling,” *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 6, No. 2, April 1996, pp. 191–199.

[155] J. Tajime, Y. Senda, and Y. Miyamoto, “Fast software MPEG-2 video transcoder with optimization of requantization error compensation,” *Proceedings, IEEE Int’l Conf. on Image Processing*, 2002, Vol. 1, pp. 705–708.

[156] R. Talluri etal., "A robust, scalable, object-based video compression technique for very low bit-rate coding", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 7, No. 1, pp. 221-233, Feb. 1997.

[157] D. Taubman and A. Zakhor, “A common framework for rate and distortion based scaling for highly scalable compressed video”, IEEE Transactions on CSVT, Vol. 6, No.4, pp. 329-254, Aug. 1996.

[158] A. Trushkin, “Optimal bit allocation algorithm for quantizing a random vector”, Problems of Information Transmission, Vol. 17, pp. 156-161, 1981.

[159] C.F. Van Loan and G.H. Golub, “Matrix computations,” 3rd Edition, Johns Hopkins University Press, Baltimore, MD, 1996.

[160] A. Vetro and C. W. Chen, “Rate-reduction transcoding design for wireless video streaming,” *Proceedings, IEEE Int’l Conf. on Image Processing*, 2002, Vol.1, pp. 29–32.

[161] O. Werner, “Requantization for transcoding of MPEG-2 intra-frames,” *IEEE Trans. on Image Processing*, Vol. 8, No. 2, February 1999, pp. 179–191.

[162] M. Yong, Q-F. Zhu, and V. Eyuboglu, “VBR Transport of CBR-Encoded Video over ATM Networks,” *Proceedings, Packet Video Workshop*, 1994, pp. D18.1–18.4.

[163] J. Youn and M.T. Sun, “Video Transcoding with H.263 Bit Streams,” *Journal of Visual Communication and Image Representation*, December 2000, pp. 385–404.

[164] L. Yun and D. Messerschmitt, “Variable Quality of Service in Systems by Statistical Power Control", Proc. IEEE ICC, 1995.

[165] E. Zemel, “An *O(n)* algorithm for the linear multiple choice knapsack problem and related problems”, Information Processing Letters, Vol. 18, pp. 123-128, 1984.

[166] W. Zeng and B. Liu, “Rate shaping by block dropping for transmission of MPEG-precoded video over channels of dynamic bandwidth,” *Proceedings, ACM Multimedia Conference*, 1996.

[167] J. Zhang, M.R. Frater, J.F. Arnold, and T.M. Percival, "MPEG-2 Video Services for Wireless ATM Networks", IEEE Journal on Selected Areas in Communications, Vol. 15, No. 1, pp. 119-28, Jan. 1997.